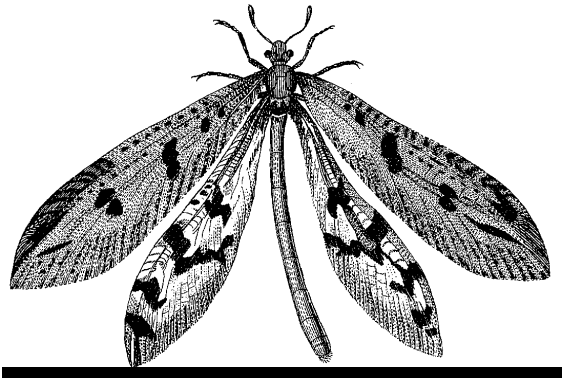
Охватывает все версии Oracle вплоть до Oracle9i

ORACLE

СПРАВОЧНИК

Символ®
O'REILLY®

Рик Гринвальд и Дэвид К. Крейнс



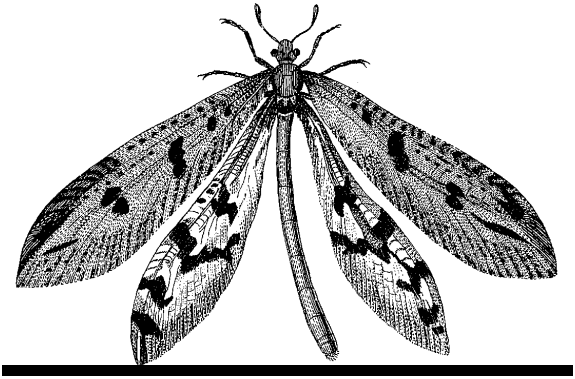
ORACLE

IN A NUTSHELL

A Desktop Quick Reference

Rick Greenwald & David C. Kreines

O'REILLY®



ORACLE

СПРАВОЧНИК

Рик Гринвальд и Дэвид К. Крейнс

Санкт-Петербург — Москва
2005

Рик Гринвальд и Дэвид Крейнс

Oracle. Справочник

Перевод П. Шера

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*А. Королев*
Редактор	*В. Овчинников*
Корректор	*О. Макарова*
Верстка	*Н. Гриценко*

Гринвальд Р., Крейнс Д.
Oracle. Справочник. – Пер. с англ. – СПб: Символ-Плюс, 2005. – 976 с., ил.
ISBN 5-93286-064-2

Oracle появилась четверть века назад и по сей день остается ведущей СУБД масштаба предприятия. Oracle – сложная система, предлагающая несметное множество продуктов, языков и инструментов. Следующие один за другим обновления, релизы и выпуски делают все более сложной задачу пользователя, стремящегося справиться с могучим потоком часто меняющейся информации об этой СУБД и ее возможностях. Задача «Oracle. Справочник» – объединить наиболее важную информацию по архитектуре, синтаксису и пользовательским интерфейсам Oracle. Синтез формы и содержания этой книги от O'Reilly дает лаконичный и очень доступный настольный справочник, содержащий важнейшие команды, конструкции языка, параметры и форматы файлов Oracle.

«Oracle. Справочник» – это кладезь информации, необходимой администраторам БД, разработчикам на PL/SQL и Java, системным и сетевым администраторам, а также специалистам по безопасности, имеющим дело с БД Oracle. Эта книга не раз поможет программистам, использующим как Oracle9*i* версии 2, так и более ранние продукты, при написании кода, работающего с данными Oracle.

ISBN 5-93286-064-2
ISBN 0-596-00336-6 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 04.04.2005. Формат 70x100 1/16 . Печать офсетная.
Объем 61 печ. л. Тираж 2000 экз. Заказ N
Отпечатано с диапозитивов в ГУП «Типографии «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Моему отцу, Роберту Гринвальду.

– Рик Гринвальд

Сюзанне, главной хранительнице единства.

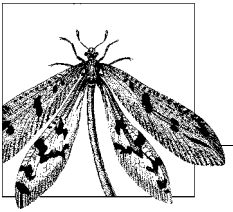
– Дэвид К. Крейнс

Оглавление

Часть II. Языки

Часть III. Инструменты и утилиты

Часть IV. Приложения



Предисловие

Появившись четверть века назад, Oracle остается мировым лидером среди реляционных систем управления базами данных (РСУБД) уровня предприятия. Oracle – это сложная система, предлагающая огромное количество продуктов, языков и инструментов. Регулярно выходят обновления, появляются новые решения и версии, и пользователям нелегко справляться с большим объемом постоянно меняющейся информации по Oracle. Помочь разобраться в ситуации призвано великое множество книг, статей и веб-сайтов. Нужна ли еще одна такая книга?

Цель издания «Oracle. Справочник» в том, чтобы предоставить вам действительно необходимые знания об Oracle в книге, содержимое и формат которой выбраны так, чтобы полезная информация всегда была у вас под рукой. Эта книга – удобный справочник по командам и параметрам основных языков и инструментов Oracle. Мы постарались собрать воедино (и представить по возможности кратко) те сведения, которые необходимы администраторам и разработчикам баз данных для управления СУБД Oracle и создания программного кода.

Как и вся серия «In a Nutshell» издательства O'Reilly, эта книга предназначена тем, кто знает, что хочет сделать, но не может вспомнить нужную команду или значение по умолчанию, соответствующий параметр или диапазон значений, правильный формат заголовка пакета или тип данных. Предполагается, что читатель уже хотя бы немного знаком с обсуждаемыми языками и инструментами. Если вам необходимо руководство, подробное описание применения или особые тонкости, можно обратиться к документации по Oracle и другим более специализированным книгам (большой список книг и других дополнительных ресурсов необходимой информации по вопросам, охваченным в нашей книге, приведен в приложении E). Но хотелось бы думать, что ответы на большинство вопросов вы сможете найти в нашей книге. Надеемся, что она станет бесценным настольным справочником для всех пользователей Oracle.

Несколько слов о том, чем является эта книга и чем не является. Это краткий справочник, необходимый практически всем пользователям Oracle. Здесь есть сведения для администраторов БД, разработчиков, использующих PL/SQL и Java, системных и сетевых инженеров, а также для специалистов по безопасности. Многие главы по большей части состоят из краткой справочной информации (например, перечень параметров инициализации; синтаксис команд SQL и PL/SQL, вызовы функций, заголовки процедур и функций встроенных пакетов; команды и параметры SQL*Plus, SQL*Loader, Import/Export и RMAN; представления словаря данных; интерфейсы и классы Oracle/Java; рекомендации по оптимизации). Но для того чтобы не делать книгу совсем

громоздкой (название серии «In a Nutshell» – «в двух словах» и так уже похоже на насмешку!), решено было не углубляться в другие интересные технические вопросы.

Основная из стоявших перед нами задач оказалась и самой сложной: как создать хороший справочник по Oracle и при этом не сделать его неподъемным? Решено было руководствоваться несколькими правилами:

1. Мы пытались следовать правилу 90/10. Аналогично правилу 80/20 оно гласит, что можно предоставить 90% наиболее важной информации об Oracle на 10% страниц. Данная книга содержит экстракт из 13 000 страниц документации Oracle (для Oracle9*i*), поэтому надеемся, что эта цель достигнута.
2. Мы хотели, чтобы несмотря на отсутствие излишних подробностей, предлагаемая информация не теряла смысла и оставалась полезной. Результат можно увидеть, например, в главе 10: для каждого из множества вызовов встроенных пакетов PL/SQL приведен заголовок и краткое описание, но ничего больше. Если потребуется изучить какой-то конкретный пакет более тщательно, обратитесь за информацией к документации Oracle, но для применения данной функциональности достаточно сведений, приведенных в главе 10.
3. Скрепя сердце, мы оставили лишь минимальное количество примеров. Если приводить пример для каждой команды, оператора и параметра, объем книги возрастет до невозможности. Наиболее заметно отсутствие примеров в главе по SQL (и без того очень длинной). При этом, как уже говорилось, мы надеемся, что книга структурно и содержательно организована так, что те, кому понадобятся какие-то дополнительные сведения, всегда будут знать, где их найти.
4. Везде, где это представлялось возможным, мы старались организовать форму и содержание так, чтобы максимально облегчить восприятие информации. В качестве примера можно рассмотреть параметры инициализации в главе 2, которые приведены не просто как список, упорядоченный по алфавиту, а разбиты на смысловые группы. Как показывает наш опыт, родственные параметры обычно используются вместе, поэтому глава была организована именно так. Кроме того, надеемся, что благодаря такой структуре читатель, обратившись за одним описанием, откроет для себя неизвестные ранее связанные параметры.
5. Наконец, было решено ограничиться только теми вопросами, которые важны для основной массы пользователей Oracle. Поэтому, хоть и неохотно, мы опускаем обсуждение Advanced Queuing, Streams, Advanced Security и множества других узкоспециализированных возможностей. Если бы мы попытались включить в повествование даже самое поверхностное их описание, наша основная цель никогда не была бы достигнута.

Надеемся, что мы приняли правильное решение. Если не согласны – ждем ваших комментариев, но, пожалуйста, будьте великодушны!

Платформа и версия

Oracle может работать на огромном количестве аппаратных и программных платформ. Большая часть информации, представленной к книге, применима к любой платформе. Если какие-то команды или параметры особым образом ведут себя в среде Windows, Linux или какой-то другой, это будет отмечено в тексте.

К моменту выхода этой книги последней версией была СУБД Oracle9*i* Release 2. Версия Oracle9*i* появилась сравнительно недавно, поэтому мы хорошо помним, какие новые возможности в ней появились, а какие исчезли. Во многих разделах будут при-

сутствовать такие комментарии, как «появилось в Oracle9*i*» и «не поддерживается после Oracle8*i*». Надеемся, что эти замечания пригодятся пользователям, работающим со старыми версиями Oracle. О системах более ранних, чем Oracle8, речь не идет.

Структура книги

Справочник состоит из четырех частей:

Часть I «Основы»

Первая часть содержит базовые сведения о СУБД Oracle, которые не зависят от используемых вами языка и инструментов.

- Глава 1 «Архитектура и комплект поставки» предлагает обзор архитектуры и основных компонентов СУБД Oracle и рассказывает о различных решениях Oracle.
- Глава 2 «Конфигурирование» описывает параметры инициализации (параметры в *INIT.ORA* и/или *SPFILE*), отвечающие за настройку вашей базы данных Oracle.
- Глава 3 «Конкурентный доступ» рассказывает о принятой в Oracle многоверсионной модели согласованности по чтению (Multiversion Read Consistency – MVRC) и обсуждает транзакции, блокировки и другие принципы конкурентного доступа к данным.
- Глава 4 «Безопасность» кратко рассказывает об аутентификации пользователей, профилях, привилегиях, ролях и аудите, а также приводит синтаксис команд управления безопасностью СУБД Oracle.
- Глава 5 «Работа в сети» описывает основы сетевых возможностей Oracle и приводит синтаксис необходимых конфигурационных файлов, а именно *TNSNAMES.ORA*, *SQLNET.ORA*, *LISTENER.ORA*, *LDAP.ORA*, *NAMES.ORA* и *CMAN.ORA*.
- Глава 6 «Словарь данных» описывает представления словаря данных Oracle, которые хранят информацию об объектах и пользователях СУБД Oracle; она охватывает как статические представления, так и динамические представления производительности.

Часть II «Языки»

Вторая часть книги посвящена синтаксису команд и функций SQL, программ PL/SQL и Java-интерфейсов для Oracle.

- В главе 7 «SQL» описывается синтаксис Oracle-версии структурированного языка запросов (SQL – Structured Query Language).
- Глава 8 «Функции» приводит синтаксис функций, которые могут быть вызваны из SQL и PL/SQL.
- В главе 9 «PL/SQL» кратко рассмотрены возможности процедурного языка Oracle и описан формат всех его операторов.
- Глава 10 «Пакеты PL/SQL» приводит перечень спецификаций заголовков всех процедур и функций встроенных пакетов Oracle, а также описания параметров.
- В главе 11 «Java и Oracle» рассказывается о Java-интерфейсах к СУБД Oracle, в том числе о драйверах Java для Oracle, о сопоставлении типов данных Java и Oracle и о синтаксисе интерфейсов SQLJ и JDBC.

Часть III «Инструменты и утилиты»

В третьей части представлены команды и спецификации файлов для различных инструментов и утилит, применяемых для управления СУБД Oracle и взаимодействия с ней.

- Глава 12 «SQL*Plus» содержит описание команд и элементов форматирования, доступных пользователю в SQL*Plus – интерфейсе командной строки для Oracle, который предназначен для ввода команд SQL, кода PL/SQL, а также для выполнения файлов сценариев.
- Глава 13 «Экспорт и импорт» содержит перечень команд, предлагаемых утилитами Export (копирование данных из базы данных в двоичный файл) и Import (импорт данных из двоичного файла в базу данных Oracle). Эти утилиты позволяют получать сведения о структуре и содержимом базы данных Oracle.
- В главе 14 «SQL*Loader» описываются команды утилиты SQL*Loader, предназначенной для загрузки данных в стандартных файловых форматах операционной системы в базу данных Oracle и выполнения различных преобразований данных в процессе загрузки.
- Глава 15 «Резервное копирование и восстановление» кратко описывает принципы резервного копирования и восстановления данных Oracle и шаблоны процедур, обеспечивающих управляемое пользователем копирование и восстановление. Приводится перечень команд специальной утилиты Oracle RMAN (Recovery Manager).
- Глава 16 «Enterprise Manager» описывает возможности Enterprise Manager – консоли с графическим интерфейсом пользователя, которая позволяет управлять сервером Oracle.
- Глава 17 «Производительность» рассматривает основные инструменты Oracle, с помощью которых можно оценить и улучшить производительность. Описываются оптимизаторы и подсказки для них в SQL. Приводится синтаксис применения таких средств оптимизации, как Explain Plan, TKPROF, AUTOTRACE, UTLBSTAT, UTLESTAT и Statspack.

Часть IV «Приложения»

В этой части книги содержится сводная и справочная информация.

- Приложение A «Типы данных» содержит перечень типов данных Oracle и правила их преобразования.
- Приложение B «Выражения, операторы и условия» предлагает список разрешенных выражений, операторов и условий, которые можно включать в команды SQL, PL/SQL и SQL*Plus.
- Приложение C «Числовые форматы» приводит форматы чисел, поддерживаемые в командах SQL, PL/SQL и SQL*Plus.
- Приложение D «Форматы даты» приводит форматы дат, поддерживаемые в командах SQL, PL/SQL и SQL*Plus.
- Приложение E «Дополнительные ресурсы» содержит перечень книг и сетевых ресурсов, предлагающих дополнительные сведения по вопросам, изучаемым в данной книге.

Соглашения, принятые в этой книге

Мы будем придерживаться следующих типографских соглашений:

Курсив

Применяется к именам файлов, каталогов, адресов URL, при первом упоминании терминов, иногда для выделения важных понятий.

`Моноширинный шрифт`

Применяется при описании синтаксических конструкций и в примерах кода.

Моноширинный курсив

В синтаксических конструкциях обозначает изменяемый элемент (например, имя файла).

Моноширинный полужирный

Применяется при описании работы пользователя с утилитами (такими как SQL*Plus, RMAN); вводимые пользователем команды выделены полужирным шрифтом, а выводимые данные набираются обычным шрифтом. Также полужирным шрифтом выделены значения, принятые по умолчанию.

ВЕРХНИЙ РЕГИСТР

Употребляется для обозначения ключевых слов при описании синтаксических конструкций.

Нижний регистр

В описании синтаксических конструкций обозначает вводимые пользователем термы, такие как переменные и параметры.

[ ]

В синтаксических конструкциях в квадратные скобки заключены необязательные элементы.

{ }

В синтаксических конструкциях в фигурные скобки помещается список элементов, из которых должен быть выбран только один.

|

В синтаксических конструкциях вертикальная черта разделяет элементы, помещенные в фигурные скобки, например {VARCHAR2 | DATE | NUMBER}.

Таким форматированием выделены советы, предложения и примечания общего характера. Например, указано, что некоторая версия имеет определенные особенности.

Так выделены предупреждения и предостережения. Например, о том, что некоторая операция может иметь нежелательные последствия.

Комментарии и вопросы

Мы протестировали и проверили информацию, содержащуюся в этой книге и в исходных текстах настолько хорошо, насколько это было возможно, но, учитывая то, о каком количестве средств мы рассказываем, и как быстро все меняется, вы можете обнаружить, что некоторые свойства могли измениться, или мы могли допустить ошибки. Обнаружив, сообщите нам, написав по адресу:

O'Reilly & Associates

1005 Gravenstein Highway

Sebastopol, CA 95472

800-998-9938 (для США или Канады)

707-829-0515 (международный или внутренний)

707-829-0104 (факс)

Также вы можете послать сообщение по электронной почте. Чтобы быть занесенным в список рассылки или получить каталог изданий, напишите письмо по адресу:

info@oreilly.com

Чтобы задать технические вопросы или дать комментарии к книге, пишите по адресу:

bookquestions@oreilly.com

У этой книги есть веб-сайт, где вы можете найти примеры программ и список опечаток (найденные ошибки и исправления доступны для просмотра). Адрес этой страницы:

http://www.oreilly.com/catalog/oraclenut

Дополнительная информация об этой и других книгах находится на веб-сайте O'Reilly:

http://www.oreilly.com

Благодарности

Как и следовало ожидать, эта книга появилась на свет благодаря помощи огромного количества людей. Мы очень благодарны всем им.

Во-первых, большое спасибо тем, чьи книги были исходным материалом для построения глав этой книги: Джонатан Генник (Jonathan Gennick), Стивен Фейерштейн (Steven Feuerstein), Билл Прибыл (Bill Pribyl), Чип Дэйвс (Chip Dawes), Брайан Лески (Brian Laskey), Дон Бэйлс (Don Bales), Дарл Кун (Darl Kuhn) и Скотт Шульце (Scott Schulze). Отдельную благодарность хотелось бы высказать Стивену Фейерштейну, нашему другу и герою; Чипу Дэйвсу, который вовремя предоставил нам массу точных данных; и Дону Бэйлсу, чьи терпение и мягкая настойчивость очень помогли в создании разделов, посвященных Java.

Мы также очень признательны нашим рецензентам: Джонатану Геннику (Jonathan Gennick), Санжею Мишра (Sanjay Mishra), Дарлу Куну (Darl Kuhn) и Алану Бьюли (Alan Beaulieu). Они всесторонне и досконально изучили книгу, согласившись при этом работать в бешеном темпе. Берем назад все проклятья, которые мы тихо посылали в их адрес в последние дни подготовки, пытаясь исправить все ошибки, найденные ими, и внести все предложенные ими изменения.

Кроме того, особая благодарность нашему редактору, Деби Рассел (Debby Russell), которая, как всегда, выступала нашим советчиком и безжалостным судебным исполнителем в одном лице; без ее усилий эта книга никогда бы не была опубликована. Спасибо всей выпускающей команде O'Reilly за то, что они превратили эту кучу страниц, синтаксических диаграмм и таблиц в замечательно оформленное единое целое.

Все ошибки и опечатки, которые остались в книге, несмотря на всеобщую помощь, остаются целиком и полностью на совести авторов.

От Рика

Создание книги – это тяжелая и утомительная работа. Чтобы дойти до конца, вам необходима поддержка всех тех многочисленных людей, о которых уже сказали выше. Но, что еще важнее, вам необходима помощь ваших близких. Для меня самые близ-

кие – это ЛуЭнн (LuAnn), Элеанора (Elinor) и Жозефина (Josephine) Гринвальд (Greenwald). ЛуЭнн подарила мне жизнь, полную счастья. Элли и Джози еще не могут даже написать слово Oracle (и не понимают, почему папочка не может поиграть с ними подольше), но в сущности, все, что я делаю, я делаю для них.

На всем протяжении этой большой работы я трудился с полной отдачей. Этот проект помог мне осознать и оценить ту трудовую этику, без которой я себя не мыслю. За это я благодарю моего отца, скончавшегося, когда я работал над этой книгой. Спасибо, папа.

От Дейва

Когда мне впервые пришла в голову мысль о создании справочника по Oracle для серии «in a Nutshell», я обсудил это кое с кем и в волнении отправил заявку Деби Рассел, редактору моих книг, уже вышедших в издательстве O'Reilly. К сожалению, оказалось, что Рик Гринуолд уже представил на рассмотрение макет такой же книги. Но мое разочарование было недолгим, поскольку Рик пригласил меня быть его соавтором. После всех беспокойств и огорчений, которых я причинил Рику в связи с соблюдением сроков, форматированием, содержанием и всем остальным, за что отвечает главный автор, он, может быть, уже пожалел о своем решении! И я хотел бы воспользоваться возможностью и поблагодарить Рика за то, что он включил меня в этот проект. Работы было очень много, и я многому научился. Спасибо, Рик! Я постараюсь компенсировать хотя бы некоторые из этих бессонных ночей!

Мою работу с Oracle поддерживали и поощряли многие друзья и знакомые, а именно Джон Бересневич (John Beresniewicz), Баф Эмсли (Buff Emslie), Стивен Фернстайн, Джонатан Генник, Стив Хейзелдайн (Steve Hazeldine), Кен Якобс (Ken Jacobs), Брайан Лески, Рич Ниемик (Rich Niemiec), Мэтт Рейган (Matt Reagan) и Марлен Терьо (Marlene Theriault). Особая благодарность моим коллегам из Rhodia: Клоду Коэну (Claude Cohen), Деб Ирвин (Deb Irwin), Дейву Флуду (Dave Flood), Рафаэлю Хевиа (Raphael Hevia), Хоакину Лусеро (Joaquin Lucero), Полю Марсу (Paul Mars), Брайану Мак-Мэхону (Brian McMahon), Бин Пэну (Bin Pan) и Кристиану Тайбергену (Christian Tiberghien). Конечно, я упомянул не всех, но те, кто помогал мне, знают, что они это делали.

Наконец, я хотел бы еще раз поблагодарить мою семью за то, что они мирились с тем, что я забросил их ради поисков, сочинительства, проверок и редактирования.

Основы

Первая часть содержит базовые сведения о СУБД Oracle, которые не зависят от используемых вами языка и инструментов.

Глава 1 «Архитектура и комплект поставки» предлагает обзор архитектуры и основных компонентов СУБД Oracle и рассказывает о различных решениях Oracle.

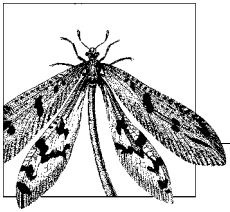
Глава 2 «Конфигурирование» описывает параметры инициализации (параметры в *INIT.ORA* и/или *SPFILE*), отвечающие за настройку вашей базы данных Oracle.

Глава 3 «Конкурентный доступ» рассказывает о принятой в Oracle многоверсионной модели согласованности по чтению (Multiversion Read Consistency – MVRC) и обсуждает транзакции, блокировки и другие принципы конкурентного доступа к данным.

Глава 4 «Безопасность» кратко рассказывает об аутентификации пользователей, профилях, привилегиях, ролях и аудите, а также приводит синтаксис команд управления безопасностью СУБД Oracle.

Глава 5 «Работа в сети» описывает основы сетевых возможностей Oracle и приводит синтаксис необходимых конфигурационных файлов, а именно *TNSNAMES.ORA*, *SQLNET.ORA*, *LISTENER.ORA*, *LDAP.ORA*, *NAMES.ORA* и *CMAN.ORA*.

Глава 6 «Словарь данных» описывает представления словаря данных Oracle, которые хранят информацию об объектах и пользователях СУБД Oracle; в ней рассмотрены как статические представления, так и динамические представления производительности.



1 Архитектура и комплект поставки

Архитектура системы управления базами данных Oracle уникальна. Такие решения, как буферы отката и Real Application Clusters являются частью внутренней архитектуры Oracle, благодаря чему базы данных Oracle предлагают широкий ряд возможностей, недоступных в других СУБД.

В этой главе представлен краткий обзор архитектуры и внутренних компонентов СУБД Oracle, а также описание некоторых «изюминок» продукта корпорации Oracle: многочисленные редакции, версии и основные свойства. Поняв, как Oracle решает свои задачи, вы будете готовы к восприятию предлагаемого в последующих главах материала: параметров инициализации Oracle и представлений словаря данных, разнообразных команд языка, особенностей инструментов и утилит Oracle, а также основных аспектов настройки и оптимизации работы СУБД.

База данных и экземпляр Oracle

Часто понятие «база данных Oracle» употребляют как по отношению к собственно базе данных, так и по отношению к экземпляру, из-за чего возникает путаница.

В Oracle термин *база данных* (*database*) относится к физическому хранилищу информации, тогда как *экземпляр* (*instance*) – это программа, работающая на сервере и обеспечивающая доступ к информации из базы данных. Экземпляр выполняется на компьютере (сервере), база данных хранится на дисках, подключенных к серверу (рис. 1.1).

База данных – это физическая сущность, она состоит из файлов, хранящихся на дисках. Экземпляр же представляет собой логический объект, состоящий из структурированных данных и процессов, расположенных в памяти сервера. Экземпляр может устанавливать соединение с одной и только одной базой данных. Экземпляр имеет ограниченный срок жизни, а вот база данных при надлежащем обслуживании может быть вечной.

Пользователи не обращаются напрямую к информации в базе данных Oracle. Они передают запросы на получение информации экземпляру Oracle.

Состав базы данных

База данных – это набор физических файлов и логических структур, которые будут описаны в последующих разделах.

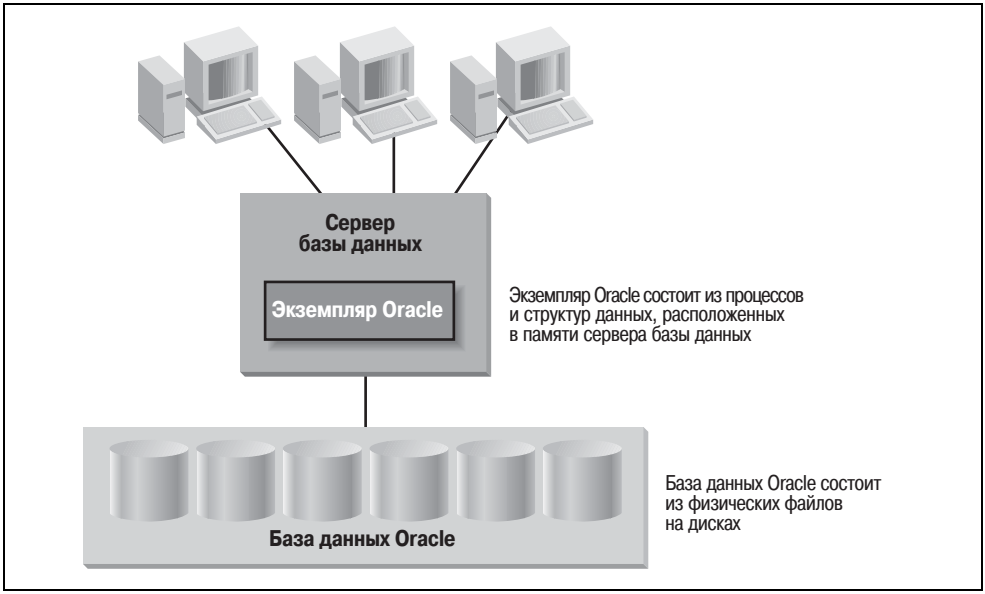


***Рис. 1.1.** Экземпляр и база данных*

У каждой базы данных есть имя, назначаемое ей в момент создания. После того как база создана, изменить ее имя невозможно, хотя можно изменить название экземпляра, обращающегося к базе данных.

Табличное пространство

Табличное пространство (*tablespace*) – это логическая структура, которая существует только в контексте базы данных Oracle. Табличное пространство состоит из физических структур, называемых *файлами данных* (*data files*). При этом табличное пространство может содержать один или несколько файлов данных, а каждый файл данных может принадлежать только одному табличному пространству. При создании таблицы вы указываете табличное пространство, в котором она должна быть создана. Oracle выделит для такой таблицы место в одном из файлов данных, образующих указанное табличное пространство.

Взаимоотношения табличных пространств и файлов данных изображены на рис. 1.2. Представленная на рисунке база данных Oracle содержит два табличных пространства. Создавая в такой базе новую таблицу, ее можно поместить либо в табличное пространство DATA1, либо в DATA2. Физически таблица будет находиться в одном из файлов данных, входящих в выбранное табличное пространство.

Физические файлы базы данных Oracle

Табличное пространство – это логическое представление физического хранения информации базы данных Oracle, которое образуется тремя основными типами файлов:

- Управляющие файлы
- Файлы данных
- Журнальные файлы

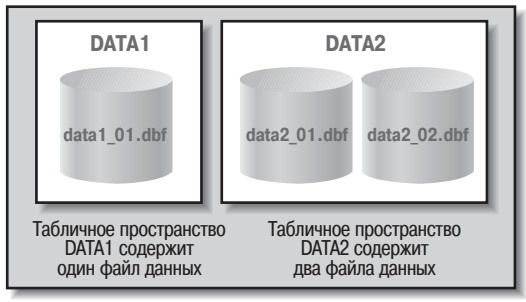


Рис. 1.2. *Табличные пространства и файлы данных*

В среде базы данных используются и другие виды файлов, такие как файлы паролей и файлы инициализации экземпляров, перечисленные же выше файлы трех базовых типов представляют собой саму физическую базу данных. Эти файлы, составляющие основу БД, и их взаимосвязи изображены на рис. 1.3.

В Oracle9*i* введено понятие файла, управляемого Oracle (Oracle Managed File – OMF). Если вы хотите использовать OMF, необходимо задать параметры инициализации DB_CREATE_FILE_DEST и DB_CREATE_ONLINE_LOG_DEST_*n* (эти, а также все остальные параметры инициализации Oracle подробно описаны в главе 2). Если выбрана работа с OMF, то сервер Oracle9*i* автоматически создает, именует и удаляет (при необходимости) все файлы вашей базы данных Oracle. Управляемые файлы призваны сократить накладные расходы на создание и отслеживание имен вашей БД

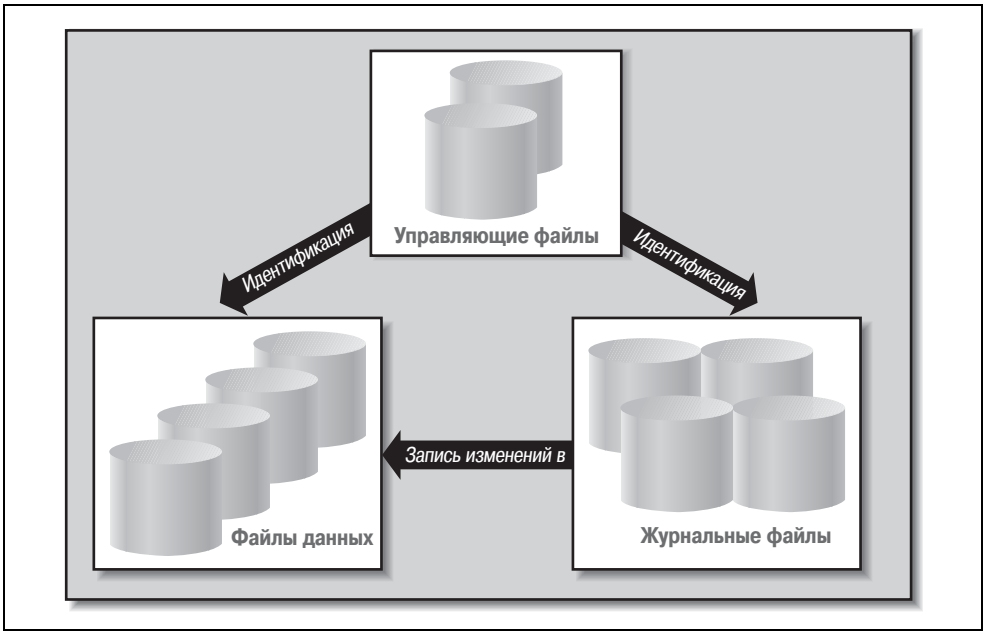


Рис. 1.3. *Файлы, составляющие базу данных*

Oracle. Они также позволяют избежать возникающих из-за человеческого фактора проблем, вызванных неправильной идентификацией файлов базы данных Oracle.

Следующие разделы будут посвящены базовым типам файлов и их взаимодействию.

Управляющие файлы

Управляющий файл (control file) содержит список всех остальных файлов, составляющих базу данных, таких как файлы данных и журнальные файлы. Кроме того, в нем присутствует информация о содержимом и состоянии базы данных, а именно:

- Имя базы данных
- Время создания базы данных
- Текущее состояние файлов данных: необходимо ли им восстановление, доступны ли они только для чтения и т. д.
- Информация о том, не произошло ли ошибок при последней остановке базы данных
- Период времени, охватываемый каждым архивным журнальным файлом
- Сведения о том, какие резервные копии базы данных были созданы

До версии Oracle8 размер управляющих файлов обычно не превышал одного мегабайта. Но в Oracle8 в управляющем файле появляется дополнительная информация, а именно подробные сведения о резервном копировании БД. Поэтому управляющие файлы сервера БД версии Oracle8 и выше вполне могут занимать более 10 Мбайт. Размер управляющего файла зависит от множества параметров инициализации, в том числе MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES и MAXINSTANCES.

Можно сделать так, чтобы экземпляр Oracle поддерживал несколько копий управляющих файлов. Теоретически вы можете восстановить управляющий файл в случае его повреждения или удаления, но фактически этот процесс занимает много времени, к тому же в некоторых случаях корректное восстановление состояния управляющего файла невозможно. Запустить экземпляр Oracle без управляющего файла нельзя, поэтому наличие нескольких копий такого файла является важной мерой безопасности. Местоположение копий управляющего файла указывается в параметре инициализации CONTROL_FILES.

Файлы данных

Файлы данных содержат реальные данные, хранящиеся в базе, в том числе: таблицы и индексы, создаваемые пользователями БД; словарь данных, содержащий информацию об этих структурах данных; и сегменты отката, используемые для поддержания внутренней непротиворечивости данных Oracle.

Файл данных состоит из блоков БД Oracle, которые в свою очередь образованы из блоков дискового пространства операционной системы. Блок Oracle имеет размер от 2 до 32 Кбайт. Если ваша версия Oracle поддерживает VLM (Very Large Memory) – сверхбольшую оперативную память, то возможна работа с большими блоками Oracle – BOB (big Oracle block), размер которых может достигать 64 Кбайт.

До выхода версии Oracle9*i* размер блока устанавливался единым для всей базы данных. В Oracle9*i* также задается размер блока по умолчанию для базы данных, но при этом разрешено использовать в БД блоки пяти различных нестандартных размеров. Каждый файл данных поддерживает только один размер блока, но в рамках базы данных возможны блоки нескольких размеров.

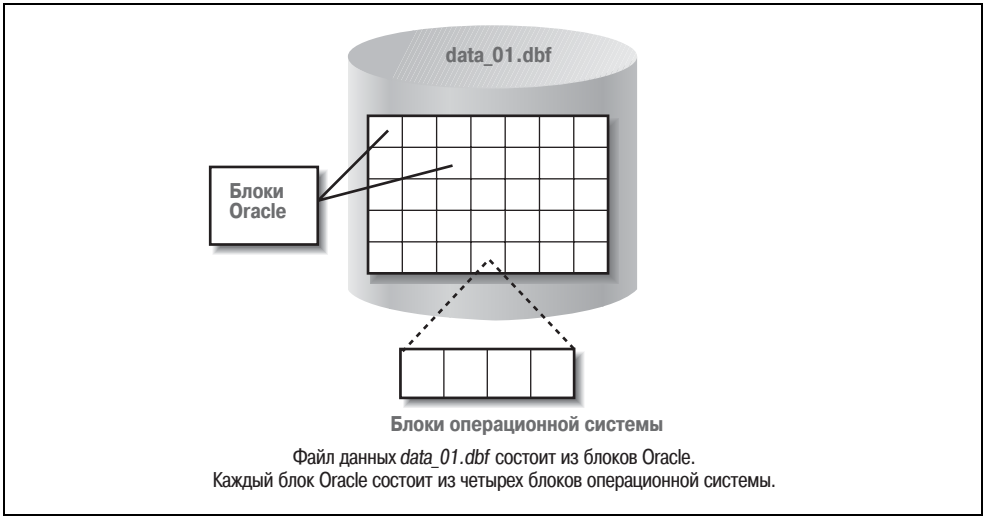


Рис. 1.4. *Блоки Oracle и блоки операционной системы*

Связь между блоками Oracle и блоками дискового пространства операционной системы представлена на рис. 1.4.

Файлы данных принадлежат только одной базе данных и только одному табличному пространству внутри базы данных. По мере необходимости данные поблочно считываются из файлов данных в оперативную память (в блоках Oracle). Точно так же по мере необходимости блоки данных записываются из оперативной памяти в файлы данных на диске, что обеспечивает отображение сделанных пользователями изменений в базе данных.

Файлы данных находятся на нижнем уровне детализации базы данных Oracle по отношению к операционной системе. При размещении базы данных в подсистеме ввода/вывода наименьшей физической порцией данных, которая может быть помещена в определенное место, является файл данных. Настройка подсистемы ввода/вывода с целью достижения наибольшей производительности Oracle обычно требует перемещения файлов данных с одних дисков на другие.

Структура файла данных

Первый блок любого файла данных называется *заголовком файла данных* (*datafile header*). Он содержит информацию, необходимую для поддержания общей целостности базы данных. Одним из наиболее существенных элементов заголовка является *контрольная точка* (*checkpoint*), логическая временная метка, которая показывает последний момент, когда изменения были записаны в базу данных. Такая метка в заголовке необходима для процесса восстановления БД: по ней определяется, какие журнальные файлы следует применить к файлу данных в текущий момент времени.

Экстенты и сегменты

С физической точки зрения файл данных хранится в виде блоков операционной системы. С логической же точки зрения существуют три промежуточных уровня организации файла данных: блоки данных, экстенты и сегменты. *Экстент* (*extent*) –

это набор смежных блоков данных файла данных Oracle. *Сегмент* (*segment*) – это совокупность экстентов, содержащих все данные некоторого объекта базы данных Oracle, такого как таблица или индекс.

Обновляя данные, сервер Oracle пытается сделать это в том же самом блоке данных. Если в блоке недостаточно места для новой информации, сервер Oracle запишет данные в новый блок данных, который может относиться к другому экстенту.

Журнальные файлы

Журнальные файлы регистрируют изменения, произведенные в базе данных в результате выполнения транзакций и внутренней активности Oracle. При нормальной работе Oracle кэширует измененные блоки в оперативной памяти, и если в экземпляре происходит фатальная ошибка, то может оказаться, что какие-то измененные блоки не были записаны в файлы данных. Тогда можно использовать хранящиеся в журнале записи для воспроизведения изменений, которые были утеряны при сбое.

Мультиплексирование журнальных файлов

В Oracle для описания работы с журналами принята специальная терминология. Каждый экземпляр Oracle записывает производимые им в базе данных изменения в журналы. Журнал может быть один, но их может быть и несколько, тогда говорят о *журнальной группе* (*redo log group*) и *элементах журнальной группы* (*redo log members*).

Логически можно воспринимать журнальную группу как единый журнальный файл. Однако Oracle может работать с несколькими копиями журнала для защиты от сбоев носителей. Несколько копий одного журнала группируются вместе и образуют журнальную группу. Все журнальные группы экземпляра называются *потоком журнала* (*redo thread*).

Существуют способы восстановления статической части управляющего файла в случае потери данных, но утраченный журнальный файл восстановить невозможно, так что позаботьтесь о поддержании нескольких его копий!

Как Oracle использует журнальные файлы

Заполнив один журнальный файл, Oracle автоматически переходит к следующему. Заполнив все доступные журнальные файлы, сервер возвращается к первому и пишет в него повторно. Для отслеживания журналов Oracle применяет последовательную нумерацию. После заполнения некоторого журнального файла и перехода к следующему сервер увеличивает внутренний счетчик, называемый *порядковым номером журнала* (*redo log sequence number*). Этот порядковый номер записывается в текущий журнальный файл. Oracle отслеживает эти внутренние номера для соблюдения правильного порядка обхода журналов, хотя имя повторно используемого файла может совпадать с именем журнала, заполненного ранее.

Архивные журнальные файлы

Читая предыдущий раздел, вы вполне могли задаться вопросом о том, как избежать потери важнейшей информации журнала при повторном обращении Oracle к уже заполненному журналу.

Есть два способа решения этой проблемы. Первый чрезвычайно прост: вы просто не думаете о возможности утраты информации в случае сбоя со всеми вытекающими последствиями. При перезаписи журнала хранящаяся в нем история изменений теряется. Если в результате сбоя будут повреждены файлы данных, то можно будет вос-

становить всю базу данных в состояние на момент последнего резервного копирования. Но воспроизвести изменения, сделанные позже последнего резервного копирования, не удастся, поскольку журнала не существует. Не повезло! Таким путем идут очень немногие компании, работающие с Oracle, ведь невозможность восстановить ситуацию на момент сбоя недопустима – в результате теряются данные.

Второй (и более распространенный) ответ на вопрос о повторном использовании журналов заключается в архивировании заполненных журналов. Для того чтобы понять, что такое архивирование журналов, необходимо знать, что Oracle на самом деле поддерживает два типа журналов:

Оперативные журналы

Файлы операционной системы, в которые Oracle последовательно записывает изменения, сделанные в базе данных.

Архивные журналы

Копии заполненных оперативных журналов, создаваемые во избежание потери данных при перезаписи оперативных журналов.

Сервер Oracle обеспечивает два режима архивирования журналов:

NOARCHIVELOG

Как видно из названия, в этом режиме журналы не архивируются. По мере обхода заполненные журналы повторно инициализируются и перезаписываются, при этом история изменений базы данных стирается. Именно о таком режиме шла речь в первом ответе на вопрос: в этом случае сбой приводит к потере данных.

Тем, кто выбирает работу без архивирования журналов, будет доступно значительно меньшее количество вариантов и параметров резервного копирования базы данных.

ARCHIVELOG

Переходя к новому журналу, сервер Oracle архивирует предыдущий. Для того чтобы не допустить появления пропусков в истории изменений, повторное заполнение журнала начинается только после его успешного архивирования. Архивные журналы и текущий оперативный журнал содержат полную историю изменений, внесенных в базу данных. Вместе они позволяют серверу восстановить все зафиксированные транзакции вплоть до того момента, когда произошел сбой.

Для того чтобы включить режим ARCHIVELOG, следует разрешить архивирование журналов, выполнив в SQL*Plus команду ALTER DATABASE ARCHIVELOG, и присвоить параметру инициализации LOG_ARCHIVE_START значение TRUE. Архивирование журналов будет выполняться в файлах с именами, заданными параметром LOG_ARCHIVE_FORMAT, расположенных в каталоге, указанном в параметре LOG_ARCHIVE_DEST.

Компоненты экземпляра

Экземпляр Oracle можно определить как область разделяемой памяти и набор фоновых процессов.

Область разделяемой памяти экземпляра называется *системной глобальной областью* (System Global Area – SGA). Фактически SGA не является одной большой однородной областью памяти, она состоит из различных компонентов, которые будут описаны в следующем разделе. Все процессы экземпляра, как системные, так и пользовательские, совместно обращаются к SGA.

До версии Oracle9*i* размер SGA устанавливался при запуске экземпляра Oracle. Единственным способом изменения размера SGA или какой-то ее составляющей было изменение соответствующих параметров инициализации, остановка и перезапуск экземпляра. В Oracle9*i* можно изменять размер SGA и ее компонентов, не останавливая экземпляр.

Фоновый процесс взаимодействует с операционной системой и другими фоновыми процессами, управляя структурами памяти экземпляра. Эти процессы также управляют реальной базой данных на диске и выполняют общие действия по обслуживанию экземпляра.

В состав экземпляра входят и другие физические файлы:

Файл инициализации экземпляра

Файл инициализации содержит множество разнообразных параметров, которые определяют, как будет работать экземпляр: сколько оперативной памяти ему будет отведено, какому количеству пользователей одновременно будет разрешено устанавливать соединение, к какой базе данных экземпляр будет обеспечивать доступ и т. д. Многие из этих параметров можно впоследствии изменить динамически на уровне системы или же сеанса. До версии Oracle9*i* файл инициализации назывался *INIT.ORA*.[1] В Oracle9*i* появился файл *SPFILE*, который выполняет ту же функцию, что и *INIT.ORA*, а кроме того сохраняет изменения параметров инициализации, которые были выполнены в процессе работы Oracle9*i*. Обратитесь к документации по вашей операционной системе, чтобы узнать, где именно по умолчанию расположен файл *INIT.ORA* вашей системы.

Файл конфигурации экземпляра

Файл конфигурации, называемый *CONFIG.ORA*, – это файл необязательных параметров, который включается, если требуется выделить некоторое множество параметров (например, параметры для Oracle Parallel Server или Real Application Clusters).

Файл паролей

Сервер Oracle может использовать необязательный файл паролей (который хранится как файл операционной системы) для обеспечения дополнительной гибкости при управлении базами данных Oracle. Это зашифрованный файл, содержащий идентификаторы и пароли, которые могут применяться при решении административных задач, таких как остановка и запуск экземпляра. Файл паролей – это стандартный способ обеспечения безопасности удаленного доступа в дополнение к мерам безопасности операционной системы. Защитные меры на уровне операционной системы обычно бывают локальными (т. е. реализуются на сервере БД). Например, в Unix-системе любой пользователь из группы DBA может запустить и остановить сервер Oracle – операционная система предоставляет пользователю такие права. Достоверность пароля (сравнить со значением, хранящимся для пользователя в базе данных) можно проверить, только если база данных открыта. В случае наличия файла паролей пользователь должен удостоверить свою личность, чтобы запустить базу данных.

Если вы используете некоторые специальные возможности базы данных, такие как Shared Server/Multi-Threaded Server (разделяемый сервер/многопоточный сервер), очереди заданий или репликацию, могут возникать дополнительные фоновые процессы.

---

[1] Конкретное имя файла *INIT.ORA* зависит от имени вашего экземпляра (подробная информация приводится в главе 2).

Структуры оперативной памяти экземпляра

В действительности SGA состоит из четырех основных областей (рис. 1.5): кэша *буферов базы данных* (*database buffer cache*), *разделяемого пула* (*shared pool*), *журнального буфера* (*redo log buffer*) и *большого пула* (*large pool*)

Кэш буферов базы данных

Кэш буферов базы данных кэширует извлеченные из базы блоки данных. Такой буфер между пользовательскими запросами и реальными файлами данных повышает производительность СУБД Oracle. Если часть данных присутствует в кэше буферов, ее можно извлечь из оперативной памяти без затрат на обращение к диску. Oracle управляет кэшем на основе алгоритма LRU (Least Recently Used) – алгоритма удаления элементов, не использовавшихся дольше всего. Другими словами, если пользователь запрашивает данные, обращение к которым осуществлялось недавно, то вероятнее всего, что они содержатся в кэше буферов, и их можно получить, не выполняя операцию чтения с диска.

В Oracle7 был один пул буферов для блоков базы данных. В Oracle8 появились новые пулы буферов: теперь их может быть три и более:

DEFAULT

Стандартный кэш буферов базы данных Oracle. Если специально не указано иное, то этот кэш используют все объекты.

KEEP

Для часто используемых объектов, которые хотелось бы кэшировать.

RECYCLE

Для объектов, к которым вы вряд ли обратитесь вновь.

Пулы буферов KEEP и RECYCLE исключают свои объекты, применяя алгоритм LRU.

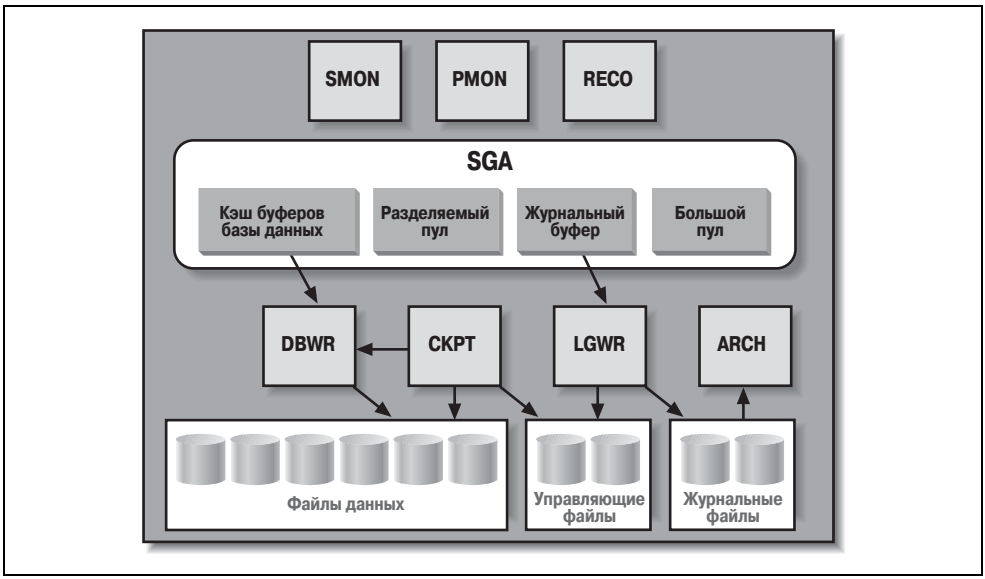


Рис. 1.5. *Экземпляр Oracle*

Можно указать, что некоторая таблица или индекс должны кэшироваться в определенном пуле буферов. Это обеспечивает хранение наиболее нужных объектов и предотвращает борьбу всех объектов за место в одном центральном кэше.

Разделяемый пул

Разделяемый пул кэширует различные конструкции, которые могут использоваться совместно. Например, выдаваемые пользователями команды SQL кэшируются, чтобы их можно было выполнить вновь в случае, если снова будет передана такая же команда. Еще одним примером могут быть хранимые процедуры – фрагменты кода, хранимые и выполняемые в базе данных. Они загружаются в разделяемый пул для исполнения и кэшируются опять-таки в соответствии с алгоритмом LRU. Разделяемый пул также служит для кэширования информации из *словаря данных* Oracle, т. е. метаданных (данных, описывающих структуры данных), которые предоставляют информацию о структуре и содержимом самой базы данных.

Журнальный буфер

Журнальный буфер кэширует журнальную информацию до тех пор, пока она не будет записана в физические журнальные файлы, расположенные на диске. Этот буфер улучшает производительность. Сервер Oracle кэширует журнальную информацию для того, чтобы записать ее на диск в наиболее удобное время, а не осуществлять запись журналов на диск постоянно.

Большой пул

Большой пул, появившийся в Oracle8, – это необязательная область SGA, предназначенная для буферизации ввода/вывода различных серверных процессов, в том числе применяемых для резервного копирования и восстановления. Область также служит для хранения памяти сеанса Multi-Threaded Server и в случае протокола XA для распределенных транзакций.

Фоновые процессы экземпляра

На рис. 1.5 показаны следующие фоновые процессы:

Процесс записи в базу данных (Database Writer – DBWR)

Процесс записывает блоки базы данных из кэша буферов SGA в файлы данных, расположенные на диске. При необходимости для экземпляра Oracle может существовать до 10 процессов DBWR (с именами DBW0-DBW9), обрабатывающих ввод/вывод в различные файлы данных. Большинство экземпляров обходится одним процессом DBWR. DBWR записывает блоки из кэша на диск по двум основным причинам:

- Для установки контрольной точки. *Контрольная точка* (*checkpoint*) – это техническое понятие, обозначающее обновление блоков файлов данных, с тем чтобы они «догнали» журналы. Сервер Oracle записывает журнальную информацию для транзакции после ее фиксации, а затем записывает актуальные блоки. Периодически сервер Oracle устанавливает контрольную точку для того, чтобы привести содержимое файла данных в соответствии с журнальной информацией, записанной для зафиксированных транзакций.
- Для освобождения пространства кэша. Если серверу Oracle необходимо прочитать запрошенные пользователем блоки в кэш, а в кэше буферов нет свободного места, то вызывается процесс DBWR, который записывает несколько бло-

ков в БД, освобождая пространство в кэше. Запись блоков осуществляется в соответствии с алгоритмом LRU – такой порядок позволяет минимизировать влияние отсутствия блоков в кэше на производительность.

Процесс записи в журнал (Log Writer – LGWR)

Процесс записывает информацию из журнального буфера SGA во все копии текущего файла журнала на диске. Пока транзакция обрабатывается, соответствующая журнальная информация хранится в журнальном буфере SGA. Когда же транзакция зафиксирована, Oracle отправляет журнальную информацию на постоянное хранение, вызывая процесс LGWR для записи ее на диск.

Системный монитор (System Monitor – SMON)

Процесс отвечает за общую исправность и безопасность экземпляра. SMON выполняет восстановление экземпляра при запуске после сбоя. Он же координирует доступ и реализует восстановление экземпляра, когда к базе данных обращаются сразу несколько экземпляров, например для Oracle Parallel Server/Real Application Clusters. SMON также приводит в порядок смежные элементы свободного пространства в файлах данных, объединяя их вместе, и избавляется от пространства, используемого для сортировки строк, когда в нем уже нет необходимости.

Монитор процессов (Process Monitor – PMON)

Процесс следит за пользовательскими процессами, обращающимися к базе данных. В случае аварийного прекращения пользовательского процесса PMON занимается очисткой оставшихся занятыми ресурсов (таких как оперативная память) и снятием всех блокировок, установленных сбойным процессом.

Архиватор (Archiver – ARCH)

Процесс читает файлы журнала, заполненные сервером Oracle, и записывает копию использованных журнальных файлов в один или несколько архивных каталогов. Версии Oracle8*i* и выше поддерживают до 10 процессов ARCH, которым присваиваются имена ARC0 – ARC9. По мере необходимости LGWR запускает дополнительные архиваторы в зависимости от нагрузки. Максимально разрешенное количество процессов архивации задается параметром инициализации LOG_ARCHIVE_MAX_PROCESSES.

Контрольная точка (Checkpoint – CKPT)

Процесс устанавливает контрольные точки с помощью DBWR. CKPT обновляет заголовки управляющих файлов и файлов данных для сохранения сведений о последней контрольной точке после того, как она завершена.

Процесс восстановления (Recover – RECO)

Процесс автоматически очищает неудавшиеся и отложенные распределенные транзакции.

Версии Oracle

Версии Oracle, как и комплектация продукта, о которой мы поговорим в следующем разделе, – это больше маркетинговая, чем техническая категория. Выпуск новых версий СУБД Oracle, как и любых программных продуктов, происходит с определенной периодичностью, но точная дата и содержимое выпуска зависит не только от цикла разработки, но и от ситуации на рынке. Из-за присутствия таких нетехнических факторов действующие соглашения по именованию версий могут измениться в любой момент.

К моменту написания этой книги корпорация Oracle, похоже, выработала однозначную систему наименования версий. Вплоть до Oracle8 основные версии получали последовательно возрастающие номера (например, Oracle6, Oracle7, Oracle8). Однако после Oracle8 появилась версия Oracle8*i*, имя которой указывало на ее связь с Интернетом (и с этого момента вызывало у тех, кто о ней писал, трудности с форматированием!). Следующая версия называлась Oracle9*i*, а ожидаемая вскоре версия должна называться Oracle10*i*, хотя к моменту ее выпуска все еще может измениться.

Для версий Oracle*Ni* характерно наличие промежуточного выпуска, который обычно получал имя Release 2 (выпуск 2). Вторые выпуски обычно служат поддержкой для первых, основных версий: они часто содержат расширения и усовершенствования функциональности, которые планировались в основном выпуске, но не были готовы в срок.

В этой книге за точку отсчета принимается Oracle8. Функции, которые были добавлены или изъяты после выхода Oracle8, соответственно помечаются.

Комплектация Oracle

На комплектацию продукта рыночные требования влияют еще сильнее, чем на выпуск новых версий (вероятно, даже больше, чем технические факторы). Решение о том, включить ли некоторую возможность в одну или несколько версий СУБД или же рассматривать определенную функциональность как отдельную, продаваемую за дополнительную плату опцию, часто является случайным.

Варианты поставки

Существуют четыре базовых варианта поставки сервера Oracle:

Standard Edition

Самая недорогая версия сервера Oracle, которая поддерживает не все возможности СУБД Oracle. Описанные в следующем разделе дополнительные опции не доступны в Standard Edition.

Enterprise Edition

Полная версия сервера, содержащая дополнительные опции, описанные в следующем разделе.

Personal Edition

Однопользовательская версия сервера Oracle, выпускаемая только для Windows. Эти версия поддерживает все опции Enterprise Edition, когда это возможно.

Lite

Урезанная версия сервера Oracle для мобильного использования.

Компания Oracle иногда принимает решение включить ранее предлагавшиеся за отдельную плату опции в один или несколько стандартных комплектов, но изъятия функциональности, присутствовавшей ранее, с целью принудить пользователей к переходу на более дорогой вариант еще не случалось.

Большая часть возможностей, описанных в книге, поддерживается любым вариантом сервера Oracle. Приведем функции, которые на момент создания книги были доступны в Enterprise Edition и недоступны в Standard Edition:

Защита данных (Data Guard)

Предоставляет набор программ и инструментальные средства управления, которые упрощают создание резервных баз данных.

Средства быстрого восстановления (Fast-start recovery)

Реализует технологию восстановления экземпляра после сбоя, позволяющую открыть базу данных сразу же после отката по журнальным файлам.

Оперативное создание и объединение индексов (Online index build and coalesce)

Позволяет выполнять операции создания и объединения индексов, не прекращая работы пользователей.

Оперативная реорганизация и переопределение таблиц (Online table reorganization and redefinition)

Позволяет выполнять реорганизацию и переопределение таблиц, не прекращая работу пользователей.

Поблочное восстановление носителя (Block-level media recovery)

Разрешает восстановление отдельных блоков носителя вместо восстановления целого файла.

Инкрементное резервное копирование и восстановление (Incremental backup and recovery)

Позволяет выполнять резервное копирование и восстановление тех данных, которые были изменены с момента последнего полного или инкрементного резервного копирования. Такой подход позволяет значительно снизить время, затрачиваемое на подобные операции.

Параллельное резервное копирование и восстановление (Parallel backup and recovery)

Разрешает распараллеливаемые операции резервного копирования и восстановления, что может значительно сократить время их выполнения.

Восстановление табличного пространства по времени (Tablespace point-in-time recovery)

Дает возможность восстанавливать табличное пространство на определенный момент времени. Такая возможность может применяться для восстановления данных из поврежденного журнального файла или для восстановления на момент, непосредственно предшествовавший возникновению ошибки.

Опытное восстановление (Trial recovery)

Позволяет проверить достоверность резервного копирования, не выполняя полный процесс.

Виртуальная частная база данных (Virtual Private Database – VPD)

Позволяет управлять доступом к строкам таблиц на основе результата выполнения процедуры обеспечения безопасности. Такая возможность позволяет предоставлять доступ к строке в зависимости от значения данных в ней (наряду с другими технологиями).

Активный аудит (Fine-grained auditing)

Разрешает осуществлять аудит, основываясь на содержимом строки.

Менеджер ресурсов базы данных (Database Resource Manager – DRM)

Позволяет ограничить объем ресурсов базы данных, выделенных одному пользователю или группе пользователей.

Битовый индекс и битовый индекс соединения (Bitmapped index and bitmapped join index)

Поддерживает дополнительные типы индексов, которые могут повысить производительность запросов, характерных для хранилищ данных.

Автоматический выбор степени параллелизма запроса (Automated parallel query degree)

Дает серверу Oracle возможность выбрать наилучшую степень параллелизма для конкретного запроса.

Параллельные операции (Parallel operations)

Ускоряет выполнение параллельных операций (например, запроса, загрузки, анализа данных, оптимизации запроса типа «звезда», DML, создания и просмотра индекса) за счет использования нескольких процессов.

Экспорт перемещаемых табличных пространств (Export of transportable tablespaces)

Поддерживает перемещаемые табличные пространства – переносить табличные пространства становится так же просто, как копировать файл. Данный метод может быть гораздо эффективнее, чем процесс импорта-экспорта.

Расширенные возможности репликации (Advanced replication)

Разрешает репликацию изменившихся данных из одного экземпляра базы данных в другой. Данная возможность позволяет создавать пользовательские методы решения конфликтов.

Дополнительные компоненты

Oracle предлагает ряд дополнительных компонентов, которые можно приобрести в составе Oracle Enterprise Edition. Состав, содержимое, стоимость и доступность этих компонентов могут меняться.

На момент написания книги имеются следующие компоненты (в Oracle9*i* Release 2):

Real Application Clusters

Позволяет использовать кластеризованную базу данных, расположенную на нескольких серверах.

Секционирование (Partitioning)

Позволяет распределить данные по нескольким дискам для повышения производительности.

Аналитическая обработка в реальном времени (OLAP – OnLine Analytical Processing)

Содержит ряд аналитических функций, которые можно использовать для оперативной аналитической обработки.

Глубинный анализ данных (Data Mining)

Содержит набор функций глубинного анализа данных для создания приложений бизнес-интеллекта.

Обработка пространственных данных (Spatial)

Разрешает использование и обработку данных пространственной информации, например в географических информационных системах (ГИС).

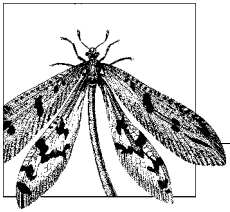
Расширенные возможности защиты (Advanced Security)

Предоставляет возможности усиленного шифрования и аутентификации.

Метки безопасности (Label Security)

Позволяет установить детальный контроль доступа к данным.

Кроме того, Oracle поддерживает ряд пакетов расширения, которые могут использоваться для расширения функциональности продукта Enterprise Manager. О таких пакетах можно прочитать в главе 16.



2

Конфигурация

СУБД Oracle спроектирована как гибкая и легко настраиваемая система. Эти качества абсолютно необходимы СУБД, способной работать на десятках аппаратных платформ в самых разнообразных конфигурациях, поддерживающей бесчисленное разнообразие приложений и пользователей. Для достижения подобной гибкости СУБД Oracle должна предоставлять администратору базы данных простой способ задания характеристик функционирования, обеспечивающий ясность и непротиворечивость. Большинство таких характеристик администраторы определяют, задавая значения *параметров инициализации* базы данных, часто называемых *параметрами INIT.ORA*. Основная задача заключается в выборе таких значений параметров, при которых достигается максимальная производительность СУБД.

Каждый из параметров инициализации отвечает за определенный аспект работы сервера Oracle. Взятые в совокупности, они адаптируют базовую технологию Oracle к конкретным условиям.

Вот типичный пример файла инициализации для универсальной конфигурации Oracle9*i*:

```
DB_NAME = "ORA9"
DB_DOMAIN = homeserver
INSTANCE_NAME = ORA9
SERVICE_NAMES = ORA9.homeserver

DB_FILES = 1024
DB_BLOCK_SIZE = 8192

COMPATIBLE = 9.0.0
SORT_AREA_SIZE = 65536
SORT_AREA_RETAINED_SIZE = 65536

CONTROL_FILES = ("C:\Oracle\oradata\ORA9\control01.ctl",
"D:\Oracle\oradata\ORA9\control02.ctl",
"E:\Oracle\oradata\ORA9\control03.ctl")

OPEN_CURSORS = 100
CURSOR_SHARING = similar

MAX_ENABLED_ROLES = 30
DB_FILE_MULTIBLOCK_READ_COUNT = 8
DB_BLOCK_BUFFERS = 2048
```

```
SHARED_POOL_SIZE = 19728640
LARGE_POOL_SIZE = 614400
JAVA_POOL_SIZE = 25971520

LOG_CHECKPOINT_INTERVAL = 10000
LOG_CHECKPOINT_TIMEOUT = 1800

PROCESSES = 200
PARALLEL_MAX_SERVERS = 5
LOG_BUFFER = 32768
MAX_DUMP_FILE_SIZE = 10240  # ограничить размер файла трассировки 5 Мбайт
GLOBAL_NAMES = true

ORACLE_TRACE_COLLECTION_NAME = ""
BACKGROUND_DUMP_DEST = D:\Oracle\admin\ORA9\bdump
RESOURCE_MANAGER_PLAN = system_plan
USER_DUMP_DEST = D:\Oracle\admin\ORA9\udump
TRACEFILE_IDENTIFIER = ORA9

REMOTE_LOGIN_PASSWORDFILE = exclusive
OS_AUTHENT_PREFIX = ""

PLSQL_COMPILER_FLAGS = debug
UNDO_MANAGEMENT = auto
```

В этом примере показаны лишь некоторые из основных параметров файла инициализации. В конкретной операционной системе наверняка придется добавить другие параметры, и, скорее всего, в файл будут добавлены комментарии, описывающие назначение параметров конфигурации, их влияние на вашу среду и историю их изменения. Такие комментарии должны предваряться символом #, как это сделано в приведенном примере для параметра MAX_DUMP_FILE_SIZE.

Основу данной главы составляет полный список параметров инициализации, но начинается она с краткого описания этих параметров и файлов, в которых они хранятся.

Файлы и типы параметров

Имя и расположение файла параметров инициализации базы данных зависит от версии Oracle и операционной системы, подробнее об этом рассказано в следующих разделах.

INIT.ORA: файл инициализации

В версиях, предшествующих Oracle9*i*, параметры инициализации указываются в файле *INIT.ORA*. Обычно имя этого файла имеет форму *INITsid.ORA*, где *sid* – это системный идентификатор (SID) соответствующего экземпляра Oracle. Это значение служит уникальным именем экземпляра, по которому тот может быть однозначно идентифицирован.

Расположение этого файла по умолчанию зависит от выбранной операционной системы и указано в соответствующей документации по Oracle.

SPFILE: файл параметров сервера

В Oracle9*i* вводится понятие файла параметров сервера. Этот файл, называемый *SPFILE*, имеет ряд отличий от стандартного *INIT.ORA*:

- Имеет двоичный, а не текстовый формат.
- Может сохранять измененные параметры в промежутках между остановкой и запуском экземпляра.

Последний пункт особенно важен. Если вы работаете с Oracle9*i*, то любые изменения параметров конфигурации, выполненные командой ALTER SYSTEM, будут сохранены в постоянном файле конфигурации. Это означает, что если в процессе настройки базы данных изменяется значение какого-либо параметра, то не приходится вносить соответствующие исправления и в файл *INIT.ORA* (или в несколько таких файлов), чтобы сделать новые значения постоянными. Существует и другой способ динамического изменения параметров, при котором они не включаются в файл *SPFILE* – для этого следует включить в команду ALTER SYSTEM инструкцию SCOPE, имеющую такой формат:

```
ALTER SYSTEM SET имя_параметра = значение_параметра
   SCOPE = {MEMORY | SPFILE | BOTH};
```

Полное описание синтаксиса команды ALTER SYSTEM приведено в главе 7.

В Oracle9*i* по-прежнему можно использовать локальный файл *INIT.ORA*, указав его расположение в инструкции PFILE=*имя* команды STARTUP. Кроме того, Oracle9*i* предлагает простой способ переноса параметров экземпляра из существующего файла *INIT.ORA* в двоичный файл *SPFILE*. Для этого надо выполнить следующую команду, заменив строку *путь* на реальный путь к файлу:

```
CREATE SPFILE FROM PFILE='путь/initsid.ora';
```

Параметр SPFILE, который может быть указан в файле *INIT.ORA*, позволяет указать расположение файла *SPFILE*, отличное от принятого по умолчанию; для этого в файл *INIT.ORA* необходимо добавить строку такого вида:

```
SPFILE=$ORACLE_HOME/dbs/spfile.ora;
```

Динамически изменяемые параметры

Большинство параметров инициализации статические и получают свои значения из файла инициализации в момент запуска экземпляра БД, но есть и такие, которые могут быть изменены при запущенном экземпляре и открытой базе данных.

Динамическое изменение – это не то же самое, что динамическое сохранение измененных значений в файле *SPFILE*. Во всех версиях Oracle параметры могут быть изменены командами ALTER SYSTEM или ALTER SESSION. Начиная с Oracle9*i* такие изменения можно динамически сохранять в файле *SPFILE* в зависимости от значения инструкции SCOPE команды ALTER.

В описании отдельных параметров в последующих разделах будет отмечено, какие из них могут изменяться динамически (пункт «Динамический» в описании параметра) и как они могут быть изменены (командой ALTER SYSTEM, ALTER SESSION или обеими).

Параметры инициализации

Оставшаяся часть этой главы описывает параметры инициализации Oracle. Для удобства восприятия имена параметров выделены прописными буквами, но в файле параметров они могут быть набраны в любом регистре, в том числе и вперемешку.

Кроме параметров инициализации, описанных в этой главе и применимых в большинстве Oracle-систем, для некоторых аппаратных платформ и операционных систем могут существовать дополнительные, специфичные для них параметры. Эти параметры описаны в руководствах «Installation Guide» (Руководство по установке), «User Guide» (Руководство пользователя) и/или «Release Notes» (Комментарии к версии), входящих в комплект документации к вашей версии Oracle.

Параметры, представленные в следующем разделе, сгруппированы по функциональным категориям, внутри категорий они упорядочены по алфавиту. Такой способ организации выбран потому, что часто приходится настраивать некоторую функциональность, такую как аудит или управление заданиями, при помощи группы параметров. Если же потребуется найти определенный параметр, обратитесь к алфавитному указателю, где в разделе «параметры инициализации» приведен полный список параметров.

Вот перечень категорий:

Аудит

Резервное копирование и восстановление

Кластеризованные базы данных

Курсоры

Связи БД

Распределенные операции и гетерогенные сервисы

Ввод/вывод и управление пространством

Java

Задания (jobs)

Лицензии

Блокировки и транзакции

Протоколирование и архивирование

Распределение памяти

Имена

Поддержка национальных языков (National Language Support – NLS)

Оптимизация и производительность

Параллельное выполнение

Параметры

PL/SQL

Удаленные узлы

Управление откатом (отменой/восстановлением)

Безопасность

Разделяемый сервер/Многопоточный сервер (Shared Server/Multi-Threaded Server – MTS)

Сортировки

Резервные базы данных

Системные операции

Трассировка (Oracle Trace)

Прочие параметры

Для каждого параметра приведены следующие данные (если таковые имеются):

- Допустимые значения параметра. Имейте в виду, что в значениях многих параметров допускаются сокращения: K (килобайты), M (мегабайты), G (гигабайты)
- Значение по умолчанию
- Возможность динамического изменения параметра и применяемая для этого команда (ALTER SESSION, ALTER SYSTEM)
- Синтаксис определения параметра (приведен только для сложных определений)
- Ключевые слова, которые можно указывать при задании значения параметра
- Краткое описание, включающее при необходимости сведения о версиях и/или вариантах поставки Oracle, в которых используется параметр

Некоторые параметры поддерживаются не во всех версиях Oracle, описанных в этой книге (Oracle8, Oracle8*i* и Oracle9*i*). Данное ограничение также отмечено в их описании.

Аудит

Следующие параметры определяют способ выполнения аудита в базе данных Oracle.

AUDIT_FILE_DEST

Допустимые значения: действительный путь к каталогу

Значение по умолчанию: *$ORACLE_HOME/RDBMS/AUDIT*

Определяет каталог для хранения файлов аудита.

AUDIT_TRAIL

Допустимые значения: NONE | FALSE | DB | TRUE | OS

Значение по умолчанию: NONE

Разрешает и запрещает запись в журнал аудита. SQL-команда AUDIT позволяет задавать режим аудита независимо от значения этого параметра.

Ключевые слова

NONE	Записи в журнал не добавляются
FALSE	Поддерживается для обратной совместимости. Равносильно NONE
DB	Разрешает аудит системы в целом и направляет записи в журнал аудита базы данных (таблицу SYS.AUD$)
TRUE	Поддерживается для обратной совместимости. Равносильно DB
OS	Разрешает аудит системы в целом и направляет записи в журнал аудита операционной системы

TRANSACTION_AUDITING

Допустимые значения: TRUE | FALSE

Значение по умолчанию: TRUE

Динамическое изменение: ALTER SYSTEM DEFERRED

Определяет, генерирует ли *уровень обработки запросов* (*transaction layer*) специальную запись в журнале с информацией о пользователе и сеансе.

Резервное копирование и восстановление

Следующие параметры определяют конфигурацию резервного копирования и восстановления и управляют их режимами. Родственные параметры описаны в разделах «Протоколирование и архивирование» и «Управление откатом (отменой/восстановлением)».

BACKUP_DISK_IO_SLAVES

Допустимые значения: 0–15

Значение по умолчанию: 0

Определяет количество серверных процессов ввода/вывода, используемых диспетчером восстановления (Recovery Manager) для резервного и обычного копирования или восстановления данных на диске. Исключен в версии 8.1.

BACKUP_TAPE_IO_SLAVES

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Динамическое изменение: ALTER SYSTEM DEFERRED

Определяет, обращается ли к серверным процессам ввода/вывода Диспетчер восстановления (Recovery Manager) для резервного и обычного копирования или восстановления данных на ленту.

FAST_START_IO_TARGET

Допустимые значения: 0 | 1000 – все буферы в кэше

Значение по умолчанию: все буферы в кэше

Динамическое изменение: ALTER SYSTEM

Определяет количество буферов операций ввода/вывода, задействуемых при аварийном завершении или восстановлении экземпляра. Меньшие значения заставляют процесс записи в БД чаще записывать содержимое «грязных» буферов на диск, чтобы количество буферов ввода/вывода при восстановлении не превышало заданного значения (ценой падения общей производительности). Значение 0 блокирует установку контрольных точек при быстром восстановлении. Параметр появился в Oracle8*i* и доступен только для Enterprise Edition.

Начиная с Oracle9*i* компания Oracle вместо этого параметра рекомендует FAST_START_MTTR_TARGET.

FAST_START_MTTR_TARGET

Допустимые значения: 0–3600

Значение по умолчанию: 0

Динамическое изменение: ALTER SYSTEM

Определяет желаемое максимальное время восстановления экземпляра в секундах. Работает аналогично параметру FAST_START_IO_TARGET, но игнорируется, если определен один из параметров FAST_START_IO_TARGET и LOG_CHECKPOINT_INTERVAL. Появился в Oracle9*i*.

FAST_START_PARALLEL_ROLLBACK

Допустимые значения: HIGH | LOW | FALSE

Значение по умолчанию: LOW

Динамическое изменение: ALTER SYSTEM

Определяет максимальное количество процессов, выполняющих параллельный откат. Значение FALSE запрещает параллельный откат, LOW и HIGH устанавливают количество равным 2*CPU_COUNT и 4*CPU_COUNT соответственно. Появился в Oracle9*i*.

FREEZE_DB_FOR_FAST_INSTANCE_RECOVERY

Допустимые значения: TRUE | FALSE

Значение по умолчанию: см. описание

Динамическое изменение: ALTER SYSTEM

Параметр компонента Parallel Server, определяющий необходимость блокирования базы данных на время восстановления экземпляра. При значении TRUE сервер Oracle приостанавливает работу базы данных на время восстановления, прекращая всякую дисковую активность, не связанную с восстановлением экземпляра, что позволяет закончить его быстрее. При значении FALSE сервер Oracle не блокирует всю базу данных, за исключением случая, когда происходит восстановление зеркальной копии файлов данных. Если для всех оперативных файлов данных устанавливаются *хеш-блокировки* (*hash locks*), то значением по умолчанию будет FALSE. Если хотя бы для одного из файлов данных устанавливаются *точечные блокировки* (*fine-grained locks*), по умолчанию принимается значение TRUE. Значение должно быть одинаковым для всех экземпляров. Исключен в Oracle8*i*.

RECOVERY_PARALLELISM

Допустимые значения: 0 – PARALLEL_MAX_SERVERS

Значение по умолчанию: зависит от операционной системы

Определяет количество процессов, участвующих в восстановлении экземпляра или носителя. Появился в Oracle9*i*.

Кластеризованные базы данных

Следующие параметры применяются в Oracle Parallel Server (до версии Oracle9*i*) или в Real Application Clusters (начиная с Oracle9*i*).

CLUSTER_DATABASE

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Определяет доступность компонента Real Application Clusters. Должен иметь значение TRUE для всех экземпляров, входящих в Real Application Cluster. Появился в Oracle9*i*.

CLUSTER_DATABASE_INSTANCES

Допустимые значения: целое положительное число

Значение по умолчанию: 1

Определяет количество экземпляров, входящих в Real Application Cluster. Должен иметь одинаковое значение для всех экземпляров в кластере. Появился в Oracle9*i*.

CLUSTER_INTERCONNECTS

Допустимые значения: один или несколько действующих IP-адресов, разделенных двоеточиями (:)

Определяет IP-адреса дополнительных соединений в Real Application Cluster, позволяющих увеличить производительность больших кластеров. Появился в Oracle9*i*.

DELAYED_LOGGING_BLOCK_CLEANOUTS

Допустимые значения: TRUE | FALSE

Значение по умолчанию: TRUE

Включает и выключает отложенную очистку блока. Позволяет уменьшить количество опросов блоков в Oracle Parallel Server. Значение TRUE соответствует выбору быстрого пути и приводит к тому, что во время фиксации транзакции протоколирование очистки блока не выполняется. Исключен в Oracle8*i*.

DRS_START

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Динамическое изменение: ALTER SYSTEM

Определяет, запускается ли процесс аварийного восстановления (Disaster Recovery Process – DRMON). Необходим только в том случае, когда экземпляр входит в конфигурацию аварийного восстановления.

GC_DEFER_TIME

Допустимые значения: любое положительное целое число

Значение по умолчанию: 10

Динамическое изменение: ALTER SYSTEM

Определяет время ожидания (в сотых долях секунды) перед принудительной записью блоков, запрошенных другими экземплярами. Нулевое значение отключает данную возможность. Исключен в Oracle9*i*.

GC_FILES_TO_LOCKS

Допустимые значения: см. описание

Параметр компонента Parallel Server, управляющий преобразованием PCM-блокировок (Parallel Cache Management – управление параллельными кэшами) в блокировки файлов данных. Параметр позволяет ограничить количество блокировок файла данных, а также указать, что каждая блокировка должна распространяться на ряд смежных блоков. Если параметр указан, Oracle не использует технологию Cache Fusion компонента Oracle9*i* Real Application Clusters. Для всех экземпляров значения параметра должны совпадать.

Синтаксис

```
GC_FILES_TO_LOCKS = "{список_файлов=количество_блокировок[!блоки][EACH]}[:]..."
```

Ключевые слова

список_файлов	Один или несколько файлов данных, заданных их номерами или диапазонами номеров, разделенных запятыми
количество_блокировок	Количество PCM-блокировок, назначенное списку *список_файлов*. Если *количество_блокировок* установлено в 0, для указанных файлов применяются точечные блокировки.
!блоки	Необязательный параметр, определяющий количество смежных блоков, охватываемых одной блокировкой. По умолчанию используются несмежные блоки.
EACH	Необязательный параметр, указывающий, что каждому файлу данных из *списка_файлов* назначен отдельный набор PCM-блокировок (в количестве *количество_блокировок*).

GC_LCK_PROCS

Допустимые значения: 1–10 или 0 (для отдельного экземпляра в монопольном режиме)
Значение по умолчанию: 1

Устанавливает количество фоновых процессов блокировок (от LCK0 до LCK9) для экземпляра, входящего в Parallel Server. Как правило, достаточным является значение 1, принятое по умолчанию, но его можно увеличить в случае большого количества запросов на распределенные блокировки. Для всех экземпляров значения параметра должны совпадать.

GC_RELEASABLE_LOCKS

Допустимые значения: 50–88
Значение по умолчанию: Значение параметра DB_BLOCK_BUFFERS

Определяет значение, задаваемое при распределении пространства памяти для точечных блокировок. Максимальное значение ограничено только объемом доступной

памяти. Применяется только для разделяемого режима Oracle Parallel Server. Исключен в Oracle9*i*.

GC_ROLLBACK_LOCKS

Допустимые значения: 1–8

Значение по умолчанию: 20

Для каждого из сегментов отката Parallel Server определяет количество распределенных блокировок, доступных одновременно изменяемым блокам сегмента отката. Значение по умолчанию подходит для большинства приложений. Значения для всех экземпляров должны совпадать. Исключен в Oracle9*i*.

Синтаксис

```
GC_ROLLBACK_LOCKS = 'список_сегментов = количество_блокировок[!блоки][R][EACH][:...]'
```

Ключевые слова

список_сегментов	Один или несколько сегментов отката, заданных их номерами или диапазонами номеров, разделенных запятыми.
количество_блокировок	Количество PCM-блокировок, назначенное списку *список_сегментов*.
!*блоки*	Необязательный параметр, определяющий количество смежных блоков, охватываемых одной блокировкой. По умолчанию блоки несмежные.
R	Указывает на то, что блокировки являются освобождаемыми и по мере необходимости извлекаются из пула освобождаемых блокировок.
EACH	Необязательный параметр, указывающий, что каждому сегменту отката из *списка_сегментов* назначен отдельный набор PCM-блокировок (в количестве *количество_блокировок*).

INSTANCE_GROUPS

Допустимые значения: строка названий групп, разделенных запятыми

Значение по умолчанию: нет

Приписывает текущий экземпляр в указанные группы. Это параметр компонента Real Application Clusters/Parallel Server (см. PARALLEL_INSTANCE_GROUP в разделе «Параллельное исполнение»).

INSTANCE_NAME

Допустимые значения: строка, содержащая имя экземпляра

Значение по умолчанию: системный идентификатор (SID) текущего экземпляра

Задает имя экземпляра. Полезен главным образом в компонентах Real Application Clusters и Parallel Server, где может потребоваться указать, через какой именно экземпляр следует соединяться с БД. Появился в Oracle8*i*.

INSTANCE_NUMBER

Допустимые значения: от 1 до максимального номера экземпляра из команды CREATE DATABASE
Значение по умолчанию: наименьший доступный номер

Задает уникальный номер, сопоставляющий экземпляру одну группу списков свободных блоков для каждой таблицы, созданной при установленном параметре хранения FREELIST GROUPS. Параметр INSTANCE команды ALTER TABLE ALLOCATE EXTENT определяет экстент для заданной группы списков свободных блоков. Если INSTANCE_NUMBER устанавливается в значение, указанное для параметра INSTANCE, то экземпляр использует этот экстент для операций вставки и обновления, которые расширяют строки. INSTANCE_NUMBER относится к параметрам компонента Parallel Server. Значения для всех экземпляров должны быть различными.

LM_LOCKS

Допустимые значения: от 512 до объема доступной разделяемой памяти
Значение по умолчанию: 12 000

Определяет количество блокировок, которое будет установлено для диспетчера блокировок (Lock Manager) при работе с Parallel Server. Количество блокировок вычисляется по следующей формуле:

```
L = R + (R*(N - 1))/N
```

где:

R – количество ресурсов

N – общее количество узлов

L – общее количество блокировок

Значения для всех экземпляров должны совпадать. Исключен в Oracle9*i*.

LM_PROCS

Допустимые значения: от 36 до (PROCESSES + максимальное количество экземпляров + запас прочности)
Значение по умолчанию: 64 + максимальное поддерживаемое количество экземпляров

Параметр компонента Parallel Server, который задает сумму значения параметра PROCESSES и максимального количества экземпляров. Значения для всех экземпляров должны совпадать. Исключен в Oracle8*i*.

LM_RESS

Допустимые значения: от 256 до объема доступной разделяемой памяти
Значение по умолчанию: 6000

Параметр компонента Parallel Server, задающий количество ресурсов, которые могут быть заблокированы каждым экземпляром диспетчера блокировок. Значение, выбранное для параметра LM_RESS, должно быть значительно меньше, чем удвоенное значение параметра DML_LOCKS плюс накладные расходы (примерно 20 блокировок). Значения для всех экземпляров должны совпадать. Исключен в Oracle9*i*.

LOG_FILE_NAME_CONVERT

Допустимые значения: символьная строка (см. описание)

Значение по умолчанию: нет

Преобразует имя нового журнального файла основной базы данных в имя журнального файла резервной базы данных. Файл резервной базы данных должен существовать и быть перезаписываемым, иначе процесс восстановления остановится с ошибкой.

Синтаксис

```
LOG_FILE_NAME_CONVERT = [()'строка1','строка2'[,'строка1,строка2...][)]
```

Ключевые слова

строка 1	образец имен журнальных файлов основной базы данных
строка 2	образец имен журнальных файлов резервной базы данных

MAX_COMMIT_PROPAGATION_DELAY

Допустимые значения: 0–90000

Значение по умолчанию: 90000

Указывает максимальный допустимый промежуток времени (в сотых долях секунды) до начала обновления системного номера обновления (System Change Number – SCN), хранящегося в SGA экземпляра, процессом записи в журнал (LGWR). Он определяет, должен ли локальный SCN обновляться из значения блокировки при получении моментальной копии SCN для запроса.

Этот параметр подлежит изменению лишь в редких специфических ситуациях, присущих Oracle Parallel Server. Значения параметра должны совпадать для всех узлов кластера.

OPS_ADMIN_GROUP

Допустимые значения: действительное имя группы

Значение по умолчанию: все активные экземпляры

Позволяет разделить входящие в Parallel Server экземпляры в целях администрирования и мониторинга. База данных должна быть смонтирована в режиме Parallel Server. Значение параметра OPS_ADMIN_GROUP определяет, какие экземпляры возвращают информацию при запросе к представлению GV$. Исключен в Oracle8*i*.

PARALLEL_SERVER

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Указывает, что для данного экземпляра должна быть разрешена работа Parallel Server. Значения для всех экземпляров должны совпадать. Исключен в Oracle9*i*.

PARALLEL_SERVER_IDLE_TIME

Допустимые значения: от 0 до определяемого в зависимости от операционной системы

Значение по умолчанию: зависит от операционной системы

Определяет длительность периода простоя (в минутах), по истечении которого сервер Oracle завершает серверный процесс запроса. Исключен в Oracle8*i*.

PARALLEL_SERVER_INSTANCES

Допустимые значения: от 0 до определяемого в зависимости от операционной системы

Значение по умолчанию: зависит от операционной системы

Указывает количество экземпляров, используемых для масштабирования структур SGA в среде Parallel Server. Доступен только в Oracle8*i*.

Курсоры

Применение курсоров регулирует следующие параметры:

CLOSE_CACHED_OPEN_CURSORS

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Указывает, должны ли открытые и кэшированные PL/SQL курсоры автоматически закрываться при каждой команде COMMIT. Если курсоры PL/SQL часто используются повторно, то значение FALSE может ускорить последующие выполнения. TRUE означает, что открытые курсоры будут закрываться при каждой команде COMMIT или ROLLBACK, затем при необходимости курсор будет открыт заново. Исключен в Oracle8*i*.

CURSOR_SHARING

Допустимые значения: SIMILAR | EXACT | FORCE

Значение по умолчанию: EXACT

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Определяет, каким образом команды SQL могут совместно использовать курсоры в оперативной памяти. Значение SIMILAR появилось в Oracle9*i*.

Ключевые слова

SIMILAR Означает, что команды SQL, отличающиеся некоторыми литералами, будут совместно использовать курсоры в оперативной памяти, если только различающиеся литералы не изменяют смысла команд SQL или степень оптимизации плана выполнения.

EXACT Указывает, что совместно использовать один курсор в оперативной памяти будут только полностью идентичные команды SQL.

FORCE Означает, что команды SQL, различающиеся некоторыми литералами, будут совместно использовать курсоры в оперативной памяти, если только различающиеся литералы не изменяют смысла команд SQL.

CURSOR_SPACE_FOR_TIME

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Управляет выделением оперативной памяти, отведенной для хранения курсоров. Если значение параметра равно TRUE, то СУБД занимает больший объем памяти для экономии времени. Но разделяемые области SQL никогда не освобождают выделенную им память в процессе исполнения, поэтому параметр следует устанавливать в TRUE, только если разделяемый пул достаточно велик для одновременного хранения всех открытых курсоров. Кроме того, значение TRUE сохраняет закрытую область SQL, выделенную для каждого курсора, между исполнениями вместо того, чтобы очищать ее после выполнения курсора, что экономит время, которое было бы затрачено на размещение и инициализацию курсоров.

OPEN_CURSORS

Допустимые значения: 1–4294967295

Значение по умолчанию: 50

Определяет максимально допустимое количество одновременно открытых курсоров для сеанса, что предотвращает открытие их чрезмерного количества. Если задать слишком большое значение, дополнительных накладных расходов не возникнет. Кроме того, параметр ограничивает размер кэша курсора PL/SQL, используемого во избежание повторного синтаксического анализа команд.

ROW_CACHE_CURSORS

Допустимые значения: 10–3300

Значение по умолчанию: 10

Указывает максимальное количество кэшированных рекурсивных курсоров, используемых диспетчером словаря данных для выбора строк из словаря данных. Исключен в Oracle8*i*.

SERIAL_REUSE

Допустимые значения: DISABLE | SELECT | DML | PLSQL | ALL | NULL

Значение по умолчанию: DISABLE

Определяет, каким типам курсоров SQL должна быть доступна возможность последовательного повторного использования памяти. Помещает закрытую область памяти курсора в разделяемый пул SGA, чтобы она могла повторно использоваться сеансами, исполняющими тот же курсор.

Ключевые слова

DISABLE	Запрещает данную возможность для всех типов команд SQL. Данное значение перекрывает любые другие значения из списка.
SELECT	Разрешает данную возможность для операторов SELECT.
DML	Разрешает данную возможность для операторов DML.
PLSQL	В настоящее время не действует.

ALL	Разрешает данную возможность для операторов DML и SELECT.
NULL	Аналогично DISABLE.

SESSION_CACHED_CURSORS

Допустимые значения: от 0 до определяемого в зависимости от операционной системы

Значение по умолчанию: 0

Динамическое изменение: ALTER SYSTEM

Указывает максимальное количество кэшируемых курсоров сеанса. Повторный разбор одной и той же команды SQL приводит к тому, что курсор сеанса такой команды помещается в кэш курсоров сеанса. Разбор последующих вызовов не требует повторного открытия курсора.

Связи базы данных

Параметры данного раздела описывают использование связей базы данных (*DB links*) с удаленными базами данных.

DBLINK_ENCRYPT_LOGIN

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Указывает, следует ли шифровать пароли при попытках соединения с другими серверами Oracle через связи БД. Если значение параметра равно TRUE и соединение разрывается, то сервер Oracle не осуществляет повторных попыток установления соединения. Если же задано значение FALSE, сервер Oracle повторяет попытку соединения, не шифруя пароль.

OPEN_LINKS

Допустимые значения: 0–255

Значение по умолчанию: 4

Определяет максимальное количество одновременно открытых в рамках одного сеанса соединений с удаленными базами данных. Значение должно быть не меньше количества баз данных, упомянутых в одной команде SQL, ссылающейся на несколько БД, чтобы можно было открыть все эти базы данных и выполнить команду. Может предотвратить лишние накладные расходы на повторное открытие связей БД при последующих попытках доступа.

OPEN_LINKS_PER_INSTANCE

Допустимые значения: 0–4294967295

Значение по умолчанию: 4

Указывает максимальное количество перемещаемых открытых соединений. В XA-транзакциях перемещаемые открытые соединения позволяют кэшировать соединения после фиксации транзакции. Параметр OPEN_LINKS_PER_INSTANCE отлича-

ется от OPEN_LINKS тем, что последний указывает количество соединений для сеанса и не применим для XA-приложений.

Распределенные операции и гетерогенные сервисы

Приведенные ниже параметры управляют распределенными операциями и гетерогенными сервисами (HS) Oracle.

COMMIT_POINT_STRENGTH

Допустимые значения: 0–255

Значение по умолчанию: зависит от операционной системы

Задает значение, определяющее *узел завершения транзакции* (*commit point site*) в распределенных транзакциях. Узел транзакции с наибольшим значением параметра COMMIT_POINT_STRENGTH и будет узлом завершения транзакции. Приоритет узла завершения транзакции для базы данных должен устанавливаться в соответствии с объемом критических совместно используемых данных в этой БД.

DISTRIBUTED_LOCK_TIMEOUT

Допустимые значения: 1–8

Значение по умолчанию: 60

Указывает период времени в секундах, в течение которого распределенные транзакции ожидают освобождения заблокированных ресурсов. Исключен в Oracle8*i*.

DISTRIBUTED_RECOVERY_CONNECTION_HOLD_TIME

Допустимые значения: 0–1800

Значение по умолчанию: 200

Задает период времени в секундах, в течение которого удаленное соединение удерживается открытым после аварийного завершения распределенной транзакции. (Соединение удерживается в надежде на восстановление связи, чтобы не устанавливать его повторно). Можно указывать значения, превышающие 1800 секунд, но, поскольку фоновые процессы перестройки соединений и восстановления запускаются каждые 30 минут (1800 секунд) независимо от факта сбоя, величина 1800 означает, что соединение никогда не будет закрыто. Исключен в Oracle8*i*.

DISTRIBUTED_TRANSACTIONS

Допустимые значения: 0–TRANSACTIONS

Значение по умолчанию: TRANSACTIONS * 0,25

Указывает максимальное количество распределенных транзакций, в которых может одновременно участвовать база данных. Значение данного параметра не может превышать значение параметра TRANSACTIONS. Если значение параметра DISTRIBUTED_TRANSACTIONS равно 0, то распределенные транзакции запрещены и процесс восстановления не запускается при запуске экземпляра.

HS_AUTOREGISTER

Допустимые значения: TRUE | FALSE

Значение по умолчанию: TRUE

Динамическое изменение: ALTER SYSTEM

Определяет, разрешена ли автоматическая саморегистрация агентов гетерогенных сервисов. Компания Oracle рекомендует устанавливать значение этого параметра равным TRUE, т. к. это снижает накладные расходы при открытии последовательных соединений через одного и того же агента. Появился в Oracle8*i*.

MAX_TRANSACTION_BRANCHES

Допустимые значения: 1–32

Значение по умолчанию: 8

Регулирует количество ветвей распределенной транзакции, разрешая работу вплоть до 32 серверов или групп серверов в одной распределенной транзакции (для одного экземпляра).

Ввод/вывод и управление пространством

Параметры, описанные в данном разделе, настраивают и регулируют операции ввода/вывода и управление пространством.

DB_BLOCK_CHECKING

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Определяет, проводится ли проверка на поврежденность блоков данных перед началом записи. Установка этого параметра в TRUE снижает производительность более чем на 10%, поэтому применять его следует с осторожностью. Появился в Oracle8*i*.

DB_BLOCK_SIZE

Допустимые значения: 2048–32768 в большинстве случаев, но может быть и меньше в зависимости от операционной системы

Значение по умолчанию: зависит от операционной системы

Указывает размер блоков БД Oracle в байтах. Значение DB_BLOCK_SIZE при создании базы данных определяет размер блоков. Если используется Real Application Cluster в Oracle9*i*, то значение этого параметра влияет на максимальное значение параметра хранения FREELISTS для таблиц и индексов.

Это значение может быть установлено только в момент создания базы данных и не должно изменяться впоследствии.

DB_FILE_DIRECT_IO_COUNT

Допустимые значения: зависит от операционной системы

Значение по умолчанию: 64

Определяет количество блоков, задействованных в операциях ввода/вывода, которые выполняются при резервном копировании, восстановлении или функциями прямого чтения и записи. Размер буфера ввода/вывода равняется произведению значений параметров DB_FILE_DIRECT_IO_COUNT и DB_BLOCK_SIZE и не может превышать максимального объема ввода/вывода, разрешенного для платформы. Исключен в Oracle9*i*.

DB_FILE_MULTIBLOCK_READ_COUNT

Допустимые значения: от 1 до определяемого в зависимости от операционной системы

Значение по умолчанию: 8

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Указывает максимальное количество блоков, прочитываемых одной операцией ввода/вывода при последовательном опросе. В системах оперативной обработки транзакций значения этого параметра обычно находятся в диапазоне от 4 до 16. Системы поддержки принятия решений и хранилища данных зачастую используют максимальное значение параметра. Фактический максимум зависит от операционной системы, он никогда не превышает максимального объема ввода/вывода ОС, выраженного в блоках Oracle (максимальный объем ввода/вывода / DB_BLOCK_SIZE).

DB_FILE_SIMULTANEOUS_WRITES

Допустимые значения: 1 – 4 * количество дисков рассредоточенного файла или 4 при отсутствии рассредоточения

Значение по умолчанию: 4

Определяет максимальное количество одновременных операций записи в указанный файл базы данных. Сервер Oracle также основывается на значении данного параметра при вычислении различных внутренних параметров, которые относятся к операциям чтения и записи файлов БД. Кроме того, значение параметра применяется для определения количества прочтений файла при опережающем чтении журнала, когда журнал считывается при восстановлении. Исключен в Oracle8*i*.

DB_FILES

Допустимые значения: указанное в инструкции MAXDATAFILES последней команды CREATE DATABASE, или CREATE CONTROLFILE, или текущее фактическое количество файлов данных в БД

Значение по умолчанию: зависит от операционной системы

Указывает максимальное количество файлов базы данных, которые могут быть открыты для данной БД. Этот параметр должен устанавливаться равным максимальному количеству файлов (с учетом ограничений ОС), которые всегда будут указываться для базы данных, включая файлы, которые будут добавляться командой ADD DATAFILE. Если значение DB_FILES увеличивается, то новое значение вступает в силу только после остановки и повторного запуска всех экземпляров, обращающихся к данной БД.

DISK_ASYNCH_IO

Допустимые значения: TRUE | FALSE

Значение по умолчанию: TRUE

Определяет, является ли асинхронным ввод/вывод для файлов данных, управляющих и журнальных файлов. Если платформа не поддерживает асинхронный ввод/вывод на диск, то параметр не действует. Если значение параметра DISK_ASYNCH_IO равно FALSE, то для имитации асинхронного ввода/вывода необходимо задать ненулевое значение параметра DBWR_IO_SLAVES.

HASH_MULTIBLOCK_IO_COUNT

Допустимые значения: зависит от операционной системы

Значение по умолчанию: 1

Динамическое изменение: ALTER SESSION

Указывает, сколько последовательных блоков хеш-соединение читает и пишет в рамках одной операции ввода/вывода. При работе сервера в многопоточном режиме этот параметр игнорируется, устанавливается значение 1. Максимальное значение всегда меньше максимального объема ввода/вывода ОС, выраженного в блоках Oracle. Данный параметр заметно влияет на производительность, т. к. он управляет количеством разделов, на которые может быть разбит ввод. Исключен в Oracle9*i*.

LGWR_IO_SLAVES

Допустимые значения: от 0 до определяемого в зависимости от операционной системы

Значение по умолчанию: 0

Определяет количество подчиненных процессов ввода/вывода, используемых процессом LGWR. Исключен в Oracle8*i*.

Java

Следующие параметры относятся к применению Java в базе данных.

JAVA_MAX_SESSIONSPACE_SIZE

Допустимые значения: 0–4 G

Значение по умолчанию: 0

Определяет максимальный объем (в байтах) пространства сеанса, который будет доступен Java-программе, работающей на сервере. Появился в Oracle8*i*.

Если пользовательский сеанс пытается выделить больший объем памяти, чем указано в параметре, то виртуальная машина Java (Java Virtual Machine – JVM) генерирует исключение нехватки памяти и сеанс уничтожается.

JAVA_POOL_SIZE

Допустимые значения: 1 000 000–1 000 000 000
Значение по умолчанию: 20 000

Указывает размер пула Java (в байтах) в SGA. Появился в Oracle8*i*.

JAVA_SOFT_SESSIONSPACE_LIMIT

Допустимые значения: 0–4 G
Значение по умолчанию: 0

Указывает объем памяти (в байтах), который может быть занят Java в сеансе, прежде чем в файл трассировки будет выведено предупреждение. Появился в Oracle8*i*.

Задания

Параметры данного раздела применяются к заданиям и очередям заданий.

JOB_QUEUE_INTERVAL

Допустимые значения: 1–3600
Значение по умолчанию: 60

Задает интервал (в секундах) между запусками фоновых процессов SNP*n* экземпляра. Доступен только в Oracle8 и Oracle8*i*.

JOB_QUEUE_KEEP_CONNECTIONS

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Определяет, оставляют ли процессы очередей заданий сетевые соединения открытыми между заданиями. Исключен в Oracle8*i*.

JOB_QUEUE_PROCESSES

Допустимые значения: 0–1000
Значение по умолчанию: 0
Динамическое изменение: ALTER SYSTEM

Указывает максимальное количество фоновых процессов экземпляра, которые могут создаваться для выполнения заданий.

Лицензии

Лицензирование управляется следующими параметрами.

LICENSE_MAX_SESSIONS

Допустимые значения: от 0 до количества сеансовых лицензий
Значение по умолчанию: 0

Указывает максимальное количество разрешенных одновременно пользовательских сеансов. Если достигнуто предельное значение, то соединение с сервером разрешено только пользователям с привилегией RESTRICTED SESSION. Пользователи, которым не разрешено устанавливать соединение, получают предупреждение о том, что емкость системы полностью исчерпана.

Не следует одновременно применять лицензирование сеансов и лицензирование пользователей; необходимо всегда устанавливать в ноль, или LICENSE_MAX_SESSIONS, или же LICENSE_MAX_USERS. Если значение данного параметра отлично от нуля, следует также установить параметр LICENSE_SESSIONS_WARNING.

LICENSE_MAX_USERS

Допустимые значения: от 0 до количества пользовательских лицензий
Значение по умолчанию: 0

Задает максимальное количество пользователей, которое может быть создано в базе данных. Если достигнуто предельное значение, то создание дополнительных пользователей невозможно. Значения для всех экземпляров должны совпадать.

LICENSE_SESSIONS_WARNING

Допустимые значения: 0–LICENSE_MAX_SESSIONS
Значение по умолчанию: 0

Указывает порог предупреждения для количества параллельных пользовательских сеансов. Когда этот предел достигнут, подключение дополнительных пользователей все еще возможно, но для каждого нового соединения сервер Oracle записывает сообщение в сигнальный файл. Пользователи с привилегией RESTRICTED SESSION, которым разрешено соединение после достижения порогового значения, получают предупреждение о том, что система близка к исчерпанию доступного количества сеансов.

Блокировки и транзакции

В разделе приведены параметры, влияющие на то, как Oracle управляет поведением блокировок и транзакций.

DISCRETE_TRANSACTIONS_ENABLED

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Реализует простой, быстрый механизм отката, увеличивающий производительность для некоторых видов транзакций. Существуют ограничения на виды транзакций, которые могут происходить в дискретном режиме (подробная информация приведена в документации по Oracle). Исключен в Oracle8*i*.

DML_LOCKS

Допустимые значения: 0 | 20–∞
Значение по умолчанию: 4 * TRANSACTIONS

Указывает максимальное количество DML-блокировок; по одной для каждой таблицы, изменяемой в транзакции. Значение должно равняться общему количеству блокировок таблиц, на которые ссылаются в текущий момент все пользователи. Если значение параметра равно 0, то постановка в очередь запрещена и производительность немного увеличивается. Однако вам не удастся выполнить команды DROP TABLE, CREATE INDEX, а также явные команды блокировки, такие как LOCK TABLE IN EXCLUSIVE MODE. При работе нескольких экземпляров все значения должны быть либо положительными, либо нулевыми.

ENQUEUE_RESOURCES

Допустимые значения: 10–∞

Значение по умолчанию: Производная от SESSIONS

Определяет количество ресурсов, которые могут быть одновременно заблокированы диспетчером блокировок. Если сеансов меньше 4, то значение по умолчанию равно 20. Если сеансов от 4 до 10, то значение по умолчанию равно ((SESSIONS – 3) * 5) + 20. Для более чем 10 сеансов значение по умолчанию равно ((SESSIONS – 10) * 2) + 55. Если значение параметра ENQUEUE_RESOURCES задано и превышает сумму DML_LOCKS + 20, то берется предложенное значение. Значение может быть увеличено при наличии нескольких таблиц. Каждый ресурс независимо от количества сеансов и курсоров, использующих его, требует единицу из данного параметра. Увеличьте значение, если сервер Oracle возвращает ошибку, указывая, что возможности постановки в очередь исчерпаны.

ROW_LOCKING

Допустимые значения: ALWAYS | DEFAULT | INTENT

Значение по умолчанию: ALWAYS

Указывает, запрашиваются ли блокировки строк при обновлении таблицы. Если установить значения ALWAYS и DEFAULT, то при обновлении таблицы запрашиваются только блокировки строк. Если выбрать значение INTENT, то для команды SELECT FOR UPDATE устанавливаются только блокировки строк, а в момент обновления потребуется блокировка самих таблиц.

SPIN_COUNT

Допустимые значения: 1–1000000

Значение по умолчанию: 1

Определяет, сколько попыток получения защелки предпримет процесс. Если количество запросов превышает SPIN_COUNT, то процессу не удается получить защелку, он переходит в режим ожидания, а затем вновь пытается получить ее. Защелка представляет собой блокировку низкого уровня, поэтому процесс удерживает ее недолго и расходовать время процессора на циклический запрос менее накладно, чем вынуждать процесс переходить к ожиданию. Исключен в Oracle8*i*.

TRANSACTIONS

Допустимые значения: 4–232

Значение по умолчанию: 1,1 * SESSIONS

Задает максимальное количество одновременных транзакций.

TRANSACTIONS_PER_ROLLBACK_SEGMENT

Допустимые значения: 1–зависит от операционной системы

Значение по умолчанию: 5

Задает количество одновременных транзакций, разрешенных для одного сегмента отката. Минимальное количество сегментов отката, полученных при запуске, равно частному от деления значения параметра TRANSACTIONS на значение данного параметра.

Протоколирование и архивирование

Параметры этого раздела управляют протоколированием и архивированием. Родственные параметры описаны в разделе «Управление откатом (отменой/восстановлением)».

ARCH_IO_SLAVES

Допустимые значения: 0–15

Значение по умолчанию: 0

Задает количество подчиненных процессов ввода/вывода, которые процесс ARCH использует для архивации оперативных журнальных файлов. Процесс ARCH и подчиненные ему всегда ведут запись на диск. Обычно значение параметра корректируется, если оказывается, что процессов ввода/вывода недостаточно для обеспечения нормальной работы процесса ARCH. Исключен в Oracle8*i*.

ARCHIVE_LAG_TARGET

Допустимые значения: 0 или 60–7200

Значение по умолчанию: 0

Динамическое изменение: ALTER SYSTEM

Задает время (в секундах), по истечении которого происходит принудительный переход к новому журналу. Значение 0 указывает, что возможность перехода, контролируемого по времени, отключена. Появился в Oracle9*i*.

CPU_COUNT

Допустимые значения: 0–∞

Значение по умолчанию: 0 или фактическое количество процессоров

Указывает количество процессоров, доступных серверу Oracle. По этому параметру сервер Oracle устанавливает значение по умолчанию для параметра LOG_SIMULTANEOUS_COPIES. Для однопроцессорных компьютеров значение CPU_COUNT равно 0.

Для большинства платформ сервер Oracle автоматически устанавливает значение CPU_COUNT равным количеству процессоров, доступных экземпляру Oracle. При активной конкуренции за защелки измените значение LOG_SIMULTANEOUS_COPIES на удвоенное количество доступных процессоров, но не изменяйте значение CPU_COUNT.

DB_CREATE_ONLINE_LOG_DEST_*n*

Допустимые значения: строка, содержащая имя каталога
Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Указывает каталог по умолчанию, где будут создаваться оперативные журнальные файлы и управляющие файлы. Для обеспечения отказоустойчивости необходимо задать этот параметр хотя бы дважды. Выбранный каталог должен уже существовать, и сервер Oracle должен иметь разрешение на создание файлов в нем. Появился в Oracle9*i*.

Синтаксис

```
DB_CREATE_ONLINE_LOG_DEST_n = каталог
```

Ключевые слова

n	Целое число в диапазоне от 1 до 5, задающее экземпляр файла.
каталог	Задает имя каталога, который будет содержать один элемент каждой группы оперативных журналов и один управляющий файл.

FAL_CLIENT

Допустимые значения: допустимое имя клиента
Значение по умолчанию: нет
Динамическое изменение: ALTER SYSTEM

Указывает имя FAL-клиента (*fetch archive log client*), используемого FAL-сервисом. Появился в Oracle9*i*.

FAL_SERVER

Допустимые значения: допустимое имя сервера
Значение по умолчанию: нет
Динамическое изменение: ALTER SYSTEM

Определяет FAL-сервер (*fetch archive log server*) для резервной базы данных, значение должно быть именем FAL-сервера. Появился в Oracle9*i*.

LOG_ARCHIVE_BUFFER_SIZE

Допустимые значения: от 1 до определяемого в зависимости от операционной системы
Значение по умолчанию: зависит от операционной системы

Задает размер каждого буфера архивации (в блоках журнала). Значение по умолчанию должно быть достаточным для большинства приложений. Этот параметр совместно с LOG_ARCHIVE_BUFFERS может применяться для тонкой настройки архивации. Исключен в Oracle8*i*.

LOG_ARCHIVE_BUFFERS

Допустимые значения: зависят от операционной системы
Значение по умолчанию: зависит от операционной системы

Указывает количество буферов, выделяемых для архивирования. Совместно с LOG_ARCHIVE_BUFFER_SIZE может сделать архивирование настолько быстрым, насколько необходимо, но не настолько, чтобы снизить производительность системы. Исключен в Oracle8*i*.

LOG_ARCHIVE_DEST

Допустимые значения: корректный путь или имя устройства

Значение по умолчанию: зависит от операционной системы

Динамическое изменение: ALTER SYSTEM

Указывает местоположение по умолчанию каталога на диске или ленте, в который архивируются журнальные файлы. Имейте в виду, что архивирование на ленту поддерживается не всеми операционными системами.

Компания Oracle рекомендует при работе с Oracle8*i* Enterprise Edition или Oracle9*i* Enterprise Edition задавать параметр LOG_ARCHIVE_DEST_*n* вместо LOG_ARCHIVE_DEST.

LOG_ARCHIVE_DEST_*n*

Допустимые значения: см. описание

Значение по умолчанию: нет

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Задает до 10 местоположений архивных журналов. Параметр действует только для Oracle8*i* или Oracle9*i* Enterprise Edition. Параметр нельзя задавать одновременно с LOG_ARCHIVE_DEST.

Синтаксис

```
LOG_ARCHIVE_DEST_n = ((SERVICE = служба | LOCATION = местоположение)
[AFFIRM | NOAFFIRM]
[ALTERNATE = место_назначения | NOALTERNATE]
[ARCH | LGWR]
[DELAY[= минуты] | NODELAY]
[DEPENDENCY = место_назначения | NODEPENDENCY]
[MANDATORY | OPTIONAL]
[MAX_FAILURE = количество | NOMAX_FAILURE]
[QUOTA_SIZE = блоки | NOQUOTA_SIZE]
[QUOTA_USED = блоки | NOQUOTA_USED]
[REGISTER | NOREGISTER]
[REOPEN=секунды | NOREOPEN]
[SYNC | ASYNC=блоки])
```

Ключевые слова

n

Число от 1 до 10, являющееся идентификатором места назначения.

SERVICE=служба

Указывает имя сетевой службы, используемой для передачи архивного файла журнала резервному экземпляру.

LOCATION=местоположение

Определяет местоположение в локальной файловой системе. Должно быть указано хотя бы одно местоположение.

MANDATORY

Указывает, что архивирование в указанное место назначения должно завершиться успешно, прежде чем журнальный файл будет повторно использован.

OPTIONAL

Указывает, что для разрешения повторного использования журнала успешное архивирование в заданное место назначения не требуется. Это выбор по умолчанию.

REOPEN=секунды

Задает количество секунд, по прошествии которых архивирование после ошибки может выполняться в указанное место назначения. По умолчанию равно 300 секундам.

LOG_ARCHIVE_DEST_STATE_*n*

Допустимые значения: ENABLE | DEFER

Значение по умолчанию: ENABLE

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Указывает, доступно ли соответствующее мето назначения LOG_ARCHIVE_DEST_*n*. Параметр можно задавать только в сочетании с LOG_ARCHIVE_DEST_*n*. Компания Oracle рекомендует при работе с Oracle8*i* Enterprise Edition или Oracle9*i* Enterprise Edition задавать параметр LOG_ARCHIVE_DEST_*n* вместо LOG_ARCHIVE_DEST.

LOG_ARCHIVE_DUPLEX_DEST

Допустимые значения: корректный путь, или имя устройства, или NULL

Значение по умолчанию: NULL

Динамическое изменение: ALTER SYSTEM

Указывает второе место хранения архива. Успех архивации в данном месте может быть как обязательным, так и необязательным (требуются лишь максимальные усилия) в зависимости от того, в какое количество мест архивация должна завершиться успешно (см. описание LOG_ARCHIVE_MIN_SUCCEED_DEST далее в разделе).

Компания Oracle рекомендует при работе с Enterprise Edition вместо данного параметра задавать LOG_ARCHIVE_DEST_*n*. Если установлен параметр LOG_ARCHIVE_DUPLEX_DEST, не используйте LOG_ARCHIVE_DEST_*n*.

LOG_ARCHIVE_FORMAT

Допустимые значения: разрешенный формат имени файла

Значение по умолчанию: зависит от операционной системы

Задает устанавливаемый по умолчанию формат имени файла для архивирования журнальных файлов при работе в режиме ARCHIVELOG. Полученная строка добавляется в конец строки, указанной в параметре LOG_ARCHIVE_DEST.

Формат имени файла можно определить следующими переменными:

%s порядковый номер журнала

%t номер потока

Значение по умолчанию зависит от операционной системы, но обычно это *%t_%s.dbf*. Если имя переменной составлено буквами в верхнем регистре (например, %S), то значение будет иметь фиксированную длину и дополняться нулями слева.

LOG_ARCHIVE_MAX_PROCESSES

Допустимые значения: 1–10

Значение по умолчанию: 1

Динамическое изменение: ALTER SYSTEM

Указывает максимальное количество архивных процессов (носящих имена ARC0 – ARC9), которые изначально создает сервер Oracle. Появился в Oracle8*i*.

LOG_ARCHIVE_MIN_SUCCEED_DEST

Допустимые значения: 1–10

Значение по умолчанию: 1

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Задает минимальное количество мест назначения, в которые архивация должна быть успешной. Если включено автоматическое архивирование и задан параметр LOG_ARCHIVE_DEST, то допустимыми являются значения 1 и 2. Если значение параметра равно 1, то LOG_ARCHIVE_DEST – это местоположение обязательного успешного архивирования, а LOG_ARCHIVE_DUPLEX_DEST – направление приложения максимальных усилий для успешного архивирования. Если же параметр установлен в 2, то архивирование в оба места назначения, LOG_ARCHIVE_DEST и LOG_ARCHIVE_DUPLEX_DEST, должно пройти успешно.

Если установлен параметр LOG_ARCHIVE_DEST или LOG_ARCHIVE_DUPLEX_DEST, то его динамическое изменение невозможно.

LOG_ARCHIVE_START

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Определяет, будет ли архивирование автоматическим или же ручным при запуске экземпляра в режиме ARCHIVELOG. Команды Server Manager или SQL*Plus ARCHIVE LOG START и ARCHIVE LOG STOP аннулируют этот параметр.

Для включения режима ARCHIVELOG при создании базы данных установите значение данного параметра равным TRUE. Обычно база данных создается в режиме NOARCHIVELOG, а после создания переводится в режим ARCHIVELOG.

LOG_ARCHIVE_TRACE

Допустимые значения: 0–255

Значение по умолчанию: 0

Динамическое изменение: ALTER SYSTEM (изменяется при следующей операции архивирования)

Определяет уровень детализации протокола, записываемого фоновыми архивными процессами в сигнальный файл. Значения параметра:

0 отменить трассировку ARCHIVELOG; регистрировать только ошибки.
1 отслеживать архивацию журнального файла.
2 отслеживать состояние архивации для каждого места назначения ARCHIVELOG.
4 отслеживать этапы выполнения архивации.
8 отслеживать активность мест назначения ARCHIVELOG.
16 подробно отслеживать активность мест назначения ARCHIVELOG.
32 отслеживать изменения параметров мест назначения ARCHIVELOG.
64 отслеживать состояние процессов ARC*n*.
128 отслеживать деятельность, связанную с FAL-сервером.

Уровни можно комбинировать, складывая соответствующие значения. Например, если требуется выполнять трассировку архивирования журнального файла (1) и состояния каждого из мест назначений ARCHIVELOG (2), то надо задать значение 3. Появился в Oracle9*i*.

LOG_FILES

Допустимые значения: 2–255
Значение по умолчанию: 255

Все экземпляры должны иметь одинаковые значения.

Указывает максимальный номер журнальной группы и устанавливает верхний предел номеров групп, которые могут задаваться при выдаче связанных с журналированием команд. Исключен в Oracle8*i*.

REMOTE_ARCHIVE_ENABLE

Допустимые значения: TRUE | FALSE
Значение по умолчанию: TRUE

Указывает, разрешено ли архивирование журналов в удаленные места назначения. Появился в Oracle9*i*.

TAPE_ASYNCH_IO

Допустимые значения: TRUE | FALSE
Значение по умолчанию: TRUE

Указывает, является ли ввод/вывод на последовательные устройства (например, резервное копирование и восстановление данных Oracle на ленту или с ленты) асинхронным. Если платформа не поддерживает асинхронный ввод/вывод на последовательные устройства, то параметр не действует. Исключен в Oracle8*i*.

Распределение памяти

Следующие параметры управляют распределением и расходованием памяти.

BITMAP_MERGE_AREA_SIZE

Допустимые значения: зависят от операционной системы
Значение по умолчанию: 1 048576 (1 Мбайт)

Указывает объем памяти, выделяемый для слияния битовых массивов, которые извлекаются в результате диапазонного сканирования индекса. Большее значение должно увеличивать производительность, т. к. сегменты битовых массивов необходимо сортировать перед слиянием.

BUFFER_POOL_KEEP

Допустимые значения: целое | BUFFERS:целое | LRU_LATCHES:целое

Указывает количество буферов (размера DB_BLOCK_BUFFER), которые выделяются под буферный пул KEEP. Кроме того, дополнительно (необязательно) может быть указано количество LRU-защелок, выделяемых для буферного пула KEEP.

Начиная с Oracle9*i* комания Oracle рекомендует вместо BUFFER_POOL_KEEP задавать параметр DB_KEEP_CACHE_SIZE.

Не следует задавать параметр DB_KEEP_CACHE_SIZE, если установлен BUFFER_POOL_KEEP, в противном случае возникнет ошибка.

BUFFER_POOL_RECYCLE

Допустимые значения: целое | BUFFERS:целое | LRU_LATCHES:целое

Указывает количество буферов (размера DB_BLOCK_BUFFER), которые выделяются под буферный пул RECYCLE. Кроме того, дополнительно (не обязательно) может быть указано количество LRU-защелок, выделяемых для буферного пула RECYCLE.

Начиная с Oracle9*i* компания Oracle рекомендует вместо BUFFER_POOL_RECYCLE задавать параметр DB_RECYCLE_CACHE_SIZE.

Не следует указывать параметр DB_RECYCLE_CACHE_SIZE, если задан BUFFER_POOL_RECYCLE, в противном случае возникнет ошибка.

CACHE_SIZE_THRESHOLD

Допустимые значения: 0–DB_BLOCK_BUFFERS

Значение по умолчанию: 0,1 * DB_BLOCK_BUFFERS

Задает максимальный размер кэшируемой секции таблицы, распределенной между кэшами нескольких экземпляров. Если размер секции превышает значение параметра, таблица не разбивается по кэшам экземпляров. Значения для всех экземпляров должны совпадать. Параметр также может определять максимальный размер кэшируемой секции для одного экземпляра. Исключен в Oracle8*i*.

CREATE_BITMAP_AREA_SIZE

Допустимые значения: зависит от операционной системы

Значение по умолчанию: 8 388 608 (8 Мбайт)

Указывает объем памяти, выделяемый для создания битовых массивов. Значение по умолчанию равно 8 Мбайт, и чем значение больше, тем быстрее создается индекс. Если кардинальность индекса (количество уникальных значений) очень мала, можно установить небольшое значение для параметра.

DB_CACHE_ADVICE

Допустимые значения: ON | OFF | READY
Значение по умолчанию: OFF
Динамическое изменение: ALTER SYSTEM

Определяет, как будет выполняться сбор статистик, используемых для прогнозирования производительности СУБД с различным размером кэша. Если параметр установлен в OFF, а затем его значение меняется на ON при помощи команды ALTER SYSTEM, то возможно возникновение ошибки, т. к. память выделена не будет. Если вы хотите в дальнейшем иметь возможность изменить значение параметра на ON, задавайте значение READY. Появился в Oracle9*i*.

Ключевые слова

ON	прогнозирование включено, статистики собираются.
OFF	прогнозирование выключено, память для сбора статистик не выделяется.
READY	прогнозирование выключено, память для сбора статистик выделена.

DB_*n*K_CACHE_SIZE

Допустимые значения: 2 | 4 | 8 | 16 | 32
Значение по умолчанию: отсутствует
Динамическое изменение: ALTER SYSTEM

Указывает размер кэша для *n*K буферов, если значение параметра DB_BLOCK_SIZE отличается от *n*K. Не может быть меньше минимального и больше максимального размера блока для платформы. Появился в Oracle9*i*.

DB_BLOCK_BUFFERS

Допустимые значения для Oracle8/8i: от 4 до определяемого в зависимости от операционной системы
Значение по умолчанию: 50
Допустимые значения для Oracle9i: от 50 до определяемого в зависимости от операционной системы
Значение по умолчанию: 48 Мбайт / DB_BLOCK_SIZE

Определяет количество буферов базы данных, доступных в кэше буферов. Это один из основных параметров, задающих требования к объему памяти, занимаемому SGA экземпляра Oracle. Параметр DB_BLOCK_BUFFERS совместно с DB_BLOCK_SIZE определяет общий размер кэша буферов.

С появлением Oracle9*i* компания Oracle рекомендует вместо DB_BLOCK_BUFFERS задавать параметр DB_CACHE_SIZE.

DB_CACHE_SIZE

Допустимые значения: целое [Кбайт | Мбайт | Гбайт]
Значение по умолчанию: 48 Мбайт
Динамическое изменение: ALTER SYSTEM

Указывает размер буферного пула DEFAULT в буферах SGA с блоками исходного размера (заданного в DB_BLOCK_SIZE). Появился в Oracle9*i*.

DB_KEEP_CACHE_SIZE

Допустимые значения: целое число [Кбайт | Мбайт | Гбайт]
Значение по умолчанию: 0
Динамическое изменение: ALTER SYSTEM

Определяет количество буферов (размера DB_BLOCK_SIZE) в буферном пуле KEEP. Появился в Oracle9*i*.

DB_RECYCLE_CACHE_SIZE

Допустимые значения: целое число [Кбайт | Мбайт | Гбайт]
Значение по умолчанию: 0
Динамическое изменение: ALTER SYSTEM

Определяет количество буферов (размера DB_BLOCK_SIZE) в буферном пуле RECYCLE. Появился в Oracle9*i*.

HASH_AREA_SIZE

Допустимые значения: от 0 до определяемого в зависимости от операционной системы
Значение по умолчанию: 2 * SORT_AREA_SIZE
Динамическое изменение: ALTER SESSION

Указывает максимальный объем памяти (в байтах), выделяемой для хеш-соединений.

HI_SHARED_MEMORY_ADDRESS

Допустимые значения: целое число (адрес)
Значение по умолчанию: 0

Указывает исходный адрес SGA. Если задано значение 0, то адрес SGA по умолчанию определяется системой. На 64-разрядных платформах этот параметр определяет 32 старших разряда адреса, а младшие разряды задаются параметром SHARED_MEMORY_ADDRESS.

LARGE_POOL_MIN_ALLOC

Допустимые значения: 16 Кбайт–64 Мбайт
Значение по умолчанию: 16 Кбайт

Определяет минимальный размер области, выделяемой в большом пуле, в мегабайтах (Мбайт) или килобайтах (Кбайт). Исключен в Oracle8*i*.

LARGE_POOL_SIZE

Допустимые значения: 300 Кбайт–2 Гбайт или больше (максимум зависит от операционной системы)
Значение по умолчанию: 0

Указывает размер (в байтах) распределяемой кучи большого пула. Если параметр задан, то минимальный объем – это наибольшее из значений: 600 Кбайт (300 Кбайт в Oracle8) и LARGE_POOL_MIN_ALLOC. Значение параметра может указываться в байтах, мегабайтах (Мбайт) или килобайтах (Кбайт). По умолчанию параметр равен 0, т. е. большой пул не выделяется.

LOCK_SGA

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Определяет, блокируется ли вся SGA в реальной (физической) памяти. Должен быть установлен в TRUE, если существует возможность подкачки SGA на диск, т. к. это может значительно снизить производительность.

OBJECT_CACHE_MAX_SIZE_PERCENT

Допустимые значения: от 0 до определяемого в зависимости от операционной системы
Значение по умолчанию: 10
Динамическое изменение: ALTER SYSTEM DEFERRED, ALTER SESSION

Определяет количество процентов оптимального размера кэша, на которое может вырасти кэш объектов сеанса сверх оптимального размера. Максимальный размер равен сумме оптимального размера и произведения деленного на 100 значения данного параметра на значение оптимального размера. Если размер кэша превышает этот максимум, то система пытается его сократить.

OBJECT_CACHE_OPTIMAL_SIZE

Допустимые значения: от 102 400 до определяемого в зависимости от операционной системы
Значение по умолчанию: 102 400
Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Указывает размер, до которого сокращается кэш объектов сеанса, в случае если размер кэша превышает максимальный.

PGA_AGGREGATE_TARGET

Допустимые значения: 10 Мбайт–4096 Гбайт – 1
Значение по умолчанию: 0

Определяет суммарный объем памяти PGA, доступной всем серверным процессам, закрепленным за экземпляром. Должен быть установлен для того, чтобы включить автоматический выбор размера рабочих областей SQL, используемых активно работающими с памятью операторами SQL. Появился в Oracle9*i*.

PRE_PAGE_SGA

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Если задано значение TRUE, сервер Oracle при запуске экземпляра обращается ко всем страницам SGA, вызывая их подкачку в оперативную память. Другими словами, задание параметра увеличивает время запуска экземпляра и время входа пользователя в систему, но зато может сократить количество последующих проблем, связанных с отсутствием страниц. Уменьшение количества отсутствующих страниц позволяет экземпляру скорее достичь максимальной производительности. Параметр особенно полезен в системах, обладающих достаточным объемом памяти для хране-

ния всех страниц SGA без потерь производительности в других областях. Необходимо убедиться в том, что размер SGA установлен корректно по отношению к существующей физической памяти, чтобы избежать сброса страниц на диск.

SEQUENCE_CACHE_ENTRIES

Допустимые значения: 10–32 000
Значение по умолчанию: 10

Указывает количество последовательностей, которые могут быть кэшированы в SGA для немедленного доступа. Наивысшая степень параллелизма достигается, когда значение параметра равно максимально возможному количеству последовательностей, которые будут использоваться экземпляром одновременно. Последовательности, созданные с параметром NOCACHE, не хранятся в данном кэше и записываются в словарь данных при каждом использовании. Исключен в Oracle8*i*.

SGA_MAX_SIZE

Допустимые значения: от 1 до определяемого в зависимости от операционной системы
Значение по умолчанию: исходный размер SGA при запуске

Определяет максимальный размер (в байтах) SGA в течение жизни экземпляра. Появился в Oracle9*i*.

SHARED_MEMORY_ADDRESS

Допустимые значения: целое число (адрес)
Значение по умолчанию: 0

Указывает адрес начала SGA. Если параметр равен 0, то адрес SGA по умолчанию определяется системой. В 32-разрядных системах этот параметр задает весь адрес целиком. На 64-разрядных платформах этот параметр определяет 32 младших разряда адреса, а старшие разряды задаются параметром HI_SHARED_MEMORY_ADDRESS.

SHARED_POOL_RESERVED_MIN_ALLOC

Допустимые значения: 5 000–SHARED_POOL_RESERVED_SIZE
Значение по умолчанию: 5 000

Управляет распределением зарезервированной памяти. Если будет запрошено выделение блока памяти, размер которого превышает значение, указанное в данном параметре, и блок подходящего размера не будет найден в списке разделяемого пула, то память будет выделена из резервного списка. Если увеличить значение параметра, то сервер Oracle реже будет разрешать выделение памяти из резервного списка и чаще запрашивать память из списка разделяемого пула. Значение может быть числовым или представляться числом, за которым следует буква K (1000) или M (1 000 000). Исключен в Oracle8*i*.

SHARED_POOL_RESERVED_SIZE

Допустимые значения: SHARED_POOL_RESERVED_MIN_ALLOC–SHARED_POOL_SIZE / 2
Значение по умолчанию: SHARED_POOL_SIZE * 0,05

Задает размер разделяемого пула, зарезервированного для выделения больших непрерывных блоков памяти. Этот параметр совместно с SHARED_POOL_RESERVED_MIN_ALLOC может применяться для предотвращения потерь производительности разделяемого пула в тех ситуациях, когда фрагментация пула вынуждает сервер Oracle искать и освобождать неиспользованные порции пула для удовлетворения текущего запроса. Если значение SHARED_POOL_RESERVED_SIZE превышает 1/2 SHARED_POOL_SIZE, Oracle выдает ошибку. Обычно следует устанавливать значение параметра SHARED_POOL_RESERVED_SIZE равным 10% значения SHARED_ POOL_SIZE. Значение может быть числовым или представляться числом, за которым следует буква K (1000) или M (1 000 000).

SHARED_POOL_SIZE

Допустимые значения: от 300 K до определяемого в зависимости от операционной системы

Значение по умолчанию: 64 M для 64-разрядной системы, иначе – 8 M

Динамическое изменение: ALTER SYSTEM.

Указывает размер разделяемого пула (в байтах). Значение может быть числовым или представляться числом, за которым следует буква K (килобайты) или M (мегабайты).

SORT_AREA_RETAINED_SIZE

Допустимые значения: 2 * DB_BLOCK_SIZE–SORT_AREA_SIZE

Значение по умолчанию: SORT_AREA_SIZE

Динамическое изменение: ALTER SYSTEM DEFERRED, ALTER SESSION

Указывает максимальный размер (в байтах) пользовательской памяти, сохраняемой после завершения сортировки. Сохраняемый размер регулирует размер буфера чтения, предназначенного для хранения в памяти части сортировки. Можно выделить несколько пространств сортировки указанного размера.

SORT_AREA_SIZE

Допустимые значения: от 6 * DB_BLOCK_SIZE до определяемого в зависимости от операционной системы

Значение по умолчанию: зависит от операционной системы

Динамическое изменение: ALTER SYSTEM DEFERRED, ALTER SESSION

Указывает максимальный размер (в байтах) области PGA, отводимой для сортировки. Если включен режим Shared Server/MTS, то область сортировки выделяется из SGA. После завершения сортировки, когда остается лишь извлечь строки, память освобождается до размера, указанного в параметре SORT_AREA_RETAINED_SIZE. После выборки последней строки вся память освобождается. Память освобождается обратно в PGA, а не в операционную систему.

С появлением Oracle9*i* компания Oracle рекомендует задавать данный параметр только для экземпляра с включенным компонентом Shared Server. В остальных случаях вместо него рекомендуется параметр PGA_AGGREGATE_TARGET.

USE_INDIRECT_DATA_BUFFERS

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Определяет, как SGA расходует оперативную память. Если значение равно TRUE, то разрешено использование расширенного механизма кэша буферов для 32-разрядных платформ, которые могут поддерживать более 4 Гбайт физической памяти. Если значение равно FALSE, то такая возможность отключена. Появился в Oracle8*i*.

WORKAREA_SIZE_POLICY

Допустимые значения: AUTO | MANUAL

Значение по умолчанию: вычисляется

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Указывает политику задания размеров рабочих областей. Если значение равно AUTO, то размер рабочих областей, предназначенных для операций, активно работающих с памятью, устанавливается автоматически на основе PGA-памяти, которую использует система. Это значение может быть выбрано, только если определен параметр PGA_AGGREGATE_TARGET; в этом случае оно является и значением по умолчанию. Если значение параметра равно MANUAL, то размер рабочих областей задается вручную на основе значения параметра *_AREA_SIZE, соответствующего выполняемой операции (например, в случае сортировки устанавливается параметр SORT_AREA_SIZE). По умолчанию устанавливается значение MANUAL, если параметр PGA_AGGREGATE_TARGET не определен. Исключен в Oracle9*i*.

Имена

Следующие параметры управляют назначением имен различным файлам и объектам.

DB_CREATE_FILE_DEST

Допустимые значения: строка с именем каталога

Значение по умолчанию: не определено

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Указывает каталог по умолчанию, где будут создаваться файлы данных, управляющие файлы и оперативные журнальные файлы. Указанный каталог должен уже существовать и быть доступным серверу Oracle для создания файлов. Появился в Oracle9*i*.

DB_DOMAIN

Допустимые значения: любая разрешенная строка компонентов имени, разделенных точками, и длиной до 128 символов

Значение по умолчанию: WORLD

Указывает компоненты расширения глобального имени базы данных, состоящие из допустимых идентификаторов, разделенных точками. Доменное имя базы данных может включать в себя буквы, цифры, символ подчеркивания (_) и знак решетки (#).

DB_FILE_NAME_CONVERT

Допустимые значения: корректный образец имени файла данных

Преобразует имя нового файла данных основной базы данных в имя файла резервной базы данных. Файл резервной базы данных должен существовать и не быть защищенным от записи, иначе процесс восстановления остановится с ошибкой. Значением параметра являются две строки: первая – это образец имен файлов данных основной базы данных, а вторая – образец имен файлов данных резервной базы данных.

DB_NAME

Допустимые значения: любое корректное имя базы данных не длиннее восьми символов

Значение по умолчанию: NULL

Параметр можно не определять. Если он задается, то его значение должно совпадать с именем, указанным в команде CREATE DATABASE. Если параметр не задается, имя базы данных должно присутствовать в командной строке STARTUP или ALTER DATABASE MOUNT каждого экземпляра Parallel Server. Значения для всех экземпляров должны совпадать, или же команды STARTUP OPEN *имя_базы_данных* и ALTER DATABASE *имя_базы_данных* MOUNT должны содержать одно и то же имя.

ENT_DOMAIN_NAME

Допустимые значения: корректное доменное имя

Значение по умолчанию: нет

Указывает доменное имя предприятия. Применяется только в Oracle8*i*.

GLOBAL_NAMES

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Динамическое изменение: ALTER SESSION, ALTER SYSTEM

Указывает, должно ли название связи базы данных совпадать с именем базы данных, к которой эта связь подключена. Если значение равно FALSE, проверка не выполняется. Значение TRUE вводит в действие правила согласованного назначения имен баз данных и связей. В случае распределенной обработки установите GLOBAL_NAMES в TRUE, чтобы обеспечить уникальную идентификацию имени для каждой базы данных в сетевой среде.

SERVICE_NAMES

Допустимые значения: корректные имена служб

Значение по умолчанию: значение DB_NAMEDB_DOMAIN

Динамическое изменение: ALTER SYSTEM

Указывает одно или несколько имен службы базы данных, с которой связан данный экземпляр. Если в имени службы не указан домен, используется значение параметра DB_DOMAIN (если он задан), иначе используется домен локальной базы данных, определенный в словаре данных. Этот параметр должен быть установлен для каждого из экземпляров. Появился в Oracle8*i*.

Поддержка национальных языков (NLS)

Параметры раздела управляют использованием наборов символов для национальных языков.

NLS_CALENDAR

Допустимые значения: корректное название формата календаря

Значение по умолчанию: нет

Динамическое изменение: ALTER SESSION

Определяет календарную систему для базы данных. Можно выбрать одно из следующих значений:

ARABIC HIJRAH
ENGLISH HIJRAH
GREGORIAN
JAPANESE IMPERIAL
PERSIAN
ROC OFFICIAL
THAI BUDDHA

NLS_COMP

Допустимые значения: BINARY | ANSI

Значение по умолчанию: BINARY

Динамическое изменение: ALTER SESSION

Указывает, как будут выполняться сравнения в инструкции WHERE запросов. Если значение равно ANSI, то вместо обычного двоичного сравнения будет выполняться лингвистическая (языковая) сортировка, определяемая параметром NLS_SORT. Для столбца, по которому планируется осуществлять лингвистическую сортировку, необходимо наличие индекса. Появился в Oracle8*i*.

NLS_CURRENCY

Допустимые значения: любая допустимая символьная строка размером до 10 байт

Значение по умолчанию: вычисляется

Динамическое изменение: ALTER SESSION

Указывает строку, выступающую в качестве обозначения местной валюты для элементов числового формата L. Значение по умолчанию определяется параметром NLS_TERRITORY.

NLS_DATE_FORMAT

Допустимые значения: любая разрешенная маска формата даты

Значение по умолчанию: вычисляется

Динамическое изменение: ALTER SESSION

Указывает формат даты по умолчанию для функций TO_CHAR и TO_DATE. Значение по умолчанию определяется параметром NLS_TERRITORY. Значением может быть любая разрешенная маска формата даты, заключенная в двойные кавычки.

NLS_DATE_LANGUAGE

Допустимые значения: любое действительное значение NLS_LANGUAGE

Значение по умолчанию: значение NLS_LANGUAGE

Динамическое изменение: ALTER SESSION

Задает язык, на котором будут записываться словами названия дней и месяцев и аббревиатур в датах (AD, BC, AM, PM).

NLS_DUAL_CURRENCY

Допустимые значения: любое разрешенное название формата

Значение по умолчанию: вычисляется

Динамическое изменение: ALTER SESSION

Задает обозначение второй валюты (например, для евро). Значение по умолчанию – это обозначение второй валюты, определенной для указанной территории. Появился в Oracle8*i*.

NLS_ISO_CURRENCY

Допустимые значения: любое корректное значение NLS_TERRITORY

Значение по умолчанию: вычисляется

Динамическое изменение: ALTER SESSION

Указывает строку, определенную в качестве международного обозначения валюты для элементов числового формата C. Значение по умолчанию задается параметром NLS_TERRITORY.

NLS_LANGUAGE

Допустимые значения: любое корректное название языка

Значение по умолчанию: вычисляется

Динамическое изменение: ALTER SESSION

Указывает заданный по умолчанию язык базы данных. Применяется для сообщений, названий дней и месяцев, обозначений в дате, таких как AD, BC, AM и PM; а также для выбора механизма сортировки по умолчанию. Данный параметр определяет значения по умолчанию для параметров NLS_DATE_LANGUAGE и NLS_SORT. Полный список языков приведен в документах «Oracle9*i* Globalization Support Guide» (Поддержка глобализации в Oracle 9*i*), «Oracle Server Reference Manual» (Справочное руководство по Oracle Server) и замечаниях по соответствующему выпуску Oracle Server.

NLS_LENGTH_SEMANTICS

Допустимые значения: BYTE | CHAR

Значение по умолчанию: BYTE

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Определяет, в какой семантике создаются столбцы CHAR и VARCHAR2 – в байтовой или же символьной. В таблицах схем SYS и SYSTEM всегда соблюдается байтовая семантика. Столбцы NCHAR, NVARCHAR, CLOB и NCLOB всегда создаются на символьной основе. Появился в Oracle9*i*.

NLS_NUMERIC_CHARACTERS

Допустимые значения: см. описание

Значение по умолчанию: вычисляется

Динамическое изменение: ALTER SESSION

Указывает символы, выступающие в качестве разделителя десятичной дроби и разделителя групп. Установка этого параметра подменяет значения, неявно определяемые NLS_TERRITORY.

Два устанавливаемых символа должны быть однобайтными и не могут совпадать. Они не могут быть цифровыми, а также следующими символами: плюсом (+), дефисом (–), знаком «меньше» (<), знаком «больше» (>). Символы задаются в следующем формате:

```
NLS_NUMERIC_CHARACTERS = "<разделитель десятичной дроби><разделитель групп>"
```

NLS_SORT

Допустимые значения: BINARY или корректное имя языка

Значение по умолчанию: вычисляется

Динамическое изменение: ALTER SESSION

Задает схему упорядочения для запросов ORDER BY. Если выбрано значение BINARY, то в основе схемы упорядочения запросов ORDER BY лежит числовое значение кода символов (двоичная сортировка, требующая меньших накладных расходов).

Если значение – это имя языка, то упорядочивание выполняется на основе определенной лингвистической сортировки. Значение параметра по умолчанию зависит от значения параметра NLS_LANGUAGE. Если необходимо лингвистическое упорядочение, то в операциях сравнения должен применяться оператор NLS_SORT.

Если присвоить параметру NLS_SORT значение, отличное от BINARY, то при сортировке таблица сканируется полностью вне зависимости от маршрута, выбранного оптимизатором.

NLS_TERRITORY

Допустимые значения: любое

Значение по умолчанию: зависит от операционной системы

Динамическое изменение: ALTER SESSION

Указывает название территории, правила которой определяют нумерацию дней и недель, установленные по умолчанию формат даты, разделитель десятичной дроби и разделитель групп, местное обозначение валюты и код валюты по ISO. Полный список территорий приведен в документах «Oracle9*i* Globalization Support Guide» (Поддержка глобализации в Oracle 9*i*) и «Oracle Server Reference Manual» (Справочное руководство по Oracle Server).

NLS_TIMESTAMP_FORMAT

Допустимые значения: любая разрешенная маска формата даты и времени

Значение по умолчанию: вычисляется

Динамическое изменение: ALTER SESSION

Определяет установленный по умолчанию формат для *временных меток* (*timestamps*), который будет применяться в функциях TO_CHAR и TO_TIMESTAMP. Зна-

чение параметра должно быть заключено в одинарные кавычки. Значение по умолчанию определяется параметром NLS_TERRITORY. Появился в Oracle9*i*.

NLS_TIMESTAMP_TZ_FORMAT

Допустимые значения: любая разрешенная маска формата даты/времени
Значение по умолчанию: вычисляется
Динамическое изменение: ALTER SESSION

Определяет заданный по умолчанию формат для временных меток с указанием *часового пояса* (*time zone*), который будет применяться в функциях TO_CHAR и TO_TIMESTAMP_TZ. Значение параметра определяется параметром NLS_TERRITORY и должно быть заключено в одинарные кавычки. Появился в Oracle9*i*.

Оптимизация и производительность

Параметры, описанные в данном разделе, управляют работой оптимизатора и различными аспектами производительности сервера Oracle.

B_TREE_BITMAP_PLANS

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Указывает, что оптимизатор рассматривает путь доступа к битовому массиву, даже если таблица имеет только обычные индексы (B-деревья). Значение параметра следует изменять только в соответствии с инструкциями службы технической поддержки компании Oracle. Исключен в Oracle8*i*.

BLANK_TRIMMING

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Значение TRUE разрешает присваивание значения исходной символьной строки/переменной символьному столбцу/переменной, даже если длина источника превышает длину получателя, в случае когда дополнительная длина возникает исключительно за счет пробелов. Значение FALSE запрещает подобное присваивание.

FAST_FULL_SCAN_ENABLED

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Определяет, разрешено ли быстрое полное сканирование пути доступа запроса. Исключен в Oracle8*i*.

HASH_JOIN_ENABLED

Допустимые значения: TRUE | FALSE
Значение по умолчанию: TRUE

Динамическое изменение: ALTER SESSION

Указывает, будет ли оптимизатор рассматривать применение хеш-соединений.

OPTIMIZER_FEATURES_ENABLE

Допустимые значения: 8.0.0 | 8.0.3 | 8.0.4 | 8.0.5 | 8.0.6 | 8.1.0 | 8.1.3 | 8.1.4 | 8.1.5 | 8.1.6 | 8.1.7 | 9.0.0 |9.0.1 | 9.2.0
Значение по умолчанию: текущая версия

Указывает, что поведение оптимизатора Oracle должно учитывать особенности конкретной версии продукта.

OPTIMIZER_INDEX_CACHING

Допустимые значения: 0–100
Значение по умолчанию: 0

Указывает предполагаемый процент кэшированных индексных блоков. При больших значениях предпочтение оказывается планам выполнения на основе индексов.

OPTIMIZER_INDEX_COST_ADJ

Допустимые значения: 1–10 000
Значение по умолчанию: 100
Динамическое изменение: ALTER SESSION

Задает «скидку» для стоимости планов выполнения на основе индексов.

OPTIMIZER_MAX_PERMUTATIONS

Допустимые значения: 4–232 (Oracle8i); 4–80 000 (Oracle9i)
Значение по умолчанию: 80 000
Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Указывает максимальное количество перестановок таблиц, которое оптимизатор будет рассматривать для запросов с соединениями. Если значение слишком мало, то время синтаксического анализа запроса будет уменьшено, что увеличит риск пропуска более удачного плана исполнения. По умолчанию равен 80 000, что означает отсутствие ограничений.

OPTIMIZER_MODE

Допустимые значения: RULE | CHOOSE | FIRST_ROWS_{1 | 10 | 100 | 1000} | FIRST_ROWS | ALL_ROWS
Значение по умолчанию: CHOOSE
Динамическое изменение: ALTER SESSION

Регулирует поведение оптимизатора. Описания возможных значений параметра приведены в главе 7. Оптимизация по стоимости всегда будет применяться для любых запросов, ссылающихся на объект с ненулевой степенью параллелизма.

OPTIMIZER_PERCENT_PARALLEL

Допустимые значения: 0–100
Значение по умолчанию: 0

Динамическое изменение: ALTER SESSION

Указывает, в каком объеме оптимизатор применяет параллелизм для своих функций стоимости. Если оставлено значение по умолчанию, то оптимизатор выбирает наилучший последовательный план. Если значение параметра равно 100, то оптимизатор при вычислении стоимости операции полного просмотра таблицы учитывает степень параллелизма каждого объекта. Исключен в Oracle9*i*.

OPTIMIZER_SEARCH_LIMIT

Допустимые значения: 0–10
Значение по умолчанию: 5
Динамическое изменение: ALTER SESSION

Задает количество таблиц, для которых в операциях соединения будет рассматриваться декартово произведение. Исключен в Oracle8*i*.

PARTITION_VIEW_ENABLED

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Определяет поведение оптимизатора по отношению к секционированным представлениям. Если значение параметра равно TRUE, то оптимизатор отсекает или пропускает ветви доступа к ненужным таблицам для секционированного представления. Кроме того, изменяется способ, посредством которого оптимизатор по стоимости вычисляет статистики секционированного представления из статистик базовых таблиц.

QUERY_REWRITE_ENABLED

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE
Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Указывает, будет ли разрешена для базы данных перезапись запросов таким образом, чтобы использовались материализованные представления. Для того чтобы сделать доступными преимущества такой перезаписи запроса, следует включить этот параметр для каждого материализованного представления; кроме того, должна применяться оптимизация по стоимости. Появился в Oracle9*i*.

QUERY_REWRITE_INTEGRITY

Допустимые значения: STALE_TOLERATED | TRUSTED | ENFORCED
Значение по умолчанию: ENFORCED
Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Определяет степень перезаписи запросов сервером Oracle.

Ключевые слова

STALE_TOLERATED

Разрешает перезапись не введенных в принудительном порядке отношений. Материализованные представления могут быть перезаписаны, даже если известно, что они не согласованы с лежащими в их основе детальными данными.

TRUSTED

Разрешает перезапись с использованием отношений, которые были объявлены, но не введены принудительно сервером Oracle.

ENFORCED

Oracle принудительно вводит и гарантирует согласованность и целостность.

READ_ONLY_OPEN_DELAYED

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Указывает, когда производится доступ к таблицам в табличных пространствах, открытых только для чтения. Значение TRUE означает, что файлы данных открываются только при попытке их прочтения. Если же параметр установлен в FALSE, то файлы открываются в момент открытия базы данных.

SEQUENCE_CACHE_HASH_BUCKET

Допустимые значения: 10–SEQUENCE_CACHE_ENTRIES
Значение по умолчанию: 7

Указывает количество столбцов, используемых для ускорения поиска вновь запрошенных последовательностей в кэше, организованном как хеш-таблица. Следует указать в качестве значения параметра простое число, в противном случае сервер Oracle увеличит значение до ближайшего простого числа. Исключен в Oracle8*i*.

STAR_TRANSFORMATION_ENABLED

Допустимые значения: TRUE | FALSE | TEMP_DISABLE
Значение по умолчанию: FALSE
Динамическое изменение: ALTER SESSION

Определяет, будет ли преобразование запроса по стоимости применяться к запросам типа «звезда».

TIMED_OS_STATISTICS

Допустимые значения: 0–∞
Значение по умолчанию: 0
Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Определяет интервал времени (в секундах), в течение которого сервер Oracle собирает статистики работы операционной системы. Если параметр равен 0, то сбор статистик не ведется. Исключен в Oracle8*i*.

TIMED_STATISTICS

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE
Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Определяет, ведется ли сбор статистик, связанных со временем. Если выбрано значение FALSE, статистики устанавливаются в 0 и сервер избегает накладных расходов на запрашивание времени от операционной системы.

Параллельное выполнение

Параметры данного раздела управляют параллельным выполнением.

PARALLEL_ADAPTIVE_MULTI_USER

Допустимые значения: TRUE | FALSE
Значение по умолчанию: вычисляется
Динамическое изменение: ALTER SYSTEM

Указывает, будет ли сервер Oracle динамически регулировать степень параллелизма в зависимости от загрузки системы на момент начала выполнения запроса. Значение по умолчанию берется из параметра PARALLEL_AUTOMATIC_TUNING. Применяемый алгоритм предполагает, что система настроена на оптимальную производительность для однопользовательской работы.

PARALLEL_AUTOMATIC_TUNING

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Определяет, будет ли сервер Oracle автоматически выбирать значения по умолчанию для большинства параметров настройки параллельного выполнения. Кроме того, необходимо задать инструкцию PARALLEL для целевых таблиц системы. Появился в Oracle8*i*.

PARALLEL_BROADCAST_ENABLED

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE
Динамическое изменение: ALTER SESSION

Указывает, может ли сервер Oracle при необходимости скопировать все исходные строки небольшого результирующего множества и разослать копии каждой базе данных кластера, которая обрабатывает строки большего множества.

PARALLEL_DEFAULT_MAX_INSTANCES

Допустимые значения: от 0 до количества экземпляров
Значение по умолчанию: зависит от операционной системы

Указывает задаваемое по умолчанию количество экземпляров для разбиения таблицы с целью обработки параллельных запросов. Определяется, если в инструкции PARALLEL определения таблицы указано INSTANCES DEFAULT. Исключен в Oracle8*i*.

PARALLEL_EXECUTION_MESSAGE_SIZE

Допустимые значения: 2148–65535

Значение по умолчанию: зависит от операционной системы

Указывает в байтах размер сообщений для параллельного выполнения. В большинстве систем значение по умолчанию равно 2148, если значение параметра PARALLEL_AUTOMATIC_TUNING равно FALSE, и 4096, если значение этого параметра равно TRUE. Значения для всех экземпляров должны совпадать. Большие значения приводят к повышению производительности, но за счет использования большего объема памяти.

PARALLEL_INSTANCE_GROUP

Допустимые значения: корректное имя группы

Значение по умолчанию: группа всех активных на текущий момент экземпляров

Указывает группу параллельных экземпляров, которая будет использоваться для порождения подчиненных параллельных запросов. Параллельные операции будут порождать параллельные подчиненные запросы только для экземпляров, у которых в параметре INSTANCE_GROUPS указана соответствующая группа.

PARALLEL_MAX_SERVERS

Допустимые значения: 0–3599

Значение по умолчанию: вычисляется

Определяет максимальное количество серверов параллельных запросов или процессов параллельного восстановления для экземпляра. Сервер Oracle увеличит количество серверов запроса, созданных при запуске экземпляра, до указанного. Если значение будет слишком мало, то может оказаться, что для обработки каких-то запросов не будет доступных серверов запросов. Если же значение слишком большое, то в пиковые периоды может наблюдаться нехватка ресурсов памяти. Значения для всех экземпляров должны совпадать.

PARALLEL_MIN_MESSAGE_POOL

Допустимые значения: 0–SHARED_POOLSIZE * 0,9

Значение по умолчанию: (кол-во процессоров) * PARALLEL_MAX_SERVERS * 1,5 * (размер буфера сообщений ОС) или (кол-во процессоров) * 5 * 1,5 * (размер сообщений ОС)

Указывает минимальный объем памяти, постоянно выделенной из разделяемого пула для сообщений при параллельном выполнении. Выделяется в момент запуска, если значение параметра PARALLEL_MIN_SERVERS не равно нулю, или при первом распределении сервера. Данному параметру не может быть присвоено значение, превышающее 90% разделяемого пула. Исключен в Oracle8*i*.

PARALLEL_MIN_PERCENT

Допустимые значения: 0–100

Значение по умолчанию: 0

Динамическое изменение: ALTER SESSION

Определяет минимальный процент потоков, необходимых для параллельного выполнения. Гарантирует, что параллельная операция не будет выполняться последовательно в случае отсутствия достаточных ресурсов. Если запросу доступно слишком мало подчиненных, выводится сообщение об ошибке, запрос не выполняется.

PARALLEL_MIN_SERVERS

Допустимые значения: 0–PARALLEL_MAX_SERVERS
Значение по умолчанию: 0

Указывает минимальное количество серверных процессов параллельного выполнения для экземпляра.

PARALLEL_THREADS_PER_CPU

Допустимые значения: 1–∞
Значение по умолчанию: обычно 2
Динамическое изменение: ALTER SYSTEM

Указывает степень параллелизма по умолчанию для экземпляра и представляет собой количество процессов параллельного выполнения, которые процессор может обработать в режиме параллельного выполнения или в многоэкземплярной среде. Как правило, следует уменьшать значение, если машина кажется перегруженной при выполнении стандартной параллельной операции, и увеличивать его, если перегружена подсистема ввода/вывода.

PARALLEL_TRANSACTION_RESOURCE_TIMEOUT

Допустимые значения: от 0 до определяемого в зависимости от операционной системы
Значение по умолчанию: 300
Динамическое изменение: ALTER SYSTEM

Указывает максимальный интервал времени, после которого для сеанса, выполняющего параллельную операцию, истекает время ожидания ресурса, занятого другим сеансом в режиме несовместимых блокировок. Исключен в Oracle8*i*.

Параметры

Параметры, приведенные в данном разделе, указывают на другие файлы параметров.

IFILE

Допустимые значения: корректное имя файла
Значение по умолчанию: нет

Включает в текущий файл параметров другой файл параметров. Допускается до трех уровней вложения. Можно включить в один файл несколько дополнительных, указав IFILE несколько раз с разными значениями.

SPFILE

Допустимые значения: любое корректное имя файла параметров сервера (*SPFILE*)
Значение по умолчанию: *$ORACLE_HOME/dbs/spfile.ora*

Указывает имя текущего файла параметров сервера. Для всех экземпляров должны быть установлены одинаковые значения.

PL/SQL

Параметры раздела относятся к языку PL/SQL

PLSQL_COMPILER_FLAGS

Допустимые значения: [DEBUG | NON_DEBUG] [INTERPRETED | NATIVE]
Значение по умолчанию: NON_DEBUG INTERPRETED
Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Указывает флаги компилятора PL/SQL. Параметр не действует на скомпилированные модули PL/SQL. Появился в Oracle9*i*.

PLSQL_LOAD_WITHOUT_COMPILE

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE
Динамическое изменение: ALTER SESSION

Указывает, можно ли загружать PL/SQL без компиляции.

PLSQL_NATIVE_C_COMPILER

Допустимые значения: любой корректный путь к файлу
Значение по умолчанию: содержится в make-файле, поставляемом для каждой конкретной платформы
Динамическое изменение: ALTER SYSTEM

Указывает полный путь к компилятору C, используемому для компиляции сгенерированного кода на C в объектный файл. Появился в Oracle9*i*.

PLSQL_NATIVE_LIBRARY_DIR

Допустимые значения: любой корректный путь к каталогу
Значение по умолчанию: нет
Динамическое изменение: ALTER SYSTEM

Указывает полный путь к каталогу, где сохраняются совместно используемые объекты, созданные собственным компилятором. Появился в Oracle9*i*.

PLSQL_NATIVE_LIBRARY_SUBDIR_COUNT

Допустимые значения: от 0 до 232–1
Значение по умолчанию: 0
Динамическое изменение: ALTER SYSTEM

Определяет количество подкаталогов, создаваемых администратором БД в каталоге, указанном в параметре PLSQL_NATIVE_LIBRARY_DIR. Разрешается создание нескольких подкаталогов, что позволяет избежать неблагоприятно влияющего на производительность создания большого количества файлов в одном каталоге. Появился в Oracle9*i*.

PLSQL_NATIVE_LINKER

Допустимые значения: любой корректный путь к файлу

Значение по умолчанию: содержится в make-файле, поставляемом для каждой конкретной платформы

Динамическое изменение: ALTER SYSTEM

Указывает полное имя компоновщика, применяемого для включения объектного файла в общий объектный файл или DLL. Появился в Oracle9*i*.

PLSQL_NATIVE_MAKE_UTILITY

Допустимое значение: любой корректный путь к файлу

Значение по умолчанию: нет

Динамическое изменение: ALTER SYSTEM

Указывает полное имя утилиты make, применяемой для формирования общего объектного файла или DLL из сгенерированного кода на C. Появился в Oracle9*i*.

PLSQL_V2_COMPATIBILITY

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Регулирует совместимость версий для PL/SQL. Если значение равно FALSE, то соблюдается поведение, присущее версии PL/SQL 8, а особенности версии 2 не поддерживаются. Если же значение параметра равно TRUE, то при работе с PL/SQL версии 8 поддерживаются следующие возможности, которые были доступны только в версии 2:

- Разрешается изменение и удаление элементов параметра IN типа *index-by table*.[1]
- В некоторых случаях разрешается использование выходных параметров в контексте выражений. Это относится всего к нескольким ситуациям: к полям выходных параметров, представляющих собой записи, и к выходным параметрам, присутствующим в списке FROM команды SELECT.
- Разрешается включение выходных параметров в инструкции FROM списка SELECT, где считываются их значения.
- Разрешается передача входного параметра в другую процедуру в качестве выходного (только для полей входных параметров, являющихся записями).
- Разрешается применять опережающую ссылку на тип до его определения.

UTL_FILE_DIR

Допустимые значения: корректное имя каталога

Значение по умолчанию: нет

Указывает каталог, предназначенный для ввода/вывода файлов PL/SQL. Можно задать несколько каталогов, но каждый из них должен быть указан в отдельном параметре UTL_FILE_DIR.

---

[1] Начиная с версии Oracle 9 наряду с термином «индекс-таблица» употребляется также «ассоциативный массив». – *Примеч. перев.*

Удаленные узлы

Параметры данного раздела регулируют взаимодействие с удаленными узлами.

REMOTE_DEPENDENCIES_MODE

Допустимые значения: TIMESTAMP | SIGNATURE
Значение по умолчанию: TIMESTAMP

Указывает, как зависимости от удаленных хранимых процедур должны обрабатываться сервером БД. Если значение параметра равно TIMESTAMP, то клиент, запускающий процедуру, сравнивает временную метку, записанную в серверной процедуре, с текущей временной меткой локальной процедуры и выполняет процедуру, только если такие метки совпадают. Если значение параметра равно SIGNATURE, то выполнение процедуры разрешается до тех пор, пока сигнатуры считаются надежными, что позволяет запускать клиентские приложения PL/SQL без перекомпиляции.

REMOTE_LISTENER

Допустимые значения: корректное сетевое имя
Значение по умолчанию: нет

Указывает сетевое имя для одного адреса или списка адресов удаленных сетевых процессов прослушивания Oracle, которые приведены в файле *tnsnames.ora*. Появился в Oracle9*i*.

REMOTE_OS_AUTHENT

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Если значение равно TRUE, включается аутентификация удаленных клиентов на основе значения параметра инициализации OS_AUTHENT_PREFIX (см. далее раздел «Безопасность»).

REMOTE_OS_ROLES

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Если значение равно TRUE, включается механизм ролей операционной системы для удаленных клиентов.

Управление откатом (отменой/восстановлением)

Параметры раздела отвечают за использование буферов отката, табличных пространств отката, журналов и управления ими.

CLEANUP_ROLLBACK_ENTRIES

Допустимые значения: целое число
Значение по умолчанию: 20

Указывает количество записей отмены, обрабатываемых за один раз при откате транзакции. Предотвращает «замораживание» длинными транзакциями коротких транзакций, которые также ждут отката. Исключен в Oracle8*i*.

LOG_BLOCK_CHECKSUM

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Указывает, будут ли блоки журнала содержать контрольную сумму. Исключен в Oracle9*i*.

LOG_BUFFER

Допустимые значения: зависит от операционной системы

Значение по умолчанию: Максимальное из значений 512 K и 128 K * CPU_COUNT

Определяет объем памяти (в байтах), выделяемый для буфера журнала. В общем случае большие значения сокращают ввод/вывод для файлов журнала, особенно если транзакции длинны или многочисленны. Если в системе доступно несколько процессоров, то значение параметра может достигать 128K * CPU_COUNT.

LOG_CHECKPOINT_INTERVAL

Допустимые значения: 0–∞

Значение по умолчанию: зависит от операционной системы

Динамическое изменение: ALTER SYSTEM.

Задает частоту установки контрольных точек в терминах количества блоков ОС журнального файла, которые записываются между последовательными точками. Если значение превышает действительный размер файла журнала, то контрольные точки будут устанавливаться только при переключении журнала. Если интервалы настолько близки, что запросы на выполнение контрольных точек поступают быстрее, чем сервер может их выполнить, то Oracle может проигнорировать некоторые запросы во избежание чрезмерного введения контрольных точек.

LOG_CHECKPOINT_TIMEOUT

Допустимые значения: 0–∞

Значение по умолчанию: 0

Динамическое изменение: ALTER SYSTEM

Задает максимальный интервал времени (в секундах) между контрольными точками. Отсчет времени начинается от начала предыдущей контрольной точки.

LOG_CHECKPOINTS_TO_ALERT

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Динамическое изменение: ALTER SYSTEM

Указывает, что контрольные точки должны записываться в сигнальный файл. Позволяет определить, с нужной ли частотой устанавливаются контрольные точки.

LOG_SIMULTANEOUS_COPIES

Допустимые значения: 0–∞
Значение по умолчанию: CPU_COUNT

Указывает максимальное количество защелок копирования журнального буфера, позволяющих одновременно записывать журнальные записи. Для повышения производительности количество защелок копирования необходимо установить равным двукратному количеству процессоров. Исключен в Oracle8*i*.

LOG_SMALL_ENTRY_MAX_SIZE

Допустимые значения: зависит от операционной системы
Значение по умолчанию: зависит от операционной системы
Динамическое изменение: ALTER SYSTEM

Указывает размер (в байтах) наибольшей записи, которая может копироваться в журнальный буфер под защелкой распределения без получения защелки копирования. Исключен в Oracle8*i*.

MAX_ROLLBACK_SEGMENTS

Допустимые значения: 2–65535
Значение по умолчанию: MAX (30, TRANSACTION/TRANSACTIONS_PER_ROLLBACK_SEGMENT)

Определяет максимальный размер кэша сегмента отката в SGA, а тем самым и максимальное количество сегментов отката, которые может одновременно поддерживать в оперативном состоянии один экземпляр.

ROLLBACK_SEGMENTS

Допустимые значения: список имен сегментов отката (за исключением SYSTEM), разделенных запятыми
Значение по умолчанию: нет (по умолчанию экземпляр использует открытые сегменты отката)

Указывает (по имени) один или несколько сегментов отката, выделяемых для данного экземпляра. Если параметр установлен, экземпляр получает все сегменты отката, названные в параметре, даже если их количество превышает минимальное необходимое для экземпляра (вычисляемое как TRANSACTIONS / TRANSACTIONS_PER_ROLLBACK_SEGMENT). Также можно указать открытые сегменты отката, если они еще не используются. Если же параметр не установлен, экземпляр по умолчанию использует все открытые сегменты отката. Различные экземпляры должны иметь собственные сегменты отката.

UNDO_MANAGEMENT

Допустимые значения: MANUAL | AUTO
Значение по умолчанию: MANUAL

Указывает действующий режим распределения пространства отката. Значение MANUAL означает, что пространство отката выделяется в виде сегментов отката. Значение AUTO указывает, что применяется автоматическое управление откатом. Появился в Oracle9*i*.

UNDO_RETENTION

Допустимые значения: от 0 до 232–1
Значение по умолчанию: 900
Динамическое изменение: ALTER SYSTEM

Определяет, как долго (в секундах) зафиксированная информация об откате должна храниться в базе данных. Значения для всех экземпляров должны совпадать. Появился в Oracle9*i*.

UNDO_SUPPRESS_ERRORS

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE
Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Определяет, подавляются ли ошибки при выполнении ручных операций управления откатом в автоматическом режиме. Применяется для сценариев версий до Oracle9*i* с командами управления откатом версии Oracle9*i*.

UNDO_TABLESPACE

Допустимые значения: имя существующего табличного пространства отката
Значение по умолчанию: первое доступное табличное пространство отката
Динамическое изменение: ALTER SYSTEM

Указывает табличное пространство отката, которое будет использовано при запуске экземпляра. Если табличные пространства отката не указаны или недоступны, сервер Oracle использует сегмент отката SYSTEM (такой режим работы не рекомендуется). Если значение параметра указывается для базы данных, работающей в режиме ручного управления, то экземпляр не будет запущен. Появился в Oracle9*i*.

Безопасность

Соблюдение мер безопасности регулируется следующими параметрами.

MAX_ENABLED_ROLES

Допустимые значения: 0–148
Значение по умолчанию: 20

Указывает максимальное количество ролей базы данных, которые может ввести пользователь в дополнение к собственной роли и роли PUBLIC.

O7_DICTIONARY_ACCESSIBILITY

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Управляет ограничениями для привилегий пользователя SYSTEM. Если значение равно TRUE, то доступ к объектам схемы SYS разрешен (поведение, присущее серве-

ру Oracle версии 7), а если – FALSE, то привилегии SYSTEM, разрешающие доступ к объектам других схем, не разрешают доступ к объектам схемы словаря данных. До версии Oracle9*i* значением параметра по умолчанию было TRUE.

OS_AUTHENT_PREFIX

Допустимые значения: любая корректная строка

Значение по умолчанию: OPS$

Указывает префикс, добавляемый в начало каждого имени пользователя ОС для того, чтобы дать серверу Oracle возможность проводить аутентификацию пользователей на основе имени учетной записи ОС и пароля. При попытке соединения значение имени пользователя, дополненное префиксом, сравнивается с именами пользователей Oracle из базы данных. Механизм не работает для учетных записей Oracle, созданных без указания ключевых слов IDENTIFIED EXTERNALLY.

OS_ROLES

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Определяет, разрешает ли сервер Oracle операционной системе идентифицировать роли пользователей. Если параметр равен TRUE, то при попытке пользователя создать сеанс происходит инициализация домена защиты имени пользователя на основании ролей, распознанных ОС. Кроме того, в этом случае операционная система полностью управляет выдачей ролей для всех имен пользователей БД. Отмена ролей, назначенных операционной системой, будет проигнорирована, как и ранее выданные роли. Если значение параметра равно FALSE, роли определяются и управляются сервером БД.

RDBMS_SERVER_DN

Допустимые значения: любое характерное имя X.500.

Значение по умолчанию: нет

Указывает характерное имя сервера Oracle для извлечения ролей для службы каталогов предприятия. Не задавайте этот параметр, если предпочитаете использовать только аутентификацию SSL.

REMOTE_LOGIN_PASSWORDFILE

Допустимые значения: NONE | EXCLUSIVE | SHARED

Значение по умолчанию: NONE

Указывает, проверяет ли сервер Oracle файл паролей и сколько баз данных может использовать один и тот же файл паролей. NONE означает, что Oracle будет игнорировать любые файлы паролей, т. е. привилегированные пользователи должны аутентифицироваться операционной системой. EXCLUSIVE показывает, что файл паролей может использоваться всего одной базой данных и при этом может содержать имена, отличные от SYS и INTERNAL. Если же параметр установлен в SHARED, то использовать файл паролей могут несколько баз данных, но содержаться в нем будут только пользователи SYS и INTERNAL. Значения параметра для всех экземпляров должны совпадать.

SQL92_SECURITY

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Определяет, необходимы ли привилегии SELECT уровня таблицы для выполнения операций обновления и удаления, ссылающихся на значения столбцов таблицы.

Разделяемый сервер/Многопоточный сервер

Параметры разделы описывают и регулируют работу разделяемого сервера в Oracle9*i* (Shared Server) и многопоточного сервера (Multi-Threaded Server – MTS) до версии Oracle9*i*.

CIRCUITS

Допустимые значения: целое число
Значение по умолчанию: см. описание

Указывает общее количество виртуальных каналов, доступных для сетевых соединений в среде Shared Server. По умолчанию значение параметра равно SESSIONS, если используется MTS; в противном случае значение по умолчанию равно 0. Появился в Oracle9*i*.[1]

DISPATCHERS

Допустимые значения: см. описание
Значение по умолчанию: нет
Динамическое изменение: ALTER SYSTEM

Конфигурирует диспетчерские процессы в среде Shared Server. Появился в Oracle9*i*.

Синтаксис

```
DISPATCHERS =
'{ (PROTOCOL = протокол)
 | (ADDRESS = адрес)
 | (DESCRIPTION = описание}
}
[({DISPATCHERS = целое_число
  | SESSIONS = целое_число
```

[1] Для обеспечения согласованности в Oracle9*i* был внесен ряд изменений в наименования сетевых продуктов, опций и параметров. Так, Multi-Threaded Server (MTS) был переименован в Shared Server. В связи с этим были переименованы и соответствующие параметры:

MTS_CIRCUITS → CIRCUITS
MTS_DISPATCHERS → DISPATCHERS
MTS_MAX_DISPATCHERS → MAX_DISPATCHERS
MTS_MAX_SERVERS → MAX_SHARED_SERVERS
MTS_SERVERS → SHARED_SERVERS
MTS_SESSIONS → SHARED_SERVER_SESSIONS. – *Примеч. перев.*

```
    | CONNECTIONS = целое_число
    | TICKS = секунды
    | POOL = {1 | ON | YES | TRUE | BOTH |({IN | OUT} = целое_число) |
              0 | OFF | NO | FALSE | целое_число }
  | MULTIPLEX = {1 | ON | YES | TRUE |
                 0 | OFF | NO | FALSE | BOTH | IN | OUT}
  | LISTENER = tnsname
  | SERVICE = служба
  | INDEX = целое число
  })]`
```

Ключевые слова

протокол	Сетевой протокол для диспетчера.
адрес	Адрес сетевого протокола для диспетчера.
описание	Описание сети для диспетчера.
DISPATCHERS	Начальное количество диспетчеров.
SESSIONS	Максимальное количество сетевых сеансов для одного диспетчера.
CONNECTIONS	Максимальное количество сетевых соединений для одного диспетчера.
TICKS	Длительность сетевого тика в секундах.
POOL	Управляет организацией пула соединений. Значения ON, YES, TRUE и BOTH задают пул для сетевых соединений IN и OUT. Значения NO, OFF и FALSE задают отсутствие пула сетевых соединений IN и OUT. IN и OUT указывают пул для соответствующих сетевых соединений. Целое число задает тайм-аут для обоих типов соединений. Число можно указать для IN или OUT, например (IN = 20)(OUT = 10), задав тем самым тик для каждого направления. Тайм-аут по умолчанию равен 10.
MULTIPLEX	Значения 1, ON, YES, TRUE и BOTH разрешают мультиплексирование сетевых сеансов для сетевых соединений IN и OUT. Значения 0, NO, OFF и FALSE запрещают. Значения IN и OUT разрешают мультиплексирования в соответствующих направлениях.
LISTENER	Сетевое имя адреса или списка адресов процессов прослушивания Oracle Net, где будут регистрироваться диспетчерские процессы.
SERVICE	Имена служб, которые диспетчеры регистрируют при помощи процессов прослушивания.
INDEX	Применяется для идентификации конкретного диспетчера при использовании в команде ALTER SYSTEM SET DISPATCHERS. Обратите внимание, что нумерация начинается с 0.

MAX_DISPATCHERS

Допустимые значения: 5 или количество сконфигурированных диспетчеров (большее из них)

Значение по умолчанию: 5

Указывает максимальное количество диспетчерских процессов, для которых разрешен одновременный запуск (применяется, только если диспетчеры сконфигурированы). Значение параметра должно быть не меньше, чем частное от деления максимального количества параллельных сеансов на количество соединений для каждого

диспетчера (хорошую производительность обеспечивают 250 сеансов для одного диспетчера). Появился в Oracle9*i*.

MAX_SHARED_SERVERS

Допустимые значения: зависит от операционной системы

Значение по умолчанию: 2 * SHARED_SERVERS или 20

Указывает максимальное количество процессов Shared Server, которые могут работать одновременно.

SHARED_SERVERS

Допустимые значения: зависит от операционной системы

Значение по умолчанию: 1 при использовании Shared Server, иначе 0

Динамическое изменение: ALTER SYSTEM

Указывает количество серверных процессов, создаваемых при запуске экземпляра. Появился в Oracle9*i*.

MTS_CIRCUITS

Допустимые значения: целое число

Значение по умолчанию: см. описание

Указывает общее количество виртуальных каналов, доступных для сетевых соединений в среде Multi-Threaded Server/Shared Server. Значение по умолчанию равно SESSIONS, если используется MTS; в противном случае равно 0. Появился в Oracle8*i*. Начиная с версии Oracle9*i* компания Oracle рекомендует применять вместо него параметр CIRCUITS.

MTS_DISPATCHERS

Допустимые значения: строка

Значение по умолчанию: нет

Динамическое изменение: ALTER SYSTEM

Конфигурирует диспетчерские процессы в среде Multi-Threaded Server/Shared Server. Начиная с версии Oracle9*i* компания Oracle рекомендует применять вместо него параметр DISPATCHERS.

Синтаксис

```
DISPATCHERS =
'{ (PROTOCOL = протокол)
 | (ADDRESS = адрес)
 | (DESCRIPTION = описание}
}
[({DISPATCHERS = целое число
  | SESSIONS = целое число
  | CONNECTIONS = целое число
  | TICKS = секунды
  | POOL = {1 | ON | YES | TRUE | BOTH |({IN | OUT} = целое число) |
            0 | OFF | NO | FALSE | целое число }
```

```
| MULTIPLEX = {1 | ON | YES | TRUE |
              0 | OFF | NO | FALSE | BOTH | IN | OUT}
| LISTENER = tnsname
| SERVICE = служба
| INDEX = целое число
})]`
```

Ключевые слова

Ключевые слова рассмотрены в описании параметра DISPATCHERS ранее в этом же разделе.

MTS_LISTENER_ADDRESS

Допустимые значения: допустимая спецификация MTS_LISTENER_ADDRESS

Значение по умолчанию: NULL

Определяет конфигурацию процесса прослушивания – адреса, которые тот будет прослушивать на предмет запросов соединения для каждого используемого в системе сетевого протокола. Каждый адрес должен быть указан в отдельном параметре. Заменен параметрами LOCAL_LISTENER и LISTENER в Oracle8. Исключен в Oracle9*i*.

MTS_MAX_DISPATCHERS

Допустимые значения: зависит от операционной системы

Значение по умолчанию: 5 или количество сконфигурированных диспетчеров (большее из них)

Указывает максимальное количество диспетчерских процессов, для которых разрешен одновременный запуск. Начиная с версии Oracle9*i* следует вместо него применять параметр MAX_DISPATCHERS.

MTS_MAX_SERVERS

Допустимые значения: зависит от операционной системы

Значение по умолчанию: 2 * SHARED_SERVERS или 20

Указывает максимальное количество процессов Multi-Threaded Server/Shared Server, которые могут работать одновременно. Начиная с версии Oracle9*i* следует применять вместо него параметр MAX_SHARED_SERVERS.

MTS_MULTIPLE_LISTENERS

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Определяет синтаксис параметра MTS_LISTENER_ADDRESS. Исключен в Oracle9*i*.

MTS_RATE_LOG_SIZE

Допустимые значения: см. описание

Значение по умолчанию: 10 для каждого приведенного имени

Указывает объем выборки, по которой вычисляются статистики работы диспетчеров. Объем выборки определяет, сколько будет занято памяти и с какой частотой будут вычисляться максимальные показатели. Исключен в Oracle8*i*.

MTS_RATE_LOG_SIZE принимает строку, состоящую из одного или более разделов (*имя = значение*), и значения распределяются между всеми диспетчерами. Список возможных имен:

DEFAULTS	Заменяет количество регистрируемых в журнале событий по умолчанию для неопределенных статистик
EVENT_LOOPS	Количество циклов событий для регистрации
MESSAGES	Количество сообщений для регистрации
SERVER_BUFFERS	Количество журналируемых буферов, передаваемых на сервер
CLIENT_BUFFERS	Количество журналируемых буферов, передаваемых клиенту
TOTAL_BUFFERS	Количество журналируемых буферов, передаваемых в обоих направлениях
IN_CONNECTS	Количество регистрируемых входящих соединений
OUT_CONNECTS	Количество регистрируемых исходящих соединений
RECONNECTS	Количество регистрируемых повторных соединений в пуле соединений

MTS_RATE_SCALE

Допустимые значения: см. описание

Значение по умолчанию: см. описание

Определяет темп выдачи статистических сообщений диспетчера в сотых долях секунды. Исключен в Oracle8*i*. Значения по умолчанию:

DEFAULTS	нет
EVENT_LOOPS	6000
MESSAGES	100
SERVER_BUFFERS	10
CLIENT_BUFFERS	10
TOTAL_BUFFERS	10
IN_CONNECTS	6000
OUT_CONNECTS	6000
RECONNECTS	6000

MTS_SERVERS

Допустимые значения: зависит от операционной системы

Значение по умолчанию: 1 при использовании MTS, иначе 0

Указывает количество серверных процессов, создаваемых при запуске экземпляра. Начиная с версии Oracle9*i* следует вместо него применять параметр SHARED_SERVERS.

MTS_SERVICE

Допустимые значения: имя службы

Значение по умолчанию: NULL

Указывает имя службы, ассоциированной с диспетчером. Если это имя задано в строке CONNECT, то пользователь может установить соединение с экземпляром через диспетчер. Проследите за тем, чтобы имя было уникальным и не было заключено в кавычки. Если параметр не задан, то значение по умолчанию для MTS_SERVICE совпадает со значением, указанным в DB_NAME. Если DB_NAME тоже не задан, сервер Oracle возвращает ошибку при запуске, выводя сообщение об отсутствии значения данного параметра. Исключен в Oracle8*i*.

SHARED_SERVER_SESSIONS

Допустимые значения: от 0 до SESSIONS–5

Значение по умолчанию: меньшее из значений CIRCUITS и SESSIONS–5

Указывает общее количество пользовательских сеансов соединений в среде Shared Server. Появился в Oracle9*i*.

Сортировки

Параметры данного раздела управляют выполнением сортировок.

SORT_DIRECT_WRITES

Допустимые значения: AUTO | TRUE | FALSE

Значение по умолчанию: AUTO

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Определяет, могут ли сортировки записывать промежуточные результаты на диск, минуя кэш буферов. Если выбрано значение AUTO и размер области сортировки превышает десятикратный размер блока, то память выделяется из области сортировки. Если параметру присвоено значение TRUE, то дополнительные буферы выделяются из памяти при каждой сортировке и может потребоваться дополнительный временный сегмент. Если параметр установлен в FALSE, то сортировки осуществляют запись на диск через кэш буферов. Исключен в Oracle8*i*.

SORT_READ_FAC

Допустимые значения: зависит от операционной системы

Значение по умолчанию: зависит от операционной системы

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Определяет отношение интервала времени, необходимого для чтения отдельного блока базы данных, к скорости передачи блоков. Исключен в Oracle8*i*.

SORT_SPACEMAP_SIZE

Допустимые значения: зависит от операционной системы

Значение по умолчанию: зависит от операционной системы

Размер (в байтах) рабочей области сортировки в контекстной области. Лишь при создании очень больших индексов имеет смысл изменять этот параметр. Сортировка автоматически увеличивает этот размер, если необходимо, но она не обязательно делает

это, если сочтет нужным задействовать дисковую память. Сортировка расходует дисковую память оптимально, если значение SORT_SPACEMAP_SIZE установлено следующим образом: (*всего_байт_для_сортировки*)/SORT_AREA_SIZE + 64, где *всего_байт_для_сортировки* рассчитывается как (*число_записей*) * [*сумма_средних_размеров_столбцов* + (2 * *число_столбцов*)]. Исключен в Oracle8*i*.

SORT_WRITE_BUFFER_SIZE

Допустимые значения: 32768–65536

Значение по умолчанию: 32768

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Задает размер буфера ввода/вывода сортировки для случая, когда значение параметра SORT_DIRECT_WRITES равно TRUE. Рекомендуется для использования при симметричной репликации. Исключен в Oracle8*i*.

SORT_WRITE_BUFFERS

Допустимые значения: 2–8

Значение по умолчанию: 2

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Указывает количество буферов сортировки для случая, когда значение параметра SORT_DIRECT_WRITES равно TRUE. Рекомендуется для использования при симметричной репликации. Исключен в Oracle8*i*.

Резервные базы данных

Использование резервных баз данных регулируется параметрами данного раздела.

ACTIVE_INSTANCE_COUNT

Допустимые значения: положительное целое число

В кластере из двух экземпляров значение 1 означает, что запущенный первым экземпляр считается основным, а второй работает как резервный. Любые другие значения игнорируются. Появился в Oracle9*i*.

LOCK_NAME_SPACE

Допустимые значения: максимум 8 буквенно-цифровых символов. Применение специальных символов запрещено

Значение по умолчанию: нет

Указывает пространство имен, которое распределенный диспетчер блокировок использует для формирования названий блокировок.

STANDBY_ARCHIVE_DEST

Допустимые значения: любой действительный путь или имя устройства

Значение по умолчанию: зависит от операционной системы

Динамическое изменение: ALTER SYSTEM

Определяет местонахождение архивных журнальных файлов, прибывающих на резервную базу данных в режиме управляемого восстановления. Появился в Oracle9*i*.

STANDBY_FILE_MANAGEMENT

Допустимые значения: MANUAL | AUTO
Значение по умолчанию: MANUAL
Динамическое изменение: ALTER SYSTEM

Указывает, разрешено ли автоматическое управление резервными файлами. Если значение равно AUTO, то операции управления файлами, такие как добавление и удаление файлов, выполняются для резервной базы данных автоматически. Появился в Oracle9*i*.

STANDBY_PRESERVES_NAMES

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE
Динамическое изменение: ALTER SYSTEM

Определяет, совпадают ли имена файлов резервной базы данных с именами файлов основной базы данных. Появился в Oracle9*i*.

Системные операции

Параметры этого раздела относятся к различным системным процессам, работе сервера Oracle в целом, файлам дампа и управляющим файлам.

BACKGROUND_CORE_DUMP

Допустимые значения: FULL | PARTIAL
Значение по умолчанию: FULL

Значение определяет, включается ли SGA в дамп ядра. FULL – включать, PARTIAL – не включать.

BACKGROUND_DUMP_DEST

Допустимые значения: строка, содержащая действительное имя каталога
Значение по умолчанию: зависит от операционной системы
Динамическое изменение: ALTER SYSTEM

Указывает путь к каталогу, в который записываются отладочные файлы трассировки для фоновых процессов (LGWR, DBWR и т. д.) во время работы Oracle.

CONTROL_FILE_RECORD_KEEP_TIME

Допустимые значения: 0–365
Значение по умолчанию: 7

Динамическое изменение: ALTER SYSTEM

Определяет минимальное количество дней, которое запись должна провести в циклически повторно используемой секции управляющего файла, прежде чем та может быть повторно использована. Если необходимо добавить новую запись в секцию, допускающую повторное использование, а старейшая запись еще недостаточно стара, то секция расширяется. Если параметр установлен в 0, то допускающие повторное использование секции никогда не расширяются и записи используются повторно по мере необходимости. К повторно используемым относятся следующие секции управляющего файла:

ARCHIVED LOG
BACKUP CORRUPTION
BACKUP DATAFILE
BACKUP PIECE
BACKUP REDO LOG
BACKUP SET
COPY CORRUPTION
DATAFILE COPY
DELETED OBJECT
LOGHISTORY
OFFLINE RANGE

CONTROL_FILES

Допустимые значения: от 1 до 8 имен файлов
Значение по умолчанию: зависит от операционной системы

Указывает одно или несколько имен управляющих файлов, разделенных запятыми.

CORE_DUMP_DEST

Допустимые значения: действительный путь к каталогу
Значение по умолчанию: *$ORACLE_HOME/dbs/*
Динамическое изменение: ALTER SYSTEM

Определяет каталог, в который записываются файлы дампа ядра.

DB_BLOCK_CHECKPOINT_BATCH

Допустимые значения: от 0 до вычисляемого
Значение по умолчанию: 8
Динамическое изменение: ALTER SYSTEM

Указывает максимальное число блоков, которое процесс DBWR будет записывать одним пакетом для контрольной точки. Установка маленького значения DB_BLOCK_CHECKPOINT_BATCH препятствует переполнению системы ввода/вывода операциями записи для контрольной точки и позволяет записывать на диск другие модифицированные блоки. Если установить более высокое значение, то контрольные точки будут выполняться быстрее. Если параметр DB_BLOCK_CHECKPOINT_BATCH равен 0, будет взято значение по умолчанию. Если указанное значение слишком велико, сервер Oracle уменьшает его до максимального количества блоков, которые могут быть записаны в базу данных одним пакетом. Исключен в Oracle8*i*.

DB_BLOCK_CHECKSUM

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Динамическое изменение: ALTER SYSTEM

Определяет, будут ли процесс DBWR и непосредственный загрузчик вычислять контрольную сумму и сохранять ее в заголовке каждого блока данных при записи на диск. Установка DB_BLOCK_CHECKSUM в TRUE может несколько увеличить накладные расходы. Рекомендуется устанавливать параметр в TRUE только после консультации с группой поддержки Oracle.

DB_BLOCK_LRU_EXTENDED_STATISTICS

Допустимые значения: от 0 до определяемого в зависимости от памяти системы

Значение по умолчанию: 0

Включает или выключает сбор статистик в таблице X$KCBRBH, которые измеряют эффект увеличения количества буферов в буферном кэше в SGA. В Oracle8 такие статистики доступны также в V$RECENT_BUCKET. Этот инструмент настройки должен быть отключен (0) во время нормальной работы. Исключен в Oracle8*i*.

DB_BLOCK_LRU_LATCHES

Допустимые значения: от 1 до количества процессоров

Значение по умолчанию: CPU_COUNT / 2

Указывает верхнюю границу количества наборов защелок LRU. Если параметр не установлен, то сервер Oracle вычисляет значение, которое обычно является адекватным. Следует увеличивать его, только если промахи составляют более 3% в V$LATCH. Исключен в Oracle9*i*.

DB_BLOCK_LRU_STATISTICS

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Выключает или включает сбор статистики в таблице X$KCBCBH, которая измеряет эффект уменьшения количества буферов в буферном кэше в SGA. Такие статистики доступны также в V$CURRENT_BUCKET. Этот инструмент настройки должен быть отключен (FALSE) во время нормальной работы. Исключен в Oracle8*i*.

DB_BLOCK_MAX_DIRTY_TARGET

Допустимые значения: от 100 до общего количества буферов кэша

Значение по умолчанию: общее количество буферов кэша

Определяет количество буферов, которые могут быть «грязными» – измененными и отличающимися от того, что записано на диске. Если количество измененных буферов в буферном кэше превышает указанное значение, то DBWR осуществит запись буферов, чтобы попытаться сделать количество измененных буферов меньше указанного. Параметр может влиять на длительность восстановления экземпляра, т. к. скорость восстановления связана с количеством буферов, измененных на момент сбоя. Чем меньше значение параметра, тем быстрее будет восстановлен экземпляр. Установка значения в 0 отключает запись буферов для накопления контрольных точек; все остальные операции записи продолжаются без изменений. Исключен в Oracle9*i*.

DB_WRITER_PROCESSES

Допустимые значения: 1–20

Значение по умолчанию: 1

Определяет количество процессов записи в базу данных, создаваемых для экземпляра.

DBWR_IO_SLAVES

Допустимые значения: от 0 до определяемого в зависимости от операционной системы

Значение по умолчанию: 0

Указывает количество подчиненных процессов ввода/вывода, с помощью которых процесс DBW0 осуществляет запись на диск и имитирует асинхронный ввод/вывод для платформ, не поддерживающих асинхронный ввод/вывод или реализующих его неэффективно. Параметр действует, только если значение параметра DB_WRITER_PROCESSES равно 1. Если установить этот параметр, то параметру DB_WRITER_PROCESSES присваивается значение 1.

MAX_DUMP_FILE_SIZE

Допустимые значения: целое число или UNLIMITED

Значение по умолчанию: UNLIMITED

Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Указывает максимальный размер записываемых файлов трассировки. Допустимые значения: число (в блоках операционной системы); число, за которым следует буква K или M; строка UNLIMITED.

PROCESSES

Допустимые значения: от 6 до определяемого в зависимости от операционной системы

Значение по умолчанию: вычисляется

Определяет максимальное число пользовательских процессов операционной системы, которые могут быть одновременно соединены с сервером Oracle. При вычислении этого значения необходимо учесть все фоновые процессы, включая процессы блокирования, процессы очередей заданий и параллельных запросов. Значение по умолчанию вычисляется на основе PARALLEL_MAX_SERVERS. Данный параметр определяет значения по умолчанию для SESSIONS, так что если вы изменяете значение PROCESSES, то может потребоваться изменить значение SESSIONS.

RESOURCE_LIMIT

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Динамическое изменение: ALTER SYSTEM

Определяет режим действия ограничений на ресурсы, устанавливаемый в профилях БД.

RESOURCE_MANAGER_PLAN

Допустимые значения: любая действительная символьная строка
Значение по умолчанию: нет
Динамическое изменение: ALTER SYSTEM

Указывает план ресурсов самого высокого уровня, который может использоваться для экземпляра. Если параметр не установлен, то диспетчер ресурсов по умолчанию отключен.

SESSION_MAX_OPEN_FILES

Допустимые значения: от 1 до наименьшего из 50 или до MAX_OPEN_FILES
Значение по умолчанию: 10

Указывает максимальное количество файлов BFILE, которые могут быть открыты в любом сеансе. Когда указанное количество достигнуто, последующие попытки открытия файлов в рамках сеанса будут неудачными. Значение этого параметра также зависит от аналогичного параметра, определенного для базовой операционной системы.

SESSIONS

Допустимые значения: 1–231
Значение по умолчанию: 1,1 * PROCESSES + 5

Определяет общее количество пользовательских и системных сеансов. Значения по умолчанию для параметров ENQUEUE_RESOURCES и TRANSACTIONS вычисляются на основе SESSIONS. Если вы изменяете значение SESSIONS, то может потребоваться изменение значений вычисляемых параметров. При работе в среде Shared Server/Multi-Threaded Server следует устанавливать значение SESSIONS равным 1,1 * (*общее количество соединений*).

SHADOW_CORE_DUMP

Допустимые значения: FULL | PARTIAL
Значение по умолчанию: PARTIAL

Определяет, будет ли SGA включена в дамп ядра. Если параметр равен FULL, то SGA включается в дамп ядра, а если – PARTIAL, то дамп SGA не выполняется.

TEMPORARY_TABLE_LOCKS

Допустимые значения: от 0 до определяемого в зависимости от операционной системы
Значение по умолчанию: SESSIONS

Определяет количество временных таблиц, которые могут быть созданы в пространстве временных сегментов. Блокировка для временной таблицы требуется при каждой сортировке, размер которой слишком велик для того, чтобы уместиться в памяти. Такая ситуация возможна, например, при выборке по большой таблице с инструкцией ORDER BY или при сортировке большого индекса. Исключен в Oracle8*i*.

THREAD

Допустимые значения: от 0 до максимального количества включенных потоков
Значение по умолчанию: 0

Задает номер потока журнала, который должен использоваться экземпляром. Параметр применим только к экземплярам, работающим в режиме Real Application Clusters или Parallel Server. Можно указать любой доступный номер потока, но экземпляр не может использовать тот же номер, который уже используется другим экземпляром. Кроме того, экземпляр не может быть запущен, если его поток журнала отключен. Нулевое значение приводит к выбору доступного включенного общего потока. Потоки журнала задаются с помощью опции THREAD команды ALTER DATABASE ADD LOGFILE и включаются посредством команды ALTER DATABASE ENABLE [PUBLIC] THREAD. Ключевое слово PUBLIC указывает, что данный поток журнала может использоваться любым экземпляром. Поток 1 является потоком по умолчанию в монопольном режиме, но экземпляр, работающий в монопольном режиме, может установить параметр THREAD, чтобы использовать файлы журнала в потоке, отличном от потока 1.

USE_ISM

Допустимые значения: TRUE, FALSE
Значение по умолчанию: TRUE

Определяет, включена ли общая таблица страниц (*shared page table*). Исключен в Oracle8*i*.

Oracle Trace (трассировка)

Работой Oracle Trace управляют параметры, описанные в данном разделе.

ORACLE_TRACE_COLLECTION_NAME

Допустимые значения: действительное имя длиной до 16 символов
Значение по умолчанию: нет

Задает имя файла сбора трассировки Oracle Trace. Кроме того, параметр используется в выходных именах файлов в файле определения сбора *.cdf* и файле данных *.dat* совместно с описанным ниже параметром ORACLE_TRACE_COLLECTION_PATH.

ORACLE_TRACE_COLLECTION_PATH

Допустимые значения: действительный путь к каталогу
Значение по умолчанию: зависит от операционной системы

Задает каталог, в котором расположены файл определения сбора трассировки и файлы данных Oracle Trace.

ORACLE_TRACE_COLLECTION_SIZE

Допустимые значения: 0–4294967295
Значение по умолчанию: 5242880

Указывает максимальный размер (в байтах) файла сбора трассировки Oracle Trace. Когда размер достигает максимума, сбор трассировки отключается.

ORACLE_TRACE_ENABLE

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE
Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Определяет, разрешен ли сбор трассировочной информации для сервера. Значение TRUE не запускает сбор трассировки, а разрешает серверу Oracle использовать Oracle Trace. В этом случае можно запустить Oracle Trace, используя приложение Oracle Trace Manager и Oracle Enterprise Manager Diagnostic Pack. Используйте интерфейс командной строки Oracle Trace или укажите имя в параметре ORACLE_TRACE_COLLECTION_NAME.

ORACLE_TRACE_FACILITY_NAME

Допустимые значения: ORACLED | ORACLEE | ORACLESM | ORACLEC
Значение по умолчанию: ORACLED

Указывает файл определения продукта (*product definition file*) Oracle Trace (с расширением *.fdf*). Файл должен храниться в каталоге, указанном в параметре ORACLE_TRACE_FACILITY_PATH. Файл определения продукта содержит определяющую информацию для всех событий и элементов данных, которые могут быть собраны для продукта, использующего Oracle Trace Data Collection API. Доступны следующие файлы определения продукта:

ORACLE	Набор событий ALL
ORACLED	Набор событий DEFAULT
ORACLEE	Набор событий EXPERT
ORACLESM	Набор событий SUMMARY
ORACLEC	Набор событий CACHEIO

ORACLE_TRACE_FACILITY_PATH

Допустимые значения: действительный путь к каталогу
Значение по умолчанию: зависит от операционной системы

Указывает путь к каталогу, где находятся файлы определения средств сбора трассировки Oracle Trace.

SQL_TRACE

Допустимые значения: TRUE | FALSE
Значение по умолчанию: FALSE

Включает или отключает SQL Trace. Может быть изменен при помощи пакета DBMS_SYSTEM.

TRACE_ENABLED

Допустимые значения: TRUE | FALSE
Значение по умолчанию: TRUE
Динамическое изменение: ALTER SYSTEM, ALTER SESSION

Определяет, трассируется ли история выполнения. Значения параметра для различных экземпляров должны совпадать. Появился в Oracle9*i*.

TRACEFILE_IDENTIFIER

Допустимые значения: любое действительное имя файла
Значение по умолчанию: нет
Динамическое изменение: ALTER SESSION

Задает произвольный идентификатор, который становится частью имени файла Oracle Trace для фоновых процессов. Появился в Oracle9*i*.

USER_DUMP_DEST

Допустимые значения: действительный путь к каталогу
Значение по умолчанию: зависит от операционной системы
Динамическое изменение: ALTER SYSTEM.

Указывает каталог, в который сервер записывает отладочные файлы трассировки от имени пользовательского процесса.

Прочие параметры

В этом разделе собраны параметры, не попадающие ни в одну из категорий, описанных ранее.

ALWAYS_ANTI_JOIN

Допустимые значения: NESTED_LOOPS | MERGE | HASH
Значение по умолчанию: NESTED_LOOPS

Задает тип антисоединения, используемого сервером Oracle. Исключен в Oracle9*i*.

Ключевые слова

NESTED_LOOPS

Сервер использует *антисоединение методом вложенных циклов* (*nested loop antijoin*).

MERGE

Сервер использует *антисоединение методом сортировки и слияния* (*sort merge antijoin*).

HASH

Для вычисления подзапроса сервер использует *антисоединение методом хеширования* (*hash antijoin*).

AQ_TM_PROCESSES

Допустимые значения: 0 | 1 до версии Oracle9*i*; 0–10 в Oracle9*i*

Значение по умолчанию: 0

Динамическое изменение: ALTER SYSTEM

Если значение параметра равно 1, то для отслеживания сообщений создается одноразовый диспетчерский процесс. Если параметр не установлен или его значение равно 0, то диспетчер не создается.

COMPATIBLE

Допустимые значения: от 7.3.0 до текущей версии Oracle

Значение по умолчанию: текущая версия Oracle (например, 9.2.0)

Позволяет использовать новую версию, в то же время обеспечивая обратную совместимость с более ранней версией. Некоторые возможности новой версии могут не поддерживаться.

Если вы используете резервную базу данных, то данный параметр должен иметь одинаковые значения для обеих БД и это значение должно быть не меньше, чем 7.3.0.0.0.

COMPATIBLE_NO_RECOVERY

Допустимые значения: от версия Oracle по умолчанию до текущей версии Oracle

Значение по умолчанию: зависит от версии

Работает как параметр COMPATIBLE, за исключением того, что более старая версия может быть неприменима для текущей базы данных, если необходимо восстановление. Значением по умолчанию является самая ранняя из версий, с которой гарантирована совместимость. Исключен в Oracle8*i*.

COMPLEX_VIEW_MERGING

Допустимые значения: TRUE | FALSE

Значение по умолчанию: FALSE

Определяет, следует ли разрешать слияние составных представлений. Исключен в Oracle9*i*.

EVENT

Допустимые значения: предоставляется службой поддержки Oracle Support

Значение по умолчанию: нет

Используется для отладки системы. Устанавливайте параметр только под руководством группы поддержки Oracle.

FIXED_DATE

Допустимые значения: действительная дата в описанном ниже формате

Значение по умолчанию: нет

Указывает постоянную дату, которую всегда будет возвращать SYSDATE вместо текущей даты. Формат даты таков:

```
YYYY-MM-DD-HH24:MI:SS
```

Также принимает формат даты Oracle по умолчанию – без времени. Значение может быть указано в двойных кавычках (не в одинарных) или без кавычек, например FIXED_DATE = "30-nov-02" или FIXED_DATE = 30-nov-02.

LOCAL_LISTENER

Допустимые значения: разрешенная спецификация процесса прослушивания

Значение по умолчанию: см. описание

Идентифицирует «локальные» процессы прослушивания, чтобы они могли обеспечить соединение клиента с назначенным сервером. Указывает сетевое имя для адреса или списка адресов, которые должны работать на той же машине, что и экземпляр. Если параметр LOCAL_LISTENER указан, он подменяет значения MTS_LISTENER_ADDRESS и MTS_MULTIPLE_LISTENERS. Значение по умолчанию:

```
(ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP) (HOST=localhost) (Port=1521))
                (ADDRESS =(PROTOCOL = IPC)
                (KEY= dbname)))
```

LOGMNR_MAX_PERSISTENT_SESSIONS

Допустимые значения: 1–LICENSE_MAX_SESSIONS

Значение по умолчанию: 1

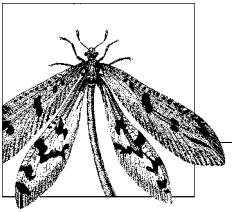
Определяет максимальное количество постоянных сеансов LogMiner, которые активны одновременно. В кластеризованной среде следует устанавливать значение параметра равным количеству узлов кластера. Появился в Oracle9*i*.

REPLICATION_DEPENDENCY_TRACKING

Допустимые значения: TRUE | FALSE

Значение по умолчанию: TRUE

Слежение за зависимостями операций чтения/записи в БД. Отслеживание зависимостей важно для параллельного распространения изменений в среде реплицирования. Значение TRUE включает отслеживание зависимостей. Значение FALSE позволяет операциям чтения/записи в БД выполняться быстрее, но не предоставляет информации о зависимостях для параллельного распространения изменений в среде реплицирования. Пользователям не стоит устанавливать параметр, за исключением тех случаев, когда приложения вообще не осуществляют операций чтения/записи для реплицируемых таблиц.



3

Конкурентный доступ

Одна из главнейших возможностей СУБД заключается в способности обеспечить работу нескольких пользователей. В Oracle реализована уникальная модель поддержки конкурентного доступа, именуемая *многоверсионной моделью согласованности по чтению* (*Multiversion Read Consistency* – MVRC). Данная функциональность представляет собой огромное преимущество для разработчиков, поскольку снимает с них большую часть забот о блокировках, возникающих в их коде. Возможность поддержки MVRC встроена в самые глубины архитектуры СУБД Oracle.

В главе представлены основы реализации конкурентного режима и кратко описана собственно работа MVRC.

Основные понятия

Для осознания самой идеи конкурентного доступа и его реализации в СУБД Oracle необходимо определить несколько базовых понятий.

Транзакции

Транзакция (*transaction*) – это основа целостности данных в многопользовательских СУБД и основа всех моделей конкурентного доступа. Транзакцией называется единая и неделимая последовательность операций над данными. Либо все изменения данных, выполненные в рамках транзакции, единовременно применяются к реляционной базе данных командой COMMIT, либо все измененные данные единовременно возвращаются в исходное состояние командой ROLLBACK. После того как транзакция зафиксирована командой COMMIT, выполненные изменения делаются постоянными и становятся видны другим транзакциям и другим пользователям.

Выполнение транзакции всегда занимает некоторое время, хотя для их большей части этот интервал времени очень невелик. Изменения, производимые транзакцией, вступают в силу только после ее фиксации, поэтому каждая отдельная транзакция должна быть изолирована от воздействия других транзакций. Механизм, реализующий *изоляцию транзакции*, называется блокировкой.

Блокировки

Для предотвращения взаимного влияния транзакций в СУБД применяется система *блокировок* (*locks*). О взаимодействии транзакций говорят, когда одна транзакция

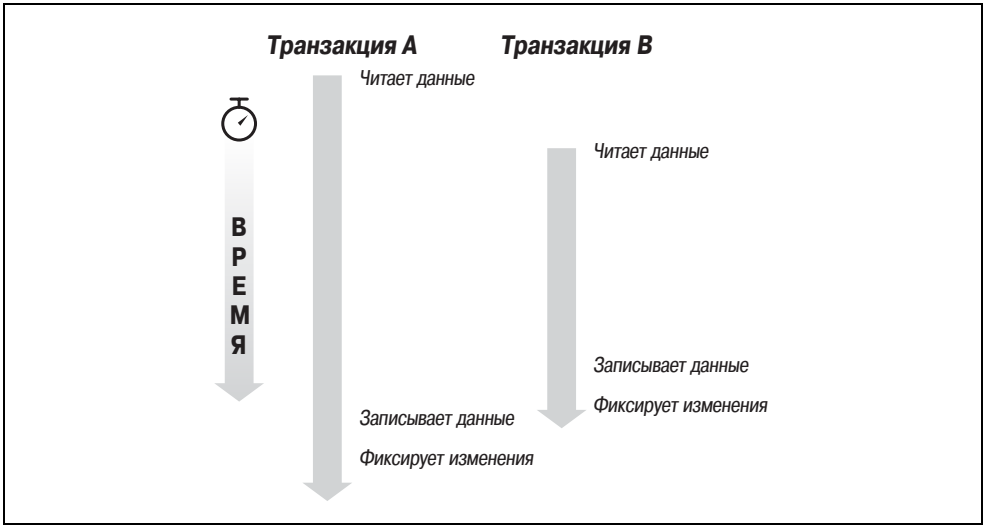


Рис. 3.1. *Транзакции во времени*

пытается изменить элемент данных, который в данный момент изменяет другая транзакция. Возможную проблему иллюстрирует рис. 3.1. Транзакция A читает элемент данных. Транзакция B читает тот же самый элемент и фиксирует изменение данных. Когда транзакция A доходит до этапа фиксации, сделанные ею изменения непредумышленно записываются поверх изменений, выполненных транзакцией B, в результате чего утрачивается целостность данных.

Обычно для разрешения подобных проблем применяются блокировки двух типов. Первый тип – это *блокировка записи* (*write lock*), или *монопольная блокировка* (*exclusive lock*). Монопольная блокировка применяется и удерживается в течение того времени, когда транзакция изменяет данные, а освобождается при завершении транзакции командой COMMIT или ROLLBACK. Блокировка записи предоставляет право на доступ к ресурсам единственному пользователю, который только и может изменить заблокированные данные.

В некоторых СУБД также применяются *блокировки чтения* (*read locks*), или *разделяемые блокировки* (*shared locks*). Блокировка чтения может применяться любым количеством пользователей, которые просто читают данные, т. к. доступ только для чтения не создает конфликтных ситуаций. Однако блокировка чтения означает, что для данных нельзя будет применить блокировку записи, т. к. блокировка записи является монопольной. Если на рис. 3.1 установить блокировку чтения в момент начала транзакции A, то транзакции B будет разрешено чтение тех же данных, но ей не удастся заблокировать данные для записи до тех пор, пока транзакция A не завершится.

Обычно в Oracle применяются блокировки чтения, только если операция SQL специально запрашивает их в инструкции FOR UPDATE команды SELECT.

Конфликты при блокировках

Блокировки, применяемые для изоляции конкурентных пользователей данных, могут в свою очередь создавать проблемы. Как видно из примера, описанного в предыдущем разделе, одна единственная транзакция может значительно ухудшить произ-

водительность, т. к. установленные ею блокировки не дают завершиться другим транзакциям. Возникает так называемый *конфликт блокировок* (*contention*), который может отрицательно сказаться на производительности СУБД.

Вопросы целостности

Несмотря на то что применение блокировок может привести к их конфликтам и ухудшению производительности, они все же необходимы для того, чтобы избежать перечисленных ниже проблем с нарушением целостности данных:

«Грязное» чтение (Dirty reads)

Считывание данных называется «грязным», если СУБД разрешает одной транзакции читать данные, измененные другой транзакцией, когда эти изменения еще не были зафиксированы. Изменения, сделанные второй транзакцией, могут быть отменены, и тогда считанные данные окажутся некорректными. Многие СУБД разрешают «грязное» чтение для того, чтобы избежать конфликтов, создаваемых блокировками чтения.

Невоспроизводимое чтение (Nonrepeatable reads)

Речь идет о ситуации, когда данные изменяются другой транзакцией. Одна транзакция выполняет запрос по какому-то определенному условию. После того как данные отправлены в транзакцию, но до того, как она зафиксирована, другая транзакция изменяет данные так, что некоторые из извлеченных ранее данных перестают удовлетворять условию выбора. Если бы запрос в первой транзакции был повторен еще раз, он вернул бы другое результирующее множество, поэтому любые изменения, выполненные на основе исходных результатов, могут уже не быть корректными. При повторном чтении уже прочитанных ранее данных в рамках одной и той же транзакции могут быть получены отличающиеся результаты.

Фантомное чтение (Phantom reads)

Такой вид чтения также вызван изменениями, выполненными другой транзакцией. Одна транзакция выполняет запрос по какому-то определенному условию. После того как данные отправлены в транзакцию, но до того, как она зафиксирована, другая транзакция вставляет в базу данных новые строки, которые удовлетворяют условию и могли бы быть выбраны первой транзакцией. Если транзакция выполняет изменения, считая, что условию удовлетворяют только строки, возвращенные запросом, то фантомное чтение может привести к получению ошибочных данных. Все данные, которые были прочитаны первым запросом, будут возвращены и вторым, но могут быть возвращены и дополнительные данные, поэтому может оказаться, что любые изменения, сделанные на основе исходных результатов, больше не являются корректными.

Сериализация

Полное решение задачи конкурентного доступа заключается в обеспечении наивысшего уровня изоляции для действий различных пользователей, обращающихся к одним и тем же данным. Согласно стандарту SQL92 высший уровень изоляции называется *сериализацией* (*serialization*). Как видно из названия, сериализованные транзакции для пользователя выглядят так, будто они выполняются в последовательной серии отдельных упорядоченных транзакций. Когда одна транзакция начинается, она изолируется от всех изменений, которые вносят в ее данные последующие транзакции.

Для пользователя все выглядит так, как если бы в течение транзакции он работал с базой данных в монопольном режиме. Сериализуемые транзакции характеризуются

предсказуемостью и воспроизводимостью, а это две главных добродетели целостности данных.

Конечно, сделать так, чтобы сервер базы данных поддерживал работу тысяч пользователей и при этом каждый их них считал себя единственным, – нелегкая задача. Но модель MVRC Oracle ее решает.

Oracle и конкурентный доступ

Модель конкурентного доступа Oracle, или *многоверсионная модель согласованности по чтению* (MVRC) гарантирует, что пользователь всегда видит непротиворечивое представление запрошенных данных. Если другой пользователь изменяет запрошенные данные в процессе выполнения запроса, Oracle хранит версию данных в том виде, в каком они находились на момент начала выполнения запроса. Если на момент начала запроса какие-то транзакции уже начали выполняться, но еще не были зафиксированы, сервер Oracle обеспечивает недоступность для запроса изменений, внесенных такими транзакциями. Возвращенные в запрос данные будут отражать результаты выполнения всех транзакций, которые были зафиксированы на момент начала выполнения запроса.

Эта особенность MVRC влияет на производительность СУБД и на то, как запросы обращаются с базой данных:

- Oracle не устанавливает на данные какие бы то ни было блокировки для операций чтения. Другими словами, операция чтения никогда не будет препятствовать операции записи. Даже если СУБД помещает одну-единственную блокировку всего на одну строку для операции чтения, это уже может привести к конфликтным ситуациям; к тому же для большей части таблиц БД операции обновления обычно производятся для небольшого количества «горячих точек» активных данных.
- Операция записи никогда не будет препятствовать выполнению операции чтения: MVRC просто предоставит версию данных, которая существовала до начала выполнения операции записи.
- Пользователь получает полный моментальный снимок данных точно на момент начала выполнения запроса. Строка, извлекаемая в конце результирующего множества, могла быть изменена за время получения всего результирующего множества. Но поскольку сервер Oracle хранит версию строки в том виде, в котором она существовала в момент начала запроса, вы всегда получаете непротиворечивую картину данных на конкретный момент времени.
- Операция записи блокирует другую операцию записи, только если вторая пытается записать ту же самую строку.

Особенности модели MVRC

Для реализации многоверсионной согласованности по чтению в сервере Oracle применяются три структуры данных:

Сегменты отката

Сегменты отката (*rollback segments*) – это структуры СУБД Oracle, служащие для хранения информации, необходимой для возврата данных в исходное состояние в случае отката транзакции. Такая информация нужна для восстановления строк БД в состояние, в котором они находились в момент начала выполнения рассматриваемой транзакции. Начиная изменять данные в блоке, транзакция

сначала записывает старый образ данных в сегмент отката. Хранящаяся в сегменте отката информация в основном используется в двух целях: для предоставления необходимых для отката транзакции сведений и для поддержки многоверсионной согласованности.

Сегмент отката – это не то же самое, что журнал. Журнал предназначен для регистрации всех транзакций базы данных и для ее восстановления в случае сбоя системы, тогда как сегмент отката служит для обеспечения отката транзакций и согласованности чтения.

Блоки сегментов отката кэшируются в SGA вместе с блоками таблиц и индексов. Если блоки сегментов отката долгое время не используются, то самые старые из них могут быть удалены из кэша и записаны на диск.

Системные номера изменений (SCN)

Для того чтобы обеспечить целостность данных в базе данных необходимо отслеживать порядок выполнения транзакций. Механизм, применяемый сервером Oracle для поддержания упорядоченности транзакций во времени, называется системным номером изменения (System Change Number – SCN).

SCN – это логическая временная метка, предназначенная для отслеживания порядка транзакций. Oracle считывает метки SCN из журнала, когда требуется воспроизвести транзакции в их корректном первоначальном порядке. На основе меток SCN Oracle также определяет, когда можно удалять уже ненужную информацию из сегментов отката (как будет показано в последующих разделах).

Блокировки в блоках данных

СУБД должна каким-то способом узнавать о том, что некоторая строка заблокирована. Большая часть СУБД хранит в памяти список блокировок, управляемых процессом Lock Manager. Oracle хранит блокировки с помощью индикатора в том блоке данных, в котором хранится строка. Для СУБД Oracle блок данных – это наименьший объем данных, который может быть прочитан с диска, поэтому при каждом запросе строки считывается блок, а внутри него возможна блокировка. Индикаторы блокировок хранятся в блоке данных, но при этом каждая блокировка действует только для отдельной строки из блока.

В дополнение к уже упомянутым структурам данных, которые непосредственно относятся к обеспечению многоверсионной согласованности чтения, Oracle характеризуется еще одной особенностью, позволяющей достичь большей степени параллелизма в условиях большого количества пользователей:

Нерасширяемые блокировки строк (nonescalating locks)

Для того чтобы снизить накладные расходы процесса управления блокировками, некоторые СУБД (не Oracle) иногда распространяют блокировки на большее количество данных. Например, если какая-то определенная часть строк таблицы заблокирована, СУБД расширяет блокировку до целой таблицы, блокируя уже все строки таблицы, в том числе и те, которые не затрагиваются выполняемой командой SQL. Этот механизм уменьшает количество блокировок, которыми управляет процесс Lock Manager, но приводит к блокировке не затрагиваемых операцией строк. Но поскольку сервер Oracle хранит блокировку каждой строки в ее блоке данных, у него нет необходимости в расширении блокировок – и Oracle никогда этого не делает.

Синтаксис MVRC

По большей части механизм MVRC работает «за кулисами», так что относящихся к нему синтаксических конструкций очень немного. Однако пользователям следует знать, какие существуют уровни изоляции, и уметь их устанавливать.

Уровень изоляции (*isolation level*) определяет, в какой мере операции одного пользователя изолированы от операций другого пользователя. В Oracle применяются два основных уровня изоляции. Эти уровни изоляции оказывают различное влияние на разработчиков и на пользователей.

READ COMMITTED

> Этот уровень вводит сериализацию на уровне команды. Иначе говоря, любой запрос получает непротиворечивое представление данных в том состоянии, в котором они существовали на момент начала запроса. Но транзакция может включать в себя несколько команд, поэтому в рамках выполняемой транзакции может произойти невоспроизводимое или фантомное чтение. Данный уровень изоляции транзакций устанавливается Oracle по умолчанию.

SERIALIZABLE

> Этот уровень вводит сериализацию на уровне транзакций. То есть любой запрос внутри транзакции получает непротиворечивое представление данных в том виде, в каком они существовали на момент начала транзакции.

Из-за различий два уровня изоляции по-разному реагируют на транзакцию, которая при помощи блокировки запрошенной строки приостанавливает их работу. Оба уровня изоляции ждут снятия блокировки. Когда блокировка снята, операция, выполняемая с уровнем изоляции READ COMMITTED, просто повторяет попытку выполнения. Это вполне логичный подход для операций, которым важно только состояние данных на момент начала выполнения команды.

С другой стороны, если блокирующая транзакция фиксирует изменения данных, то операция, выполняющаяся с уровнем изоляции SERIALIZABLE, возвращает ошибку, указывающую на то, что сериализовать операции невозможно. Это также разумно, ведь после изменения блокирующей транзакцией состояния данных по сравнению с тем, каким оно было на момент начала транзакции SERIALIZABLE, операции записи для измененных строк становятся невозможны. В подобной ситуации программист добавляет в программу дополнительную логику, чтобы вернуться к началу транзакции SERIALIZABLE и начать ее заново.

Oracle поддерживает еще один дополнительный уровень изоляции:

READ ONLY

> Пользователь может объявить для транзакции или сеанса уровень изоляции READ ONLY. Как видно из названия, такой уровень явно запрещает операции записи. Как и уровень сериализации транзакций, READ ONLY обеспечивает точное представление данных на момент начала транзакции.

Уровни изоляции можно устанавливать при помощи команд SQL. Приведем только ту часть конструкции, которая связана с согласованностью данных. Полное описание синтаксиса указанных команд приведено в главе 7 (конкретное местоположение описаний команд внутри главы можно определить по алфавитному указателю).

ALTER SESSION

```
ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE | READ COMMITTED
```

Устанавливает уровень изоляции для сеанса.

Ключевые слова

SERIALIZABLE

Вводит сериализацию на уровне транзакции.

READ COMMITTED

Вводит сериализацию на уровне команды. Это значение по умолчанию для Oracle.

SET TRANSACTION

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE | READ COMMITTED
```

Устанавливает уровень изоляции для транзакции.

Ключевые слова

SERIALIZABLE

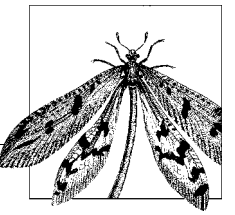
Вводит сериализацию на уровне транзакции для данной транзакции.

READ COMMITTED

Вводит сериализацию на уровне команды для данной транзакции. Это значение по умолчанию для Oracle.

4

Безопасность



Oracle предлагает большой набор возможностей, позволяющих исключить несанкционированный доступ к базе данных и помогающих защитить данные от просмотра и изменения неавторизованными пользователями. Эта глава посвящена основным концепциям Oracle, связанным с безопасностью, таким как аутентификация, профили, привилегии, роли и аудит и описывает синтаксис команд управления безопасностью базы данных.

Кроме того, здесь кратко обсуждаются и некоторые из более развитых возможностей обеспечения безопасности в Oracle. Подробное описание этих специализированных и/или поставляемых отдельно компонентов выходит за рамки данной книги. Если ваша организация приобрела компоненты Advanced Security или Label Security, информацию о них можно почерпнуть из документации Oracle.

Аутентификация

Аутентификацией называется процесс распознавания авторизованных пользователей. По существу, процедура аутентификации позволяет системе убедиться, что пользователь действительно является тем, за кого себя выдает. В основе безопасности Oracle лежит концепция индивидуальной авторизации пользователей.

Системные пользователи

При установке Oracle всегда создаются два пользователя базы данных:

SYS

Схема SYS содержит базовые таблицы и представления словаря данных. Ни при каких обстоятельствах не следует их изменять. Пользователь SYS получает роль DBA. (Роли обсуждаются в соответствующем разделе далее в этой главе.) По умолчанию пользователю SYS назначается пароль CHANGE_ON_INSTALL.

SYSTEM

Пользователь SYSTEM нужен для создания дополнительных таблиц и представлений, содержащих административную информацию. Этот пользователь также получает роль DBA. По умолчанию ему назначается пароль MANAGER.

Для ограничения доступа к обширным возможностям этих пользователей можно при создании базы данных Oracle командой CREATE DATABASE применять инструкции USER SYS IDENTIFIED BY *пароль* и USER SYSTEM IDENTIFIED BY *пароль*.

Создание пользователей

Новые пользователи создаются командой CREATE USER. Свойства существующих пользователей можно изменить командой ALTER USER.

CREATE USER

```
CREATE USER имя_пользователя
 IDENTIFIED {BY пароль | EXTERNALLY | GLOBALLY AS 'внешнее_имя'}
 [DEFAULT TABLESPACE имя_табличного_пространства]
 [TEMPORARY TABLESPACE имя_табличного_пространства]
 [QUOTA {целое_число (K | M) | UNLIMITED} ON имя_табличного_пространства]
  [QUOTA {целое_число (K | M) | UNLIMITED} ON имя_табличного_пространства …]
 [PROFILE имя_профиля]
 [PASSWORD EXPIRE]
 [ACCOUNT LOCK | UNLOCK]
```

Создает пользователя и определяет его основные характеристики.

Ключевые слова

IDENTIFIED BY

Определяет способ аутентификации пользователя. Имеются три способа аутентификации:

PASSWORD

Пользователь идентифицируется с помощью хранимого локально пароля. Пароль должен состоять из однобайтовых символов, принадлежащих набору символов базы данных.

EXTERNALLY

Пользователь идентифицируется внешней службой, в частности, операционной системой. Если пользователь должен получать доступ согласно его учетной записи в операционной системе, его имя должно начинаться значением параметра OS_AUTHENT_PREFIX.

GLOBALLY AS 'внешнее_имя'

Пользователь идентифицируется службой каталогов предприятия. Значение *внешнее_имя* может содержать характерное имя (Distinguished Name), имеющееся в каталоге или пустую строку, означающую, что каталог отобразит пользователя на соответствующую схему базы данных.

DEFAULT TABLESPACE

Указывает табличное пространство, в которое по умолчанию помещаются объекты, создаваемые пользователем. По умолчанию это табличное пространство SYSTEM.

TEMPORARY TABLESPACE

Указывает табличное пространство, отведенное для хранения временных сегментов пользователя. По умолчанию это табличное пространство SYSTEM.

QUOTA

Указывает размер области, доступной пользователю в указанном табличном пространстве. Можно указывать несколько инструкций QUOTA для нескольких табличных пространств. Допускается указание в килобайтах (K) или в мегабайтах (M). Значение UNLIMITED снимает ограничения по расходованию пространства пользователем.

PROFILE

Указывает профиль, назначенный пользователю. Подробности изложены ниже в разделе «Профили».

PASSWORD EXPIRE

Указывает на то, что пользователю или администратору БД необходимо изменить пароль, прежде чем пользователь сможет получить доступ к базе данных.

ACCOUNT LOCK | UNLOCK

Разрешает и запрещает доступ для данной учетной записи.

ALTER USER

```
ALTER USER имя_пользователя
 [IDENTIFIED {BY пароль [REPLACE старый_пароль]
  EXTERNALLY | GLOBALLY AS 'внешнее_имя'}]
 [DEFAULT TABLESPACE имя_табличного_пространства]
 [TEMPORARY TABLESPACE имя_табличного_пространства]
 [QUOTA {целое_число [K | M] | UNLIMITED} ON имя_табличного_пространства]
  [QUOTA {целое_число [K | M] | UNLIMITED} ON имя_табличного_пространства]
 [PROFILE имя_профиля]
 [DEFAULT ROLE {[имя_роли[,имя_роли . . .] |
  ALL {EXCEPT [имя_роли[,имя_роли . . .]} | NONE
 [PASSWORD EXPIRE]
 [ACCOUNT LOCK | UNLOCK]
 [имя_пользователя [,имя_пользователя . . .] инструкция_прокси-сервера]
```

Изменяет характеристики пользователя.

Ключевые слова

Большинство ключевых слов в команде ALTER USER имеют то же значение, что и в команде CREATE USER. Следующие ключевые слова применяются только в команде ALTER USER:

REPLACE старый_пароль

Если включена функция проверки сложности пароля, то при изменении пароля командой ALTER USER необходимо указывать его старое значение.

DEFAULT ROLE

Механизм ролей позволяет управлять группами привилегий для групп пользователей. Можно выдать пользователю несколько ролей, все роли (ALL), все кроме перечисленных (ALL EXCEPT) или не выдать ни одной. Более подробно роли описаны ниже в соответствующем разделе.

инструкция_прокси-сервера

Инструкция может относиться к нескольким именам пользователей. Такая возможность появилась в Oracle8*i*, где пользователь может быть идентифицирован прокси-сервером, который и передает пароль серверу БД для повторной аутентификации. В Oracle9*i* информация о личности пользователя в виде характерного имени (Distinguished Name) или полного сертификата X.509 может быть передана серверу БД для идентификации без повторной аутентификации.

Инструкция прокси-сервера имеет такой синтаксис:

```
{GRANT | REVOKE} CONNECT THROUGH прокси-сервер
 [WITH {ROLE [имя_роли[,имя_роли. . .]] |
  ALL [EXCEPT] [имя_роли [,имя_роли. . .]] |
```

```
  NO ROLES}]
 AUTHENTICATED USING {PASSWORD | DISTINGUISHED NAME |
  CERTIFICATE [TYPE имя_типа][VERSION 'имя_версии']}
```

GRANT | REVOKE

Разрешает или запрещает соединение через прокси.

CONNECT THROUGH прокси-сервер

Указывает прокси-сервер, через который устанавливается соединение с Oracle. Подробная информация о прокси-серверах содержится в разделе «Другие возможности обеспечения безопасности» в конце главы.

WITH ROLE

Назначает роль прокси-пользователю. Синтаксис аналогичен синтаксису ключевого слова DEFAULT ROLE.

AUTHENTICATED USING

Указывает, будет ли аутентификация прокси-сервера производиться источником, отличным от прокси-сервера. DISTINGUISHED NAME и CERTIFICATE указывают, что прокси-сервер действует от имени глобального пользователя базы данных.

Профили

Для того чтобы ограничить доступ пользователя к ресурсам или указать условие обработки паролей пользователя, можно сопоставить ему *профиль* (*profile*). Ограничив объем вычислительных ресурсов, доступных пользователю, вы предотвратите их перерасход и нанесение ущерба работе других пользователей. (Печально известен так называемый *отказ в обслуживании* (*denial of service*)). Налагая ограничения на администрирование паролей, вы способствуете защите процесса аутентификации в вашей БД Oracle.

Для того чтобы использовать профили, следует разрешить динамические ограничения ресурсов при помощи параметра инициализации RESOURCE_LIMIT или же команды ALTER SYSTEM SET. Определив профиль командой CREATE PROFILE, вы можете назначить его пользователю посредством команды CREATE USER или ALTER USER.

CREATE PROFILE

```
CREATE PROFILE имя_профиля LIMIT {параметр_ресурса | параметр_пароля}
```

Позволяет создать профиль и назначить для этого профиля различные типы ограничений на расходование ресурсов. Для роли может быть указано несколько параметров.

Ключевые слова для всех параметров

Приведенные ниже значения могут быть заданы как в параметрах ресурсов, так и в параметрах пароля, если они не были ранее заданы в описаниях параметров.

UNLIMITED

Указывает, что для данного параметра нет ограничений.

DEFAULT

Указывает, что параметр принимает значение, заданное для профиля DEFAULT. Изначально все значения профиля DEFAULT устанавливаются в UNLIMITED.

Значения для профиля DEFAULT можно изменить при помощи команды ALTER PROFILE.

параметр_ресурса | *параметр_пароля*

Значения параметров ресурсов задаются целыми числами. Значение параметра пароля задается выражением.

Ключевые слова для параметров ресурсов

За исключением специально описанных случаев, если пользователь пытается выполнить операцию, выходящую за пределы установленных ограничений, сервер Oracle прерывает операцию, откатывает текущую команду и оставляет транзакцию нетронутой.

SESSIONS_PER_USER

Ограничивает количество одновременных сеансов пользователя.

CPU_PER_SESSION

Ограничивает время процессора для пользовательского сеанса (в сотых долях секунды).

CPU_PER_CALL

Ограничивает время процессора для отдельного пользовательского вызова (в сотых долях секунды).

CONNECT_TIME

Ограничивает общую фактическую продолжительность сеанса (в минутах). Если пользователь превышает значение этого параметра, сервер Oracle откатывает текущую транзакцию и завершает сеанс. Следующий вызов, сделанный пользователем, возвращает ошибку.

IDLE_TIME

Ограничивает время непрерывного ожидания пользователя (в минутах). Ограничение времени ожидания не применяется для длительных запросов и других подобных операций. Если пользователь превышает значение этого параметра, сервер Oracle откатывает текущую транзакцию и завершает сеанс. Следующий вызов, сделанный пользователем, возвращает ошибку.

LOGICAL_READS_PER_SESSION

Ограничивает количество логических блоков данных, считываемых в пользовательском сеансе как из памяти, так и с диска.

LOGICAL_READS_PER_CALL

Ограничивает количество логических блоков данных, считываемых в каждом пользовательском вызове как из памяти, так и с диска.

COMPOSITE_LIMIT

Ограничивает общую *стоимость ресурсов* (*resource cost*) сеанса (в сервисных единицах). Oracle вычисляет сервисные единицы как взвешенную сумму следующих параметров инициализации: CPU_PER_SESSION, CONNECT_TIME, LOGICAL_READS_PER_SESSION и PRIVATE_SGA. Вес каждого из ресурсов можно изменить при помощи команды ALTER RESOURCE COST.

PRIVATE_SGA (integer (K | M)| UNLIMITED | DEFAULT)

Ограничивает объем закрытого пространства, которое пользовательский сеанс может выделить из разделяемого пула SGA (в килобайтах (K) или мегабайтах (M)).

Ключевые слова для параметров пароля

FAILED_LOGIN_ATTEMPTS

Ограничивает количество неудачных попыток регистрации пользователя, предшествующих блокировке его учетной записи.

PASSWORD_LIFE_TIME

Устанавливает ограничение максимального срока действия одного пароля (в днях) для пользователя. По истечении этого срока пароль теряет силу.

PASSWORD_REUSE_TIME

Указывает, сколько дней должно пройти, прежде чем пароль можно будет задать повторно. Если для параметра задано целочисленное значение, то необходимо установить PASSWORD_REUSE_MAX в UNLIMITED.

PASSWORD_REUSE_MAX

Указывает, сколько раз необходимо изменить пароль, прежде чем будет разрешено задать его повторно. Если для параметра задано целочисленное значение, то необходимо установить PASSWORD_REUSE_TIME в UNLIMITED.

PASSWORD_LOCK_TIME

Указывает количество дней, на которое будет заблокирована учетная запись пользователя в случае превышения количества попыток неудачной регистрации.

PASSWORD_GRACE_TIME

Указывает, за какое количество дней до истечения срока действия пароля выдается предупреждение.

PASSWORD_VERIFY_FUNCTION функция | NULL | DEFAULT

Разрешает применение функции PL/SQL для проверки сложности пароля.

ALTER PROFILE

```
ALTER PROFILE имя_профиля LIMIT {параметр_ресурса | параметр_пароля}
```

Изменяет ограничения на расходование ресурсов или ограничения пароля для существующего профиля.

Ключевые слова

В команде ALTER PROFILE применяются те же ключевые слова и значения, что и в CREATE PROFILE.

DROP PROFILE

```
DROP PROFILE имя_профиля (CASCADE)
```

Удаляет существующий профиль.

Ключевые слова

CASCADE

Указывает, что назначение профиля для всех активных пользователей должно быть отменено, а их профили необходимо заменить на DEFAULT. Данную инструкцию следует применять при удалении профиля текущего активного пользователя.

Привилегии

Привилегии – это права, назначаемые отдельным пользователям или ролям. Привилегии делятся на два основных вида:

Системные привилегии

Дают пользователю или роли возможность выполнять определенные системные операции.

Объектные привилегии

Дают пользователю или роли права доступа к отдельным объектам схемы.

Системные привилегии относятся к экземпляру Oracle в целом, например есть привилегия для всех объектов одного типа (скажем, для всех таблиц). Объектные же привилегии связаны с конкретным объектом схемы внутри базы данных Oracle (например, с конкретной таблицей).

Системные привилегии

В этом разделе изложена общая информация обо всех системных привилегиях Oracle. Некоторые разновидности системных привилегий могут применяться к различным типам привилегий:

ANY

Дает привилегию на выполнение операции над объектами любой схемы. В отсутствие такого ключевого слова привилегия выдается только на объекты в рамках схемы пользователя. По умолчанию ключевое слово ANY дает пользователю привилегию на все объекты всех схем, включая SYS. Для того чтобы запретить привилегии ANY на доступ к схеме SYS, можно установить параметр инициализации O7_DICTIONARY_ACCESSIBILITY в FALSE.

ALTER

Дает привилегию на изменение объекта некоторого типа.

CREATE

Дает привилегию на создание объекта данного типа.

DROP

Дает привилегию на удаление объекта данного типа.

EXECUTE

Дает привилегию на исполнение объекта данного типа или обращение к нему.

SELECT ANY

Дает привилегию на доступ к объектам. Пользователи всегда имеют возможность доступа к объектам своей собственной схемы, поэтому данная разновидность привилегий применяется только с ключевым словом ANY.

Каждая из представленных разновидностей привилегий может применяться со многими типами системных привилегий, описанных далее.

В описании каждой привилегии указаны привилегии двух категорий: общие (перечисленные в предыдущем разделе, применяемые для различных типов привилегий) и уникальные (присущие только данному конкретному типу привилегий).

AUDIT

Разрешает функции аудита.

Уникальные привилегии

AUDIT SYSTEM Дает привилегию на выдачу команд AUDIT в SQL.

Общие привилегии ANY.

CLUSTER

Предоставляет возможность работы с кластерами.

Уникальные привилегии Нет.

Общие привилегии CREATE [ANY], ALTER ANY, DROP ANY.

CONTEXT

Предоставляет возможность работы с контекстами. Появилась в Oracle8*i*.

Уникальные привилегии Нет.

Общие привилегии CREATE ANY, DROP ANY.

DATABASE

Предоставляет возможность выполнения команды ALTER DATABASE.

Уникальные привилегии Нет.

Общие привилегии ALTER.

DATABASE LINKS

Предоставляет возможность работы со связями БД (*database links*).

Уникальные привилегии

CREATE PUBLIC Дает привилегию на создание открытых связей БД.

DROP PUBLIC Дает привилегию на удаление открытых связей БД.

Общие привилегии CREATE.

DEBUG

Предоставляет возможность работы с отладчиком. Появилась в Oracle9*i*.

Уникальные привилегии

DEBUG CONNECT SESSION

Дает привилегию на подключение текущего сеанса к отладчику, который использует протокол Java Debug Wire Protocol.

DEBUG ANY PROCEDURE

Дает привилегию на отладку любого PL/SQL- и Java-кода для любого объекта БД, а также отображение всех команд SQL, выполняемых приложением.

Общие привилегии Нет.

DIMENSION

Предоставляет возможность работы с измерениями. Появилась в Oracle8*i*.

Уникальные привилегии Нет.

Общие привилегии CREATE [ANY], ALTER ANY, DROP ANY.

DIRECTORY

Предоставляет возможность работы с каталогами.

Уникальные привилегии Нет.

Общие привилегии CREATE ANY, DROP ANY.

INDEX

Предоставляет возможность работы с индексами.

Уникальные привилегии Обратитесь к информации о привилегии [GLOBAL] QUERY REWRITE в разделе «Различные привилегии» далее в этой главе.

Общие привилегии CREATE ANY, ALTER ANY, DROP ANY.

INDEXTYPE

Предоставляет возможность работы с объектами типа INDEXTYPE, созданными пользователем. Появилась в Oracle8*i*.

Уникальные привилегии Нет.

Общие привилегии CREATE [ANY], ALTER ANY (появилась в Oracle9*i*), DROP ANY, EXECUTE ANY.

LIBRARY

Предоставляет возможность работы с библиотеками.

Уникальные привилегии Нет.

Общие привилегии CREATE [ANY], DROP ANY.

MATERIALIZED VIEW

Предоставляет возможность работы с *материализованными представлениями* (*materialized views*). Привилегии MATERIALIZED VIEW в версиях, предшествующих Oracle8*i*, назывались SNAPSHOT.

Уникальные привилегии

ON COMMIT REFRESH

Дает привилегию на создание материализованного представления с обновлением при выполнении COMMIT. Появилась в Oracle8*i*.

См. также информацию о [GLOBAL] QUERY REWRITE и FLASHBACK ANY TABLE в разделе «Различные привилегии».

Общие привилегии CREATE [ANY], ALTER ANY, DROP ANY.

OPERATOR

Предоставляет возможность работы с определяемыми пользователем операторами. Появилась в Oracle8*i*.

Уникальные привилегии Нет.

Общие привилегии CREATE [ANY], DROP, EXECUTE.

OUTLINE

Предоставляет возможность работы с хранимыми планами выполнения (*stored outlines*). Появилась в Oracle8*i*.

Уникальные привилегии

SELECT ANY

Несмотря на общее ключевое слово для объектов, в данном случае речь идет о предоставлении привилегии на создание закрытого плана выполнения из открытого. Появилась в Oracle9*i*.

Общие привилегии CREATE ANY, ALTER ANY, DROP ANY.

PROCEDURE

Предоставляет возможность работы с процедурами.

Уникальные привилегии Нет.

Общие привилегии CREATE [ANY], ALTER ANY, DROP ANY, EXECUTE ANY.

PROFILE

Предоставляет возможность работы с профилями.

Уникальные привилегии Нет.

Общие привилегии CREATE, ALTER, DROP.

RESOURCE COST

Предоставляет возможность присваивания стоимостей ресурсам.

Уникальные привилегии Нет.

Общие привилегии ALTER.

ROLE

Предоставляет возможность работы с ролями.

Уникальные привилегии

GRANT ANY Дает привилегию на предоставление любых ролей в БД.

Общие привилегии CREATE, ALTER ANY, DROP ANY.

ROLLBACK SEGMENT

Предоставляет возможность работы с сегментами отката.

Уникальные привилегии Нет.

Общие привилегии CREATE, ALTER, DROP.

SEQUENCE

Предоставляет возможность работы с последовательностями.

Уникальные привилегии Нет.

Общие привилегии CREATE [ANY], ALTER ANY, DROP ANY, SELECT ANY.

SESSION

Предоставляет возможность работы с сеансами.

Уникальные привилегии

ALTER RESOURCE COST

Дает привилегию на задание стоимостей ресурсов сеанса.

RESTRICTED SESSION

Дает привилегию на вход в систему после того, как экземпляр Oracle запущен командой STARTUP RESTRICT.

Общие привилегии CREATE, ALTER.

SNAPSHOT

В версии Oracle9*i* ключевое слово *SNAPSHOT заменено* на MATERIALIZED VIEW. В версии Oracle8*i* ключевые слова SNAPSHOT и MATERIALIZED VIEW были взаимозаменяемыми.

SYNONYM

Предоставляет возможность работы с синонимами.

Уникальные привилегии Нет.

Общие привилегии CREATE [ANY] [PUBLIC], DROP ANY, DROP PUBLIC.

SYSTEM

Предоставляет возможность изменения параметров системы.

Уникальные привилегии Нет.

Общие привилегии ALTER.

TABLE

Предоставляет возможность работы с таблицами.

Уникальные привилегии

BACKUP ANY

Дает привилегию на использование утилиты Export для объектов в схемах других пользователей.

COMMENT ANY

Дает привилегию на комментирование любых таблиц, столбцов и представлений в любой схеме.

INSERT ANY

Дает привилегию на вставку строк в таблицы в схемах других пользователей.

LOCK ANY

Дает привилегию на блокировку таблиц и представлений в схемах других пользователей.

FLASHBACK ANY

Дает возможность выдавать ретроспективные запросы SQL (*flashback query*) для любой таблицы или материализованного представления в схемах других пользователей. Вы можете, как и ранее, использовать встроенные процедуры пакета DBMS_FLASHBACK, не имея такой привилегии. Появилась в Oracle9*i*.

UPDATE ANY

Дает привилегию на обновление строк таблиц и представлений в схемах других пользователей.

Дополнительная информация по FLASHBACK ANY TABLE приведена в разделе «Различные привилегии».

Общие привилегии

CREATE [ANY] (CREATE поддерживается только в Oracle8 и более ранних версиях), ALTER ANY, DELETE ANY, DROP ANY, SELECT ANY.

TABLESPACES

Предоставляет возможность работы с табличными пространствами

Уникальные привилегии

MANAGE

Дает привилегию на перевод табличных пространств в автономный и оперативный режимы, а также на запуск и завершение резервного копирования табличных пространств.

UNLIMITED TABLESPACE

Дает привилегию на перекрытие любых назначенных квот для конкретных табличных пространств. Если привилегия отзывается, пользователь может выделить дополнительное табличное пространство только в рамках существующих квот. Привилегия не может быть выдана роли (то есть назначается только пользователям).

Общие привилегии CREATE, ALTER, DROP.

TRIGGER

Предоставляет возможность работы с триггерами.

Уникальные привилегии

ADMINISTER DATABASE

Дает привилегию на создание триггера для базы данных. Для получения данной привилегии пользователь или роль должен обладать привилегией CREATE [ANY] TRIGGER. Появилась в Oracle8*i*.

Общие привилегии CREATE [ANY], ALTER ANY, DROP.

TYPES

Предоставляет возможность работы с определяемыми пользователем типами.

Уникальные привилегии

UNDER ANY

Дает привилегию на создание подтипов для всех типов, которые не определены как терминальные (*final*). Появилась в Oracle9*i*.

Общие привилегии CREATE [ANY], ALTER ANY, DROP ANY, EXECUTE ANY.

USER

Предоставляет возможность работы с пользователями базы данных.

Уникальные привилегии

BECOME

Дает возможность стать другим пользователем, что необходимо для выполнения полного импорта базы данных.

Общие привилегии CREATE, ALTER, DROP.

VIEWS

Предоставляет возможность работы с представлениями.

Уникальные привилегии

UNDER ANY

Дает привилегию на создание дочерних представлений для любого объекта представления. Появилась в Oracle9*i*.

Обратитесь также к информации о FLASHBACK ANY TABLE в разделе «Различные привилегии».

Общие привилегии CREATE [ANY], DROP.

Различные привилегии

В этом разделе собраны привилегии, не попадающие ни в одну из вышеуказанных категорий.

ANALYZE ANY

Предоставляет привилегию на проведение анализа любой таблицы, кластера или индекса в любой схеме.

EXEMPT ANY

Предоставляет привилегию на игнорирование политики безопасности, проводимой приложением. Появилась в Oracle9*i*.

FLASHBACK ANY TABLE

Предоставляет привилегию на выдачу ретроспективного запроса SQL (*flashback query*) для любой таблицы, представления или материализованного представления в любой схеме. Вы можете, как и ранее, использовать встроенные процедуры пакета DBMS_FLASHBACK, не имея такой привилегии. Появилась в Oracle9*i*.

FORCE TRANSACTION

Предоставляет привилегию на принудительную фиксацию или откат любой из распределенных транзакций пользователя в локальной базе данных.

FORCE ANY TRANSACTION

Предоставляет привилегию на принудительную фиксацию или откат любой распределенной транзакции в локальной базе данных или на вызов сбоя распределенной транзакции.

GRANT ANY PRIVILEGE

Предоставляет привилегию на выдачу любых системных привилегий.

GRANT ANY OBJECT PRIVILEGE

Предоставляет привилегию на выдачу любых привилегий доступа к объектам. Появилась в Oracle9*i*.

[GLOBAL] QUERY REWRITE

Предоставляет привилегию разрешения перезаписи запроса с использованием материализованного представления или создания индекса на основе функции. Ключевое слово GLOBAL действует как ANY. Появилась в Oracle8*i*.

RESUMABLE

Предоставляет привилегию разрешения возобновляемого выделения пространства. Появилась в Oracle9*i*.

SELECT ANY DICTIONARY

Предоставляет привилегию на запрос любого объекта словаря данных в схеме SYS, что дает пользователю избирательную возможность перекрытия значения параметра инициализации O7_DICTIONARY_ACCESSIBILITY. Появилась в Oracle9*i*.

Особые системные привилегии

Описанные в разделе системные привилегии предназначены для предоставления пользователю возможности выполнения всего набора операций. Это особые привилегии, поскольку одна привилегия предоставляет пользователю набор базовых полномочий.

SYSDBA

Дает пользователю все права, необходимые для запуска и остановки базы данных Oracle. Включает в себя привилегию RESTRICTED SESSION, а также следующие полномочия:

ALTER DATABASE

CREATE DATABASE

ARCHIVELOG и RECOVERY

CREATE SPFILE (появилась в Oracle9*i*)

SYSOPER

Предоставляет пользователю чуть более ограниченный набор прав, предназначенный для оператора системы. Включает в себя привилегию RESTRICTED SESSION, а также следующие полномочия:

ALTER DATABASE OPEN | MOUNT | BACKUP

ARCHIVELOG и RECOVERY

CREATE SPFILE (появилась в Oracle9*i*)

Привилегии доступа к объектам схемы

Существует несколько разновидностей привилегий доступа к объектам схемы. Они могут применяться к различным типам объектов схемы, как описано в последующих разделах.

Разновидности привилегий доступа к объектам схемы

ALTER

Изменяет определение объекта.

DEBUG

Обращается к PL/SQL-коду или информации о командах SQL, которые обращаются к объекту напрямую через отладчик. Появилась в Oracle9*i*.

DELETE

Удаляет строки из объекта.

EXECUTE

Компилирует или исполняет процедуру или функцию объекта, или же обращается к программному объекту, объявленному в объекте.

FLASHBACK

Выполняет ретроспективный запрос к объекту. Появилась в Oracle9*i*.

INSERT

Добавляет в объект новые строки.

REFERENCES

Создает *ограничение* (*constraint*), ссылающееся на объект.

SELECT

Запрашивает объект.

UNDER

Создает дочерний объект ниже уровня объекта. Появилась в Oracle9*i*.

UPDATE

Изменяет существующие данные объекта.

Объекты схемы и их привилегии

В разделе приведены все типы объектов схемы, при этом для каждого из них указаны общие и уникальные разновидности привилегий.

Каталоги

Предоставляет привилегии на выполнение операций над каталогами.

Уникальные привилегии

READ

Читать файлы каталога.

WRITE

Писать в файлы каталога, за исключением *BFILE*. Применяется к внешним таблицам каталога. Появилась в Oracle9*i*.

Общие привилегии Нет.

Внешние таблицы

Предоставляет привилегии на выполнение операций над внешними таблицами.

Уникальные привилегии Нет.

Общие привилегии ALTER, SELECT.

Indextypes

Предоставляет привилегии на выполнение операций над объектами INDEXTYPE (пользовательские индексы, которые появились в Oracle8*i*).

Уникальные привилегии Нет.

Общие привилегии EXECUTE.

Библиотеки

Предоставляет привилегии на выполнение операций над библиотеками.

Уникальные привилегии Нет.

Общие привилегии EXECUTE.

Материализованные представления

Предоставляет привилегии на выполнение операций над материализованными представлениями. Материализованные представления – это предварительно агрегированные сводные данные, участвующие в операциях бизнес-интеллекта. Материализованные представления появились в Oracle8*i*. В Oracle8*i* привилегии могли предоставляться как для материализованных представлений, так и для *моментальных копий данных* (*snapshots*). В Oracle8 и более ранних версиях привилегии предоставлялись только для моментальных копий данных.

Уникальные привилегии Нет.

Общие привилегии

DELETE, FLASHBACK, INSERT, SELECT, UPDATE. Привилегии DELETE, INSERT и UPDATE могут быть предоставлены только для обновляемых материализованных представлений.

Операторы

Предоставляет привилегии на выполнение операций над операторами, т. е. пользовательскими операторами для определенных видов сравнений. Операторы появились в Oracle8*i*.

Уникальные привилегии Нет.

Общие привилегии EXECUTE.

Процедуры, функции и пакеты

Предоставляет привилегии на выполнение операций над тремя типами программных единиц: процедурами, функциями и пакетами.

Уникальные привилегии Нет.

Общие привилегии DEBUG, EXECUTE.

Последовательности

Предоставляет привилегии на выполнение операций над последовательностями.

Уникальные привилегии Нет.

Общие привилегии ALTER, SELECT.

Таблицы

Предоставляет привилегии на выполнение операций над таблицами.

Уникальные привилегии

INDEX

Создать индекс для таблицы.

ON COMMIT REFRESH

Создать материализованное представление, обновляемое при выполнении операции COMMIT. Появилась в Oracle9*i*.

QUERY REWRITE

Создать материализованное представление для перезаписи запроса в определенную таблицу.

Общие привилегии ALTER, DELETE, DEBUG, FLASHBACK, INSERT, REFERENCES, SELECT, UPDATE.

Пользовательские типы

Предоставляет привилегии на выполнение операций над пользовательскими типами. Пользовательские типы – это уникальные типы данных, создаваемые пользователем. Они появились в Oracle8*i*.

Уникальные привилегии Нет.

Общие привилегии DEBUG, EXECUTE, UNDER.

Представления

Предоставляет привилегии на выполнение операций над представлениями. Для того чтобы выдать привилегию на представление, необходимо обладать данной привилегией с указанием GRANT OPTION для всех таблиц, являющихся основой представления.

Уникальные привилегии Нет.

Общие привилегии DEBUG, DELETE, FLASHBACK, INSERT, REFERENCES, SELECT, UNDER.

Привилегии и пользователи

Для назначения привилегий пользователю или роли применяется команда GRANT. Команда REVOKE позволяет лишить пользователя или роль привилегии.

Общие ключевые слова и инструкции

В разделе собраны ключевые слова и инструкции, которые могут применяться как в команде GRANT, так и в REVOKE:

системная_привилегия

Системная привилегия (см. раздел «Системные привилегии» ранее в этой же главе).

роль

Существующая роль.

ALL PRIVILEGES

Предоставляет или отбирает все системные привилегии, за исключением SELECT ANY DICTIONARY. Для объектов предоставляет все привилегии, имеющиеся для данного объекта, ключевое слово PRIVILEGES является необязательным.

получатель

Один или несколько пользователей, одна или несколько ролей или ключевое слово PUBLIC, которое предоставляет или отбирает привилегии у всех пользователей

базы данных. Если указывается несколько получателей, их следует разделить запятыми.

объектная_привилегия

Привилегия доступа к объектам, описываемая ранее в этой главе.

имя_столбца

Один или несколько столбцов, для которых предоставляется или отбирается привилегия доступа к объектам INSERT, REFERENCES или UPDATE. Если имя столбца не указано, то привилегия *предоставляется* на все столбцы таблицы или представления.

схема.объект

Указывает имя объекта, на который выдается или отбирается привилегия. Если схема не задана, то сервер Oracle считает, что объект находится в собственной схеме пользователя.

DIRECTORY имя_каталога

Указывает имя каталога, на который выдается или отбирается привилегия.

GRANT

Для предоставления системных привилегий или ролей:

```
GRANT {системная_привилегия | роль | ALL PRIVILEGES} TO получатель
   [IDENTIFIED BY пароль] [WITH ADMIN OPTION]
```

Для предоставления привилегий доступа к объектам схемы:

```
GRANT {объектная_привилегия | ALL [PRIVILEGES]}
        [имя_столбца [,имя_столбца . . .]]
   ON {схема.объект | DIRECTORY имя_каталога |
        JAVA (SOURCE | RESOURCE) [схема.]объект}
   TO получатель [WITH GRANT OPTION] [WITH HIERARCHY OPTION]
```

Предоставляет пользователю или роли привилегии или роли. Для того чтобы предоставить привилегию, пользователь должен предварительно получить привилегию или роль с указанием WITH ADMIN OPTION (см. далее раздел «Ключевые слова»). Вы также можете предоставлять привилегии, если обладаете привилегией GRANT ANY PRIVILEGE (для системных привилегий), GRANT ANY ROLE (для ролей), GRANT OPTION (для объектов схемы) или являетесь владельцем объекта.

Системные привилегии и привилегии доступа к объектам схемы не могут быть предоставлены в одной команде GRANT.

Ключевые слова

IDENTIFIED BY пароль

Может применяться для идентификации существующего пользователя по паролю или для создания нового пользователя с указанным паролем. Появилась в Oracle9*i*.

ALL PRIVILEGES

Предоставляет пользователю или роли все привилегии, за исключением SELECT ANY DICTIONARY. Появилась в Oracle9*i*.

WITH ADMIN OPTION

Позволяет пользователю выдавать или отбирать системную привилегию или роль, а также изменять и удалять роль.

WITH GRANT OPTION

Подобно WITH ADMIN OPTION позволяет пользователю или роли выдавать или отбирать привилегию доступа к объектам у других пользователей или ролей.

JAVA SOURCE | RESOURCE

Разрешает доступ к исходным текстам Java или Java-ресурсу. Появилась в Oracle8*i*.

WITH HIERARCHY OPTION

Позволяет получателю получить привилегии на все дочерние объекты указанного объекта схемы. Появилась в Oracle9*i*.

REVOKE

Для изъятия системных привилегий или ролей:

```
REVOKE {системная_привилегия | роль | ALL PRIVILEGES} FROM получатель
```

Для изъятия привилегий доступа к объектам:

```
REVOKE {объектная_привилегия | ALL [PRIVILEGES]}
        [имя_столбца [,имя_столбца . . .]]
    ON {схема.объект | DIRECTORY имя_каталога |
        JAVA [SOURCE | RESOURCE] [схема.]объект}
    FROM получатель [CASCADE CONSTRAINTS] [FORCE]
```

Аннулирует привилегии, ранее выданные пользователю или роли. Эта команда может аннулировать только те привилегии, которые были ранее выданы командой GRANT. Если вы отзываете роль у клиента, который использует ее в текущий момент, то роль остается, но уже не будет доступна клиенту после того, как он перестанет ей пользоваться.

Если несколько обладателей привилегии выдали ее пользователю (или в случае PUBLIC), то для того чтобы привилегия стала недоступна пользователю, она должна быть аннулирована всеми выдавшими.

Ключевые слова

ALL PRIVILEGES

Аннулирует все существующие системные привилегии для пользователя или роли. Появилась в Oracle9*i*.

JAVA SOURCE | RESOURCE

Аннулирует доступ к исходным текстам Java и к Java-ресурсу. Появилась в Oracle8*i*.

CASCADE CONSTRAINTS

Применяется только при отзыве привилегии REFERENCES или привилегий доступа к объектам ALL. Удаляет все ограничения, которые пользователь, лишаемый привилегий, определил для объекта.

FORCE

Применяется при отзыве привилегии доступа к объектам EXECUTE для объектов пользовательских типов, имеющих зависимости от таблиц или типов. Приводит к тому, что все зависимые объекты помечаются как INVALID, запрещает доступ к данным зависимых таблиц, помечает все зависимые *функциональные индексы* (*function-based*) как UNUSABLE.

Роли

Предоставление отдельных привилегий отдельным пользователям может существенно усложнить работу, особенно в корпоративных системах с большим количеством пользователей. Роли предназначены для того, чтобы упростить управление привилегиями.

Привилегии можно выдавать ролям, а затем назначать роли пользователям. Ведение привилегий осуществляется на уровне ролей и затрагивает всех пользователей, которым назначены такие роли. Кроме того, в зависимости от контекста роли могут избирательно разрешаться и запрещаться для пользователей. То есть посредством ролей можно объединять наборы привилегий и предоставлять их как единое целое. Например, у вас может существовать роль ADMIN, наделяющая администратора соответствующими полномочиями.

Роль может выдаваться другой роли. Если вы назначаете пользователю родительскую роль, он по умолчанию получает и все роли, выданные родительской роли.

Пользователю можно назначить несколько ролей. Количество одновременно выдаваемых ролей ограничено параметром инициализации MAX_ENABLED_ROLES. Несколько ролей дают возможность одному пользователю в разные моменты времени применять разные наборы привилегий. Если роли выданы другие роли, то применение родительской роли подразумевает применение всех дочерних.

Команда ALTER USER позволяет задать одну или несколько ролей по умолчанию. Роли, назначенные по умолчанию, действуют, когда пользователь регистрируется в базе данных Oracle.

Системные роли

СУБД Oracle предлагает ряд предопределенных системных ролей:

CONNECT

Включает в себя системные привилегии ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE и CREATE VIEW. Согласно информации корпорации Oracle, эта роль предоставляется для обеспечения совместимости с ранними версиями Oracle и может не поддерживаться в версиях выше Oracle9*i*.

RESOURCE

Включает в себя системные привилегии CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE и CREATE TYPE. Согласно информации корпорации Oracle, эта роль предоставляется для обеспечения совместимости с ранними версиями Oracle и может не поддерживаться в версиях выше Oracle9*i*.

DBA

Включает в себя все системные привилегии с указанием WITH ADMIN OPTION. Согласно информации корпорации Oracle, эта роль предоставляется для обеспечения совместимости с ранними версиями Oracle и может не поддерживаться в версиях выше Oracle9*i*.

CREATE TYPE

Включает в себя привилегии CREATE TYPE, EXECUTE, EXECUTE ANY TYPE, ADMIN OPTION и GRANT OPTION. Роль удалена в версиях выше Oracle8.

EXP_FULL_DATABASE

Предназначена для предоставления всех привилегий, необходимых для выполнения полного и инкрементного экспорта базы данных. Включает в себя привилегии SELECT ANY TABLE, BACKUP ANY TABLE, EXECUTE ANY PROCEDURE, EXECUTE ANY TYPE, ADMINISTER RESOURCE MANAGER, а также привилегии INSERT, DELETE и UPDATE для таблиц SYS.INCVID, SYS.INCFIL и SYS.INCEXP. Кроме того, включает в себя роли EXECUTE_CATALOG_ROLE и SELECT_CATALOG_ROLE.

IMP_FULL_DATABASE

Предназначена для предоставления всех привилегий, необходимых для выполнения полного импорта базы данных. Включает в себя множество системных привилегий, а также роли EXECUTE_CATALOG_ ROLE и SELECT_CATALOG_ROLE.

DELETE_CATALOG_ROLE

Содержит в себе привилегию DELETE для таблицы аудита системы (AUD$).

EXECUTE_CATALOG_ROLE

Включает в себя привилегию EXECUTE для таблицы аудита системы (AUD$) и роль HS_ADMIN_ROLE.

SELECT_CATALOG_ROLE

Включает в себя привилегию SELECT для таблицы аудита системы (AUD$) и роль HS_ADMIN_ROLE.

RECOVERY_CATALOG_OWNER

Предназначена для предоставления всех привилегий, необходимых владельцу каталога восстановления. Включает в себя системные привилегии CREATE SESSION, ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE PROCEDURE, CREATE SEQUENCE, CREATE SYNONYM, CREATE TABLE, CREATE TRIGGER и CREATE VIEW.

HS_ADMIN_ROLE

Предназначена для защиты пакетов и представлений словарей данных гетерогенных сервисов (Heterogeneous Services – HS).

AQ_USER_ROLE

Предоставляет привилегию EXECUTE для встроенных пакетов механизма улучшенной организации очередей (Advanced Queuing): DBMS_AQ и DBMS_AQIN. Устарела в Oracle9*i*.

AQ_ADMINISTRATOR_ROLE

Предназначена для предоставления всех привилегий, необходимых для администрирования механизма улучшенной организации очередей. Включает в себя привилегии ENQUEUE ANY QUEUE, DEQUEUE ANY QUEUE, MANAGE ANY QUEUE, SELECT для таблиц механизма улучшенной организации очередей и привилегию EXECUTE для пакетов механизма улучшенной организации очередей.

SNMPAGENT

Используется Enterprise Manager Intelligent Agent и включает в себя привилегии ANALYZE ANY и SELECT для различных представлений.

Определение ролей

Описанные в последующих разделах команды позволяют создавать, изменять и удалять роли.

Команда ALTER USER позволяет определить одну или несколько ролей по умолчанию. Роли, назначенные по умолчанию, действуют, когда пользователь регистрируется в базе данных Oracle.

CREATE ROLE

```
CREATE ROLE имя_роли {NOT IDENTIFIED | IDENTIFIED
    {EXTERNALLY | GLOBALLY |
    BY пароль |
    USING [схема.]пакет}}
```

Создает роль. Когда пользователь создает роль, она автоматически выдается ему как роль по умолчанию.

Ключевые слова

NOT IDENTIFIED

Указывает, что для роли не требуется пароль.

IDENTIFIED

Определяет, каким способом пользователь будет аутентифицирован, прежде чем ему будет разрешено активировать роль или назначить ее ролью по умолчанию. Возможны следующие варианты:

BY пароль

Локальный пользователь должен предоставить пароль для активирования роли.

USING (схема.)пакет

Проверку пользователя осуществит пакет. Применяется для ролей приложений. Если имя схемы не указано, то считается, что пакет находится в собственной схеме пользователя. Появилась в Oracle9*i*.

EXTERNALLY

Внешний пользователь авторизуется сторонней службой, такой как операционная система.

GLOBALLY

Пользователь авторизуется службой каталогов предприятия.

ALTER ROLE

```
ALTER ROLE имя_роли {NOT IDENTIFIED | IDENTIFIED
     {EXTERNALLY | GLOBALLY |
     BY пароль |
     USING [схема.]пакет}}
```

Изменяет способ аутентификации пользователя роли.

Ключевые слова

Ключевые слова совпадают с ключевыми словами, приведенными для команды CREATE ROLE.

DROP ROLE

```
DROP ROLE имя_роли
```

Удаляет роль из базы данных. Если роль используется в данный момент, то ничего не произойдет, но заново использовать роль пользователь уже не сможет.

SET ROLE

```
SET ROLE {роль [IDENTIFIED BY пароль [,роль [IDENTIFIED BY PASSWORD_ ._ ._ ._ ]] |
    ALL [EXCEPT роль[, роль_ ._ ._ ._ ]]
    NONE}
```

Разрешает пользователю применение одной или нескольких ролей или запрещает применение всех ролей.

Ключевые слова

ALL

Разрешает применение всех ролей, предоставленных пользователю.

EXCEPT

Применяется для исключения некоторых ролей из списка разрешенных при помощи ключевого слова ALL.

NONE

Запрещает применение всех ролей пользователя.

Аудит

Подобно тому, как мониторинг расходования ресурсов способствует решению проблем производительности, аудит применяется как способ отслеживания использования базы данных и предупреждения о возможных проблемах с безопасностью.

Раньше в Oracle был разрешен аудит трех различных типов:

Аудит команд

Аудит команд, направленных базе данных отдельными пользователями или всеми пользователями.

Аудит привилегий

Аудит использования системных привилегий отдельными пользователями или всеми пользователями.

Аудит объектов схемы

Аудит определенного набора команд SQL для конкретного объекта схемы.

Oracle9*i* вводит четвертый тип аудита, который получил название *детальный аудит* (*fine-grained auditing*). О нем мы поговорим в завершающем разделе главы.

Для всех типов аудита Oracle вносит записи в журнал аудита базы данных или таблицу SYS.FGA_LOG$ или же в файл операционной системы (в двоичном формате). Записи журнала аудита содержат различную информацию в зависимости от типа аудита и установленных параметров выполнения аудита.

Аудит системных действий

Вне зависимости от того, включен ли аудит для вашей базы данных, следующие действия всегда формируют записи в журнале аудита операционной системы:

- Запуск экземпляра
- Остановка экземпляра
- Доступ пользователей с привилегиями администратора

Применение аудита

Включать и выключать аудит можно при помощи команд AUDIT и NOAUDIT.

AUDIT

```
AUDIT инструкция_для_команд_sql | инструкция_для_объектов_схемы
   [BY SESSION | ACCESS]
   [WHENEVER [NOT] SUCCESSFUL]
```

Включает аудит для БД Oracle.

Инструкции

инструкция_для_команд_sql

Инструкция применяется для включения аудита команд или системных привилегий и имеет такой формат:

```
{[параметр_команды | ALL][, ...]} |
 {[системная_привилегия | ALL PRIVILEGES] [, ...]}
 BY {прокси-сервер(, прокси-сервер ...,] ON BEHALF OF [{пользователь [,пользователь ...]} | ANY |
 { пользователь[,пользователь ...]}
```

инструкция_для_объектов_схемы

Инструкция применяется для включения аудита объектов схемы и имеет такой формат:

```
{параметр_объекта[, параметр_объекта ...] | ALL }
 ON {[схема.]объект | DIRECTORY имя_каталога | DEFAULT }
```

Ключевые слова

BY SESSION | ACCESS

Определяет, должны ли данные аудита записываться один раз за сеанс или же при каждой попытке определенного типа доступа. Аудит для всех команд и всех привилегий для команд DDL может задаваться только как BY ACCESS.

WHENEVER [NOT] SUCCESSFUL

Указывает, следует ли проверять только успешные или неудавшиеся команды SQL. Единственные неудавшиеся команды, отслеживаемые при помощи ключевого слова NOT, – это команды, которые не удается выполнить, или приводящие к возникновению ошибок из-за недостаточных привилегий или отсутствия объекта, на который приведена ссылка. По умолчанию проверяются все команды вне зависимости от того, выполнились они успешно или же не удались по упомянутым причинам.

BY пользователь

Задает аудит по одному или нескольким именам пользователей.

BY прокси-сервер ON BEHALF OF

Задает аудит действий, выполняемых прокси-сервером от имени пользователя. Появилась в Oracle8*i*.

параметр_команды

Включает аудит отдельных команд SQL. Перечисляются значения ключевого слова (параметры команд) и команды, которые будет отслеживать каждое из значений для указанного типа объекта. В первой части списка приведены команды, которые будут проверяться, если указать ключевое слово ALL.

CLUSTER

CREATE, AUDIT, DROP, TRUNCATE.

CONTEXT

CREATE, DROP. Появилась в Oracle8*i*.

[PUBLIC] DATABASE LINK

CREATE, DROP.

DIMENSION

CREATE, ALTER, DROP. Появилась в Oracle8*i*.

DIRECTORY

CREATE, DROP.

INDEX

CREATE, ALTER, DROP.

NOT EXISTS

Все команды, которые не удается выполнить из-за того, что объект не существует.

PROCEDURE

CREATE FUNCTION, CREATE LIBRARY, CREATE PACKAGE, CREATE PACKAGE BODY, CREATE PROCEDURE, DROP FUNCTION, DROP LIBRARY, DROP PACKAGE, DROP PROCEDURE.

PROFILE

CREATE, ALTER, DROP.

ROLE

CREATE, ALTER, DROP, SET.

ROLLBACK SEGMENT

CREATE, ALTER, DROP.

SEQUENCE

CREATE, DROP.

SESSION

Начало сеанса.

[PUBLIC] SYNONYM

CREATE, DROP.

SYSTEM AUDIT

AUDIT *системные_привилегии_и_роли*; NOAUDIT *системные_привилегии_и_роли*.

SYSTEM GRANT

GRANT системные_привилегии_и_роли; REVOKE системные_привилегии_и_роли.

TABLE

CREATE, DROP, TRUNCATE.

TABLESPACE

CREATE, ALTER, DROP.

TRIGGER

CREATE, ALTER с инструкциями ENABLE или DISABLE, DROP, ALTER TABLE с инструкциями ENABLE или DISABLE.

TYPE

CREATE, CREATE TYPE BODY, ALTER, DROP, DROP TYPE BODY.

USER

CREATE, ALTER, DROP.

VIEW

CREATE, DROP.

Далее приведены ключевые слова, которые могут применяться для включения аудита команд, но не проверяются при задании ключевого слова ALL. Если специально не указано иное, такие дополнительные ключевые слова включают только аудит соответствующей команды.

ALTER SEQUENCE

ALTER TABLE

COMMENT TABLE

Аудит команды COMMENT для таблицы, представления, материализованного представления или столбцов каждого из таких объектов.

DELETE TABLE

EXECUTE PROCEDURE

Аудит CALL.

GRANT DIRECTORY

GRANT и REVOKE для каталога

GRANT PROCEDURE

GRANT и REVOKE для процедуры.

GRANT SEQUENCE

GRANT и REVOKE для последовательности.

GRANT TABLE

GRANT и REVOKE для таблицы, представления или материализованного представления.

GRANT TYPE

GRANT и REVOKE для типа.

INSERT TABLE

INSERT INTO для таблицы или представления.

LOCK TABLE

LOCK для таблицы или представления.

SELECT SEQUENCE

Любая команда, содержащая CURRVAL или NEXTVAL, для последовательности.

SELECT TABLE

SELECT FROM для таблицы, представления или материализованного представления.

UPDATE TABLE

UPDATE для таблицы или обновляемого представления.

ALL (параметр_команды)

См. приведенный выше в описании *параметра_команды* список команд, которые будет отслеживать ALL.

системная_привилегия

Указывает системные привилегии, для которых будет проводиться аудит. Можно указать роли CONNECT, RESOURCE или DBA для отслеживания всех системных привилегий, включенных в роль.

ALL PRIVILEGES

Отслеживание всех системных привилегий.

прокси-сервер

Указывает, следует ли отслеживать все действия прокси-сервера или же только выполненные от имени определенного пользователя.

пользователь

Пользователи, действия которых отслеживаются.

параметр_объекта

Для каждого объекта можно отслеживать один или несколько способов доступа к нему. Приведем список объектов и возможностей выполнения аудита для них:

Контекст

GRANT. Этот вид аудита появился в Oracle8*i*.

Каталог

AUDIT, GRANT, READ.

Библиотека

GRANT, READ.

Материализованное представление

ALTER, AUDIT, COMMENT, DELETE, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE. В Oracle8 и более ранних версиях эти параметры объектов существовали под именем объекта «моментальная копия данных». Начиная с Oracle8*i* они могут использоваться как для материализованных представлений, так и для моментальных копий данных.

Объектный тип

ALTER, AUDIT, GRANT. Этот вид аудита появился в Oracle8*i*.

Процедура, функция, пакет

AUDIT, EXECUTE, GRANT, RENAME. Этот вид аудита доступен для PL/SQL или Java в версии Oracle8*i* и выше.

Последовательность

ALTER, AUDIT, GRANT, SELECT.

Таблица

ALTER, AUDIT, COMMENT, DELETE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE.

Представление

AUDIT, COMMENT, DELETE, GRANT, INSERT, LOCK, RENAME, SELECT, UPDATE.

[схема.]объект

Объект, аудит которого будет проводиться. Если имя схемы не указано, то сервер Oracle считает, что объект находится в схеме текущего пользователя.

имя_каталога

Имя каталога, для которого будет проводиться аудит.

DEFAULT

Указывает параметры проведения аудита по умолчанию для всех объектов, созданных после выполнения данной команды.

NOAUDIT

```
NOAUDIT инструкция_для_команды_sql | инструкция_для_объекта_схемы
    [BY SESSION | ACCESS]
```

Выключает все ранее включенные виды аудита.

Ключевые слова

Ключевые слова и инструкции NOAUDIT имеют те же значения, что и для команды AUDIT.

Другие возможности обеспечения безопасности

Описанные далее возможности обеспечения безопасности по большей части относятся к администраторам баз данных Oracle или специалистам, отвечающим за безопасность. Приведем лишь краткий обзор таких функций, подробную же информацию можно найти в документации Oracle.

Представления и хранимые процедуры

Помимо способов обеспечения безопасности данных БД Oracle, рассмотренных ранее, существуют и другие возможности. До выхода версии Oracle8*i* самым распро-

страненным способом ограничения доступа к данным в зависимости от их значений было использование представлений и хранимых процедур.

Представления

Вы определяете подмножество данных таблицы и предоставляете пользователю доступ только к представлению. Начиная с Oracle8*i* вы можете добиться такого же уровня безопасности при помощи детального контроля доступа, описанного в следующем разделе.

Хранимые процедуры

Вы можете ввести аналогичное ограничение на доступ к таблице используя хранимую процедуру или пакет, выдав пользователям привилегии на хранимую процедуру или пакет. Код хранимой процедуры может содержать собственный набор правил подтверждения правильности.

В последующих разделах поговорим о дополнительных способах контроля доступа.

Детальный контроль доступа и политика безопасности

Детальный контроль доступа (*fine-grained access control*) предлагает способ контекстного обеспечения безопасности, который может быть реализован в коде приложения или в представлениях. Благодаря тому что детальный контроль доступа реализуется на уровне базы данных, он единообразно действует для всех приложений.

Детальный контроль доступа появился в версии Oracle8*i*. Он реализуется специальными правилами для таблиц, регламентирующими права доступа к этим таблицам. *Политика безопасности* (*security policy*) реализуется программным модулем, который может предоставлять доступ на основе логического условия любого вида. Для всех команд SQL, выполняемых на базе данных, создается предикат (условие, добавляемое в инструкцию WHERE и ограничивающее доступ к данным), который автоматически может применяться для обеспечения безопасности на основе содержимого таблицы.

Виртуальные частные базы данных

Механизм контекстных мер безопасности позволяет создавать виртуальные частные базы данных (VPD, Virtual Private Database). Благодаря применению VPD сразу несколько пользователей могут видеть свои собственные представления одной или нескольких таблиц базы данных. Ранее говорилось о том, что подобные меры безопасности можно обеспечить за счет создания и поддержания представлений, но применение VPD избавляет от забот по созданию представлений и разграничению доступа к ним для отдельных пользователей и групп пользователей.

Метки безопасности и управление ими

Метки безопасности появились в Oracle9*i*. Они представляют собой расширение VPD; различие в том, что программные модули для поддержки VPD уже написаны и действуют для значений единственного столбца, содержащего метки. Поэтому для реализации меток безопасности не требуется специального программирования. Метки безопасности относятся к дополнительным компонентам Oracle9*i* Enterprise Edition.

Policy Manager, инструмент администрирования с графическим интерфейсом пользователя, входящий в состав Enterprise Manager, также появился в версии Oracle9*i*. Он реализует управление метками безопасности.

Контекст приложения

Контекст приложения (*application context*), введенный в Oracle8*i*, позволяет установить для пользователя некоторые атрибуты, которые будут действовать в течение всего пользовательского сеанса. Применяя такие атрибуты для предоставления доступа, можно создать роль для конкретного приложения, которая продолжает существовать в базе данных на всем протяжении работы приложения. Контекст приложения может использоваться для реализации детального контроля доступа.

Детальный аудит

Детальный аудит (*fine-grained auditing*) появился в Oracle9*i*. Как и детальный контроль доступа, детальный аудит реализуется за счет определения предиката, вводящего ограничение для команд SQL, которые будут подвергнуты аудиту. Проводя такую политику аудита, вы можете сконцентрироваться на отслеживании действий с небольшим объемом жизненно важных данных. Уменьшение общего количества записей аудита облегчает выявление потенциальных проблем с безопасностью.

LogMiner

LogMiner – это инструментальное средство, позволяющее применять команды SQL для анализа событий в журнале базы данных. LogMiner позволяет отслеживать транзакции по мере их обработки или выявлять, какие конкретно функции вызвали изменение данных. Утилита LogMiner появилась в Oracle8*i*.

С помощью LogMiner (наряду с журналами аудита) можно определить, что происходило с вашей базой данных Oracle. В версии Oracle9*i* LogMiner включен в Enterprise Manager.

Исследование возможностей LogMiner лежит за пределами нашей книги. Подробную информацию можно найти в документации по Oracle.

Oracle Advanced Security

Продукт Oracle Advanced Security, который ранее назывался Secure Network Services, а затем Advanced Network Services, – это пакет расширения, предлагающий мощные средства шифрования данных. Oracle Advanced Security обеспечивает дополнительные возможности по обеспечению безопасности в трех областях:

Сетевая безопасность

Шифрование сообщений, передаваемых службами Oracle Net Services, реализация SSL-шифрования (Secure Sockets Layer – протокола защищенных сокетов) и поддержка RADIUS, Kerberos, смарт-карт, карточек-идентификаторов (token cards) и биометрической аутентификации.

Безопасность пользователей предприятия

Включает применение разнообразных сторонних средств поддержки каталогов, таких как LDAP-каталоги, с помощью которых можно реализовать возможность однократной регистрации. Oracle Advanced Security включает в себя сервис каталогов Oracle Internet Directory (OID), описанный в следующем разделе.

Безопасность инфраструктуры открытых ключей

Включает поддержку стандартных сертификатов X.509 версии 3. Oracle работает с основными поставщиками сервисов инфраструктуры открытых ключей (Public

Key Infrastructure – PKI), такими как Baltimore Technologies и VeriSign, для обеспечения координации с их доверенными корневыми сертификатами.

Oracle Advanced Security внедряет эти сервисы на уровне Oracle Net Services, который реализует взаимодействие между сервером и клиентом, о чем говорится в главе 5. Кроме того, Oracle Advanced Security можно использовать с драйвером Thin JDBC, не содержащим Oracle Net Services.

Oracle Advanced Security включает в себя Oracle Enterprise Security Manager, графический интерфейс пользователя для управления доменами и пользователями предприятия.

В Oracle9*i* можно шифровать данные на сервере с помощью пакета DBMS_OBFUSCATION_TOOLKIT (без участия Oracle Advanced Security). Дополнительная информация о пакете приведена в главе 10.

Интернет-каталог Oracle

В данной главе несколько раз упоминались внешние каталоги. *Внешний каталог* (*external directory*) – это средство для хранения информации о базе данных, такой как имена пользователей и полномочия. Внешний каталог может быть связан с несколькими экземплярами Oracle внутри предприятия. Oracle предлагает свой собственный внешний каталог – интернет-каталог Oracle (Oracle Internet Directory – OID). OID отвечает стандартам упрощенного протокола доступа к сетевым каталогам (Lightweight Directory Access Protocol – LDAP), разработанного в Мичиганском университете.

С помощью OID или других каталогов LDAP можно создать способ аутентификации, который будет охватывать несколько баз данных. Можно применять внешние каталоги и в других целях в глобальной IT-структуре предприятия. OID поддерживает три вида аутентификации: анонимная, основанная на пароле и основанная на сертификате.

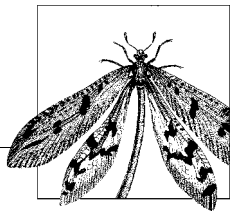
В состав OID входит сервер репликации каталогов Oracle и инструмент администрирования с графическим интерфейсом пользователя.

Права вызывающего

До версии Oracle8*i* хранимая процедура обладала собственным набором привилегий. Каждый, кто использовал некоторую хранимую процедуру для доступа к данным, использовал привилегии, выданные процедуре, вне зависимости от того, какими привилегиями обладал сам пользователь.

Начиная с Oracle8*i* при создании функции, процедуры, пакета, объектного типа или Java-кода пользователь может указать инструкцию для полномочий вызывающего объект пользователя. Если такая инструкция присутствует, то объект будет выполняться с привилегиями пользователя, а не самого объекта.

5
Работа в сети



СУБД Oracle – это не только сервер. Клиентские приложения должны иметь возможность подключаться к экземпляру Oracle, а экземпляры Oracle – взаимодействовать с другими экземплярами, а также с серверами приложений, чужими базами данных и внешними процедурами.

Глава дает общее представление о сетевой работе СУБД Oracle и приводит синтаксис файлов конфигурации, которые необходимо создать для того, чтобы сделать сеть частью среды Oracle. Речь пойдет о файлах *TNSNAMES.ORA*, *SQLNET.ORA*, *LISTENER.ORA*, *LDAP.ORA*, *NAMES.ORA* и *CMAN.ORA*.

Основы работы Oracle в сети

Программное обеспечение, применяемое для организации работы Oracle в сети, за годы своего существования сменило несколько имен. До версии Oracle8 оно называлось SQL*Net, в Oracle8 появилось новое имя – Net8, а в Oracle9*i* оно превратилось в более общее Oracle Net Services. Далее в этой главе для всех версий будет употребляться только имя Oracle Net Services с указанием отличий в версиях.

Имена меняются, но предназначение этого программного обеспечения остается прежним. Если клиенту экземпляра Oracle необходимо установить соединение с базой данных Oracle (независимо от того, обычный ли это клиент, сервер приложений или какой-то другой сервер), он должен подключиться к одному из концов сетевого соединения, установленного сетевым программным обеспечением Oracle. Сетевое программное обеспечение осуществляет прозрачное соединение с экземпляром Oracle.

Oracle Net Services поддерживает множество разнообразных клиентских и серверных платформ и множество сетевых протоколов. Лежащей в основе архитектурой и всеми сложностями сетевой инфраструктуры занимается Oracle Net Services. Но пользователю предстоит управлять службами Oracle Net Services. В этой главе будет рассказано о файлах конфигурации, отвечающих за настройку Oracle Net Services.

Oracle Net Services реализует сетевую передачу данных и сообщений, но это не единственная функция сетевых служб. Некоторые дополнительные компоненты (например, Shared Server, до выхода Oracle9*i* известный под именем Multi-Threaded Server), предназначенные для увеличения производительности за счет снижения требований к сетевым ресурсам, переключения при сбоях или выравнивания нагрузки, также реализуются при помощи Oracle Net Services.

Создание соединения

Oracle Net Services обеспечивает передачу информации к серверу и от него. Компоненты Oracle Net Services, реализующие соединение, и соответствующие им файлы конфигурации представлены на рис. 5.1.

Процедура установки соединения клиента с сервером состоит из двух основных этапов. Сначала клиент должен определить местонахождение нужного сервера, что можно сделать следующими способами:

- Просмотреть локальный файл *TNSNAMES.ORA*.
- Получить информацию от внешнего каталога LDAP (возможность появилась в Oracle8*i*) или от внешней службы имен.
- Использовать централизованную службу имен, такую как Oracle Names (предполагается, что она не будет поддерживаться в версиях выше Oracle9*i*).

Служба Oracle Names должна была действовать как централизованный каталог для сетевых служб. К сожалению, эта технология защищена патентом, тогда как новые каталоги LDAP выполняют те же функции и их использование не ограничено правом собственности. Поэтому корпорация Oracle объявила, что Oracle Names больше не будет усовершенствоваться, и предоставила пользователям Oracle Names возможности миграции на каталоги LDAP и применения серверов Oracle Names в качестве прокси-серверов для каталогов LDAP.

- Подключиться к хосту напрямую, используя TCP/IP или идентификатор DNS (начиная с версии Oracle8); но в данном случае будут недоступны некоторые возможности Oracle Net Services, например, организация пула соединений.

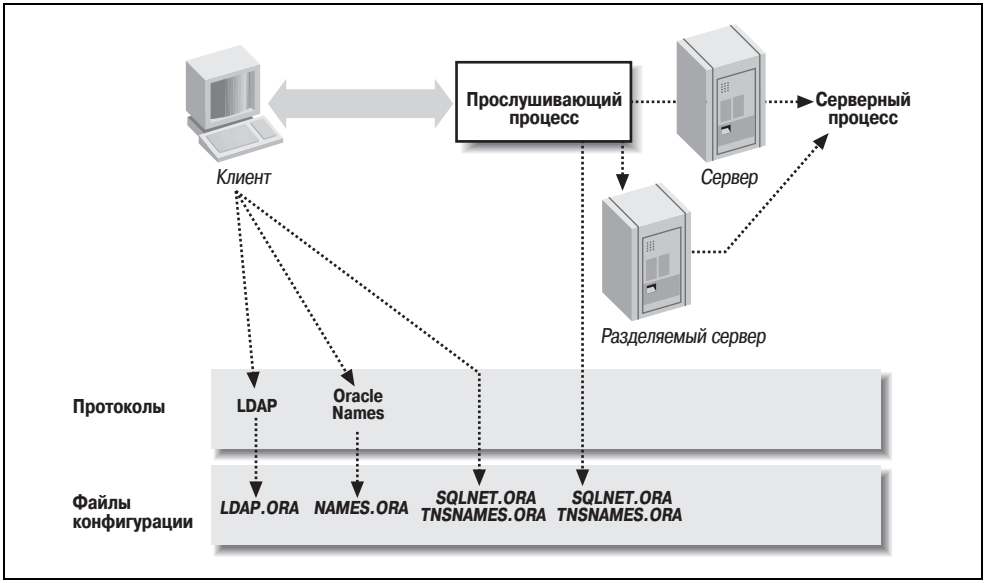


Рис. 5.1. *Работа сети и ее конфигурирование*

Фактически клиент подключается к прослушивающему процессу Oracle на компьютере, где работает сервер. При получении запроса на соединение прослушивающий процесс Oracle выполняет второй шаг по установке соединения. Он создает назначаемый клиентскому процессу специальный серверный процесс, используя предварительно запущенный процесс БД для клиента или же диспетчер, который будет обрабатывать входящие запросы от клиента и отправлять их для выполнения разделяемому серверу.

На сервере может быть несколько прослушивающих процессов – для обеспечения резервирования или организации раздельных сетевых сред. Прослушивающий процесс может поддерживать несколько экземпляров БД. Кроме того, прослушивающий процесс может прослушивать один или несколько IP-адресов.

Переключение при сбое

Сетевые службы Oracle Net Services поддерживают *переключение при сбое* (*failover*) в случае сбоя экземпляра как в момент соединения, так и в процессе работы приложения. Разрешая переключение при сбое, вы даете возможность пользователю, подключенному к поврежденному экземпляру, продолжить нормальную работу с работающим экземпляром. Для того чтобы включить режим переключения при сбое, достаточно перечислить несколько прослушивающих процессов в записи DESCRIPTION файла *TNSNAMES.ORA*.

Переключение при сбое в момент соединения (*connect-time failover*) позволяет направить запросы на установку соединения резервному экземпляру в случае, если основной экземпляр вышел из строя. Вероятно, максимальную пользу его применение приносит в сочетании с дополнительным компонентом Real Application Clusters (когда несколько экземпляров обслуживают одну базу данных). Для того чтобы разрешить переключение при сбое в момент соединения, достаточно указать несколько прослушивающих процессов в записи DESCRIPTION файла *TNSNAMES.ORA* (переключение при сбое также можно использовать при работе в средах с резервными базами данных для перенаправления запроса резервной базе данных в случае отключения основной).

Переключение при сбое в процессе работы приложения берет на себя компонент TAF (*Transparent Application Failover*), обеспечивающий прозрачное переключение. Если соединение завершается со сбоем и режим TAF включен, то пользователь автоматически переключается на другой прослушивающий процесс. Можно даже указать необходимость установки соединения с другим прослушивающим процессом уже в момент открытия начального соединения с первым прослушивающим процессом, чтобы сократить время восстановления. Кроме того, большая часть запросов, выполняющихся при включенном TAF, отслеживают свое текущее состояние, поэтому следующая строка, извлеченная после переключения при сбое, будет той же строкой, которая была бы извлечена, если бы сбоя не произошло. Только имейте в виду, что новому соединению придется для получения следующей строки прочитать все предыдущие, что может сказаться на производительности (в зависимости от размера результирующего множества и позиции внутри него).

Для того чтобы получить доступ к преимуществам TAF, приложение должно работать непосредственно через интерфейс вызовов СУБД Oracle (Oracle Call Interface – OCI) и использовать вызовы, поддерживающие переключение. Существуют и другие ограничения на применение TAF. Обратитесь к документации Oracle, для того чтобы выяснить, какие именно ограничения налагаются на применение TAF в вашей версии Oracle.

Выравнивание нагрузки

Выравнивание нагрузки можно осуществлять как на стороне сервера, так и на стороне клиента:

- На стороне клиента выравнивание нагрузки запрашивается с указанием в файле *TNSNAMES.ORA* нескольких прослушивающих процессов (указываются несколько прослушивающих процессов так же, как для переключения при сбое). Включение выравнивания нагрузки приводит к тому, что клиентское соединение произвольным образом выбирает один из адресов прослушивающих процессов в файле *TNSNAMES.ORA*.
- На стороне сервера разделяемые серверы, описанные в следующем разделе, обеспечивают выравнивание нагрузки на прослушивающие процессы. Новые соединения направляются наименее загруженному диспетчеру.

Разделяемые серверы

Обычно Oracle Net Services создает один выделенный серверный процесс для каждого пользовательского соединения. Для обслуживания каждого серверного процесса необходим некоторый объем памяти, даже если процесс не выполняет никаких активных действий.

Во избежание чрезмерного расходования ресурсов соединениями Oracle Net Services предлагает использовать *разделяемый сервер* (*shared server*), до выхода версии Oracle9*i* известный как многопотоковый сервер (Multi-Threaded Server – MTS). Разделяемый сервер – это серверный процесс, способный обслуживать несколько пользовательских соединений. При работе разделяемого сервера не создается выделенный серверный процесс, вместо этого прослушивающий процесс отправляет запрос на соединение диспетчеру. Диспетчер берет на себя ответственность за координацию взаимодействия базы данных с клиентом и направляет последующие запросы одному из существующих разделяемых серверов.

Возможна работа одного экземпляра с несколькими диспетчерами и несколькими разделяемыми серверами. Для одного экземпляра могут существовать и выделенный процесс, и разделяемый сервер. Например, можно создать разные службы имен для одного экземпляра, одну для выделенного серверного процесса, другую – для разделяемого сервера.

За инициализацию разделяемых серверов отвечают соответствующие параметры файла инициализации экземпляра, а не параметры файлов конфигурации Oracle Net Services, которые будут описаны далее в главе. Подробная информация о параметрах инициализации приведена в главе 2. Установив параметр SERVER в файле *TNSNAMES.ORA* или *USE_DEDICATED_SERVER* в *SQLNET.ORA*, можно принудительно задать работу с выделенным сервера, даже если в составе экземпляра действуют разделяемые серверы.

Наиболее эффективно применение разделяемых серверов в тех конфигурациях, где к базе данных обращается множество пользователей и каждый пользователь большую часть времени бездействует. Противоположным случаем является *пакетное задание* (*batch job*), которому всегда должен назначаться выделенный сервер.

Разделяемые серверы уменьшают загрузку памяти, но обычно работают медленнее, чем выделенные серверы, т. к. возможна некоторая задержка при отсутствии доступных серверов.

Выделенные серверы расходуют память программной глобальной области (PGA) выделенного процесса, а разделяемые серверы – память большого пула или разделяемого пула, т. к. существует возможность, что доступ к информации о соединении потребуется нескольким процессам разделяемых серверов.

Диспетчер соединений

Процессы выделенных серверов увеличивают нагрузку на систему, так же дело обстоит и с сетевыми соединениями. Каждое сетевое соединение расходует какую-то часть пропускной способности сети. Диспетчер соединений Oracle (Connection Manager), появившийся в версии Oracle8 Enterprise Edition, занимает одно сетевое соединение, обеспечивая при этом три важнейших преимущества:

Несколько сетевых соединений

Диспетчер соединений, работая совместно с разделяемыми серверами, позволяет поддерживать множество сетевых соединений. Такое мультиплексирование снижает накладные расходы.

Преобразование сетевых протоколов

Диспетчер соединений способен служить преобразователем сетевых протоколов в гетерогенной сети. Клиент может использовать один протокол при входе в диспетчер соединений и другой – на пути от диспетчера соединений к серверу Oracle.

Преобразование сетевых адресов

Можно применять диспетчер соединений для преобразования сетевых адресов (NAT). Когда клиент подключается к прослушивающему процессу Oracle, его соединение перенаправляется либо выделенному серверному процессу, либо диспетчеру, который отправляет свой адрес клиенту. Если сервер находится за межсетевым экраном, то отправленный им внутренний адрес может оказаться недоступным клиенту. Эту проблему можно обойти, применяя диспетчер соединений в качестве посредника.[1]

Диспетчер соединений также позволяет осуществлять фильтрацию TCP/IP-трафика по IP-адресам отправителя или получателя или по имени службы базы данных.

Диспетчер соединений может работать на нескольких связанных между собой компьютерах, при этом в каждом из соединений между машинами могут применяться мультиплексирование, преобразование протоколов и фильтрация IP-трафика.

Подробные сведения о диспетчере соединений можно найти в документации Oracle.

Файлы конфигурации

Работа сетевых служб Oracle Net Services регулируется информацией, хранящейся в соответствующих файлах конфигурации:

TNSNAMES.ORA

Этот клиентский файл содержит текстовые определения для имен служб Oracle. Он предоставляет всю необходимую информацию для преобразования локальных имен служб в сетевые адреса, которые могут использоваться Oracle Net Services,

[1] По этому поводу чрезвычайно полезную статью «Lock the Door on Connection Manager» написал Джонатан Генник (Jonathan Gennick), см. *http://gennick.com/lock_the_door.html*.

а также сведения, необходимые некоторым специальным функциям, поддерживаемым Oracle Net Services. Пользователи, применяющие внешнюю службу имен, именование хостов, службу Oracle Names или LDAP (именование каталогов), не нуждаются в данном файле.

SQLNET.ORA

Этот файл содержит параметры конфигурации для Oracle Net Services, а именно:

- Список методов назначения имен и порядок, в котором их следует пытаться применять
- Домен по умолчанию
- Каталог и имя файла для сетевых файлов трассировки
- Каталог и имя файла для сетевых журнальных файлов

LISTENER.ORA

Серверный файл содержит параметры конфигурации для прослушивающего процесса (процессов) на сервере Oracle.

LDAP.ORA

Данный файл параметров содержит информацию о LDAP-сервере, который будет использоваться. Появился в версии Oracle8*i*.

NAMES.ORA

Файл параметров, который необходим, только если Oracle Net Services применяется совместно с Oracle Names. Файл содержит параметры, отвечающие за управление работой сервера Oracle Names. Корпорация Oracle выводит Oracle Names из употребления, и в настоящее время с этим продуктом работает лишь небольшое количество пользователей Oracle, поэтому в книге отсутствуют подробные сведения о *NAMES.ORA* (вы можете обратиться за ними к документации Oracle).

CMAN.ORA

Файл параметров, необходимость в котором возникает, только если Oracle Net Services применяется в сочетании с диспетчером соединений. Он содержит параметры, отвечающие за управление работой диспетчера соединений. В настоящее время с этим продуктом работает лишь небольшое количество пользователей Oracle, поэтому мы не будем подробно рассматривать файл *CMAN.ORA*. Дополнительную информацию можно найти в документации Oracle.

Следующие разделы описывают синтаксис и ключевые слова, употребляемые в файлах *TNSNAMES.ORA*, *SQLNET.ORA*, *LISTENER.ORA* и *LDAP.ORA*. Значения, принимаемые по умолчанию, выделены полужирным шрифтом.

SQLNET.ORA

SQLNET.ORA – это файл конфигурации, управляющий работой Oracle Net Services. Он должен присутствовать и на сервере, и у клиента.

По умолчанию файл *SQLNET.ORA* хранится в каталоге *$ORACLE_HOME\network\admin* (в версии Oracle8 файл хранился в каталоге *\net80\admin*). Можно задать каталог хранения данного файла в переменной окружения TNS_ADMIN.

Синтаксис и ключевые слова

*BEQUEATH_DETACH=YES|**NO***

Определяет, следует ли применять обработку сигналов Unix для завершения серверных процессов, порожденных Bequeath-соединениями. Значение YES переда-

ет ответственность за завершение серверного процесса UNIX-процессу *init*. В этом случае обработка сигналов не применяется. Значение NO означает, что завершить серверный процесс, когда тот уже не будет нужен, должен будет процесс, породивший серверный процесс с помощью Bequeath. В этом случае обработка сигнала применяется. Значение по умолчанию – NO.

DAEMON.TRACE_DIRECTORY = путь_к_каталогу

Задает каталог для хранения трассировочных файлов, сформированных демоном Oracle Enterprise Manager (EM). Этот параметр действует только для версий 1.x Enterprise Manager. Каталог по умолчанию – *$ORACLE_HOME/network/trace*. Начиная с Oracle9*i* параметр не поддерживается.

*DAEMON.TRACE_LEVEL = **OFF** | USER | ADMIN | SUPPORT*

Определяет уровень детализации трассировки для демона Oracle Enterprise Manager. Этот параметр действует только для версий 1.x Enterprise Manager. Значение по умолчанию – OFF. Начиная с Oracle9*i* параметр не поддерживается.

OFF

Вывод трассировочной информации не ведется.

USER

Генерирует трассировочную информацию на пользовательском уровне подробности.

ADMIN

Генерирует трассировочную информацию на уровне администратора. Вывод будет более подробным, чем при выборе USER.

SUPPORT

Формирует чрезвычайно подробную трассировочную информацию. Это наивысший уровень подробности, выводится гораздо больше информации, чем при выборе USER или ADMIN.

DAEMON.TRACE_MASK = (маска)

Ограничивает запись в трассировочный файл – туда попадут только записи, соответствующие указанной маске. Этот параметр действует только для версий 1.x Enterprise Manager. Начиная с Oracle9*i* параметр не поддерживается.

*DISABLE_OOB = **OFF** | ON*

Определяет, разрешены ли внеполосные (out-of-band – OOB) прерывания. Речь идет о механизме, применяемом Oracle Net Service для передачи срочных сообщений между клиентом и сервером, который позволяет прерывать долго выполняющиеся запросы нажатием клавиш CTRL-C. Значение по умолчанию – OFF, оно разрешает OOB-прерывания. Значение ON запрещает внешнее прерывание.

LOG_DIRECTORY_CLIENT = путь_к_каталогу

Указывает каталог для записи клиентских журнальных файлов Oracle Net Services. Клиентские журнальные файлы формируются при работе с Oracle Net Services в режиме клиента. До выхода версии Oracle9*i* каталогом хранения журналов по умолчанию был *$ORACLE_HOME/network/log*; в Oracle9*i* журналы по умолчанию сохраняются в каталоге запуска исполняемого файла.

LOG_DIRECTORY_SERVER = путь_к_каталогу

Указывает каталог для записи серверных журнальных файлов Oracle Net Services. До выхода версии Oracle9*i* каталогом хранения журналов по умолчанию был

$ORACLE_HOME/network/log; в Oracle9*i* журналы по умолчанию сохраняются в каталоге запуска исполняемого файла.

LOG_FILE_CLIENT = имя_файла

Задает имя файла для клиентских журналов Oracle Net Services. Имя файла по умолчанию – *sqlnet.log*.

LOG_FILE_SERVER = имя_файла

Задает имя для серверных журнальных файлов Oracle Net Services. Имя файла по умолчанию – *sqlnet.log*.

NAMES.CONNECT_TIMEOUT = секунды

Период времени, в течение которого клиент ожидает соединения с сервером Oracle Names. Параметр появился в Oracle9*i*.

NAMES.DCE.PREFIX = префикс

Применяется в среде распределенных вычислений (DCE – Distributed Computing Environment) для указания имени ячейки (префикса), используемого при поиске имени. Работает, только если в качестве метода именования была выбрана служба каталогов ячейки DCE (Cell Directory Services – CDS).

NAMES.DEFAULT.DOMAIN = ONS_домен

Задает домен по умолчанию, присоединяемый к именам сетевых служб, в которых отсутствует компонент домена. Данный домен будет добавлен в конец каждого имени сетевой службы, не содержащего точку (.). Значением по умолчанию является пустая строка, или NULL. До версии Oracle8 значением по умолчанию было WORLD.

NAMES.DIRECTORY_PATH = (метод именования[, метод именования...])

Указывает метод (или методы) именования, применяемый Oracle Net Services при попытке разрешения имени сетевой службы. При этом *метод_именования* может иметь следующие значения:

TNSNAMES
LDAP (появилось в Oracle9*i*)
ONAMES
HOSTNAME
DCE (до Oracle9*i*) или CDS (начиная с Oracle9*i*)
NIS
NOVELL (не поддерживается в Oracle9*i*)

NAMES.INITIAL_RETRY_TIMEOUT = секунды

Задает период времени (в секундах), в течение которого следует ждать ответа от одного сервера Oracle Names, прежде чем обращаться к следующему серверу имен из списка. Значение по умолчанию зависит от операционной системы, но часто составляет 15 секунд. Разрешены значения от 1 до 600 секунд.

NAMES.MAX_OPEN_CONNECTIONS = макс_соединений

Определяет максимальное количество соединений Oracle Net Services, которые разрешено иметь клиенту Oracle Names в любой определенный момент времени. Значение по умолчанию равно 10. Разрешены значения от 3 до 64.

NAMES.MESSAGE_POOL_START_SIZE = число_сообщений

Указывает исходное распределение сообщений в пуле сообщений клиента. Значение по умолчанию равно 10. Разрешены значения от 3 до 256.

NAMES.NIS.META_MAP = имя_файла_соответствий

Действует, только если для именования применяются сетевые информационные службы Sun (NIS – Network Information Service). Указывает файл соответствий, определяющий, каким образом атрибуты NIS используются для определения имени сетевой службы. Имя файла по умолчанию – *sqlnet.maps*.

NAMES.PREFERRED_SERVERS=(ADDRESS_LIST = (адрес) [(адрес)...])

Задает список серверов Names, которые будут использоваться, если для определения имени сетевой службы выбрана служба Oracle Names. Значение по умолчанию не определено, но если для обнаружения в сети доступных серверов Names применялась утилита Names Control или Configuration Assistant, то файл *$ORACLE_HOME/network/names/sdns.ora* также будет содержать список доступных для использования серверов имен. Если установлен параметр NAMES.PREFERRED_SERVERS, то его значение перекрывает значение (список серверов), указанное в файле *sdns.ora*.

NAMES.REQUEST_RETRIES = количество_попыток

Устанавливает количество попыток обращения к определенному серверу Names, которые клиент должен осуществить, прежде чем счесть его недоступным. Если в качестве метода именования применяется Oracle Names, то при определении имен служб часто будете пользоваться списком серверов Names (см. описание NAMES.PREFERRED_SERVERS чуть выше). При определении имени некоторой сетевой службы сначала предпринимается попытка обращения к первому серверу Names из этого списка. Если установить соединение не удается, Oracle Net Services повторяет попытку столько раз, сколько указано в параметре NAMES.REQUEST_RETRIES. Если соединение так и не установлено, то Oracle Net Services переходит к следующему серверу Names из списка. Значением по умолчанию является 1. Разрешены значения от 1 до 5.

NAMESCTL.ECHO=TRUE | FALSE

Определяет, отражает ли утилита Oracle Names Control в журнале вывода команды вместе с приглашениями на ввод. Появился в Oracle9*i*.

*NAMESCTL.INTERNAL_ENCRYPT_PASSWORD = **TRUE** | FALSE*

Определяет, будет ли утилита Names Control шифровать пароли при отправке их серверу Names. По умолчанию установлено значение TRUE, т. е. пароли шифруются. Если этот параметр равен FALSE, то пароли будут переданы в открытом виде.

*NAMESCTL.INTERNAL_USE = TRUE | **FALSE***

Позволяет выполнять недокументированные команды утилиты Names Controls. Они применяются для выявления неполадок и начинаются с символа подчеркивания (_). Значение TRUE разрешает использование таких команд. По умолчанию параметр равен FALSE, что отключает недокументированные команды. Для того чтобы получить информацию о недокументированных командах, разрешите их выполнение и введите команду HELP в утилите Names Control.

*NAMESCTL.NO_INITIAL_SERVER = TRUE | **FALSE***

Определяет, будет ли утилита Names Control пытаться подключиться к используемому по умолчанию серверу Names при ее первом запуске. По умолчанию равен FALSE. Такое значение параметра вызывает небольшую задержку при первом запуске Names Control, т. к. утилита пытается получит доступ к серверу Names вашей сети. Если параметр равен TRUE, то Names Control не обращается к серве-

ру Names до тех пор, пока не будет выдана команда SET SERVER. В этом случае Names Control запускается быстрее.

*NAMESCTL.NOCONFIRM = ON | **OFF***

Определяет, будет ли Names Control запрашивать подтверждение выполнения некоторых особенно важных команд, а именно: STOP, SHUTDOWN, RELOAD и RESTART. По умолчанию равен OFF, т. е. подтверждения не запрашиваются. Установите параметр в ON, для того чтобы получать предупреждения при выдаче одной из перечисленных команд.

NAMESCTL.SERVER_PASSWORD = пароль

Позволяет задать пароль сервера Names в файле *SQLNET.ORA*, а не при помощи команды SET PASSWORD, вводя его при каждом запуске утилиты Names Control. Пароль должен совпадать с заданным в параметре NAMES.PASSWORD серверного файла *NAMES.ORA*.

NAMESCTL.TRACE_DIRECTORY = путь_к_каталогу

Указывает каталог, в котором будут создаваться файлы трассировки Names Control. Значением по умолчанию является *$ORACLE_HOME/network/trace*.

NAMESCTL.TRACE_FILE = имя_файла

Задает имя файла, в который будет записываться трассировочная информация Names Control. Такой файл создается и записывается, только если значение параметра NAMESCTL.TRACE_LEVEL отлично от OFF. Имя файла по умолчанию – *namesctl_pid.trc*, где *pid* – это идентификатор процесса.

*NAMESCTL.TRACE_LEVEL = **OFF** | USER | ADMIN | SUPPORT*

Управляет объемом трассировочной информации, которая формируется при запуске утилиты Names Control. Разъяснения для ключевых слов приведены в описании параметра DAEMON.TRACE_LEVEL.

*NAMESCTL.TRACE_UNIQUE = **ON** | OFF*

Определяет, добавляется ли к именам трассировочных файлов идентификатор процесса (см. описание параметра NAMESCTL.TRACE_FILE). По умолчанию равен ON, т. е. имена файлов трассировки уточняются за счет указания идентификатора процесса. Параметр не работает в Windows NT.

OSS.SOURCE.LOCATION = (SOURCE = (METHOD = ORACLE) (METHOD_DATA = (SQLNET_ADDRESS = имя_сетевой_службы)))

Сообщает службе Oracle Net Services, каким образом и откуда следует извлекать зашифрованные секретные ключи. Параметр не поддерживается в Oracle9*i*.

OSS.SOURCE.MY_WALLET = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY = путь_к_каталогу)))

Указывает каталог, в котором хранятся *накопители* (*wallets*) SSL. В настоящее время поддерживается только метод FILE. Параметр не поддерживается в Oracle9*i*.

SQLNET.AUTHENTICATION_GSSAPI_SERVICE = принципал[1]

Используется в рамках компонента Oracle Advanced Security для определения принципала службы CyberSafe.

---

[1] Принципал (principal) – термин, употребляемый в системах контроля прав доступа для обозначения пользователя или процесса, имеющего учетную запись в данной среде, к которой могут быть применены различные разрешения. – *Примеч. перев.*

SQLNET.AUTHENTICATION_KERBEROS5 = имя_службы_kerberos

Если применяется аутентификация Kerberos, то данный параметр указывает имя службы Kerberos5. Значение по умолчанию не определено.

SQLNET.AUTHENTICATION_SERVICES = (метод [,метод...])

Параметр SQLNET.AUTHENTICATION_SERVICES включает Net8-поддержку различных служб, которые могут применяться для аутентификации пользователей, подключающихся к базе данных. Обратите внимание на то, что параметр просто разрешает использование различных методов, но не выбирает метод для конкретного соединения. По умолчанию параметр равен NONE. В качестве *метода* могут выступать следующие ключевые слова:

NONE

Специальная аутентификация не проводится. Регистрация осуществляется по имени пользователя и паролю.

ALL

Разрешает применение всех методов аутентификации.

BEQ

Разрешает применение метода аутентификации BEQ (Bequeath). Параметр не поддерживается в Oracle9*i*.

CYBERSAFE

Разрешает аутентификацию пользователей с помощью CyberSafe. Данный параметр действует только для компонента Oracle Advanced Security.

DCEGSSAPI

Разрешает аутентификацию пользователей с помощью DCE GSSAPI. Данный параметр действует только для компонента Oracle Advanced Security.

IDENTIX

Разрешает аутентификацию пользователей с помощью Identix. Параметр не поддерживается в Oracle9*i*.

KERBEROS5

Разрешает аутентификацию пользователей с применением Kerberos. Данный параметр действует только для компонента Oracle Advanced Security.

NDS

Разрешает аутентификацию пользователей с применением Netware Directory Services. Параметр не поддерживается в Oracle9*i*.

NTS

Разрешает аутентификацию пользователей с применением Windows NT Native Security.

RADIUS

Разрешает аутентификацию пользователей с помощью RADIUS.

SECURID

Разрешает аутентификацию пользователей с помощью SecureID. Параметр не поддерживается в Oracle9*i*.

TCPS

Разрешает аутентификацию пользователей с применением SSL. Параметр не поддерживается в Oracle9*i*.

SQLNET.CLIENT_REGISTRATION = идентификатор_клиента

Позволяет определить уникальный идентификатор Oracle Net Services для клиентского компьютера. Этот идентификатор передается прослушивающему процессу Oracle Net Services в момент подключения клиента и записывается в журнал аудита Oracle Net Services на сервере. Значением параметра может быть любая буквенно-цифровая строка длиной до 128 символов. Значение по умолчанию не определено.

SQLNET.CRYPTO_CHECKSUM_CLIENT = использование_контрольной_суммы

Определяет, каким образом клиент Oracle Net Services договаривается с сервером об использовании контрольных сумм при осуществлении нового соединения. Если клиент и сервер не могут договориться об использовании контрольных сумм, то установить соединение не удается (см. также описание следующего параметра – SQLNET.CRYPTO_CHECKSUM_SERVER). Значение по умолчанию равно REJECTED в Oracle9*i* и ACCEPTED в предыдущих версиях. Возможны следующие значения параметра:

ACCEPTED

Клиент не требует вычисления контрольных сумм, но соглашается, если того требует сервер. Совместим с серверными параметрами REJECTED, REQUESTED и REQUIRED.

REJECTED

Клиент вообще не поддерживает применение контрольных сумм. Совместим с серверными параметрами REJECTED, ACCEPTED и REQUESTED.

REQUESTED

Клиент предпочитает, чтобы контрольные суммы вычислялись, но не настаивает на этом, если сервер отвергает такую работу. Совместим с серверными параметрами ACCEPTED, REQUESTED и REQUIRED.

REQUIRED

Клиент требует применения контрольных сумм и не устанавливает соединение в противном случае. Совместим с серверными параметрами ACCEPTED, REQUESTED и REQUIRED.

SQLNET.CRYPTO_CHECKSUM_SERVER = метод_применения_контрольных_сумм

Определяет, каким образом сервер Oracle Net Services договаривается с клиентом о применении контрольных сумм при установке нового соединения. Если клиент и сервер не могут договориться о применении контрольных сумм, то установить соединение не удается (см. описание предыдущего параметра SQLNET.CRYPTO_CHECKSUM_CLIENT). По умолчанию равен REJECTED в Oracle9*i* и ACCEPTED в предыдущих версиях. Ключевые слова повторяют описанные для параметра SQLNET.CRYPTO_CHECKSUM_CLIENT.

SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT = (MD5 | SHA1)

Задает список алгоритмов вычисления контрольной суммы, которые может применять клиент. Параметр действует для всех клиентских соединений Oracle Net Services, осуществляемых с определенного компьютера. Для установки соединения один из методов вычисления контрольной суммы, поддерживаемый сервером (см. следующий параметр SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER), должен совпадать с одним из методов вычисления контрольной суммы, поддерживаемых клиентом. Значением по умолчанию (и единственным разрешенным значением до появления Oracle9*i*) является MD5 (для алгоритма RSA Data Security – MD5);

в Oracle9*i* добавилось значение SHAL (для алгоритма безопасного хеширования). Значение MD5 ссылается на алгоритм MD5 корпорации RSA Data Security.

SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER = (MD5 | SHAL)

Задает список алгоритмов вычисления контрольной суммы, которые может использовать сервер. Для установки соединения необходимо, чтобы клиент и сервер договорились о применении общего метода вычисления контрольной суммы. Значение по умолчанию (и единственное разрешенное значение до появления Oracle9*i*) равно MD5 (для алгоритма RSA Data Security – MD5); в Oracle9*i* добавилось значение SHAL (для алгоритма безопасного хеширования). Значение MD5 ссылается на алгоритм MD5 корпорации RSA Data Security.

SQLNET.CRYPTO_SEED = "исходная_строка"

Определяет символьную строку, задаваемую при генерировании криптографических ключей. Строка может представлять собой любую последовательность символов длиной от 10 до 70 знаков.

SQLNET.ENCRYPTION_CLIENT = шифрование

Определяет, каким образом клиент Oracle Net Services договаривается с сервером о применении шифрования при установке нового соединения. Если клиент и сервер не могут договориться, то установить соединение не удается (см. также описание следующего параметра SQLNET.ENCRYPTION_SERVER). По умолчанию равен REJECTED в Oracle9*i* и ACCEPTED в предыдущих версиях. Ключевые слова совпадают с описанным для параметра SQLNET.CRYPTO_CHECKSUM_ CLIENT.

SQLNET.ENCRYPTION_SERVER = шифрование

Определяет, каким образом сервер Oracle Net Services договаривается с клиентом о применении шифрования при установке нового соединения. Если клиент и сервер не могут договориться, то установить соединение не удается (см. также описание предыдущего параметра SQLNET.ENCRYPTION CLIENT). По умолчанию устанавливается REJECTED в Oracle9*i* и в ACCEPTED в предыдущих версиях. Ключевые слова совпадают с описанным для параметра SQLNET.CRYPTO_CHECKSUM_CLIENT.

SQLNET.ENCRYPTION_TYPES_CLIENT = (метод [,метод...])

Параметр определяет методы шифрования, один из которых клиент может выбрать при инициировании зашифрованного соединения Oracle Net Services. По умолчанию разрешено выбирать из всех возможных алгоритмов. Ключи размером более 40 бит разрешены только в версии Oracle Net Services для США и Канады. Один из указанных типов шифрования должен совпадать с типом, определенным в параметре SQLNET.ENCRYPTION_TYPES_SERVER для осуществляемого зашифрованного соединения. Значениями параметра может быть одно или несколько из приведенных ключевых слов:

RC4_40

40-битное шифрование RSA RC4.

RC4_56

56-битное шифрование RSA RC4.

RC4_128

128-битное шифрование RSA RC4.

RC4_256

256-битное шифрование RSA RC4 (только для Oracle9*i*).

DES

Стандартное 56-битное шифрование DES.

DES40

40- битное шифрование DES.

3DES_112

Тройное шифрование DES с двумя ключами (112 бита) (только для Oracle9*i*).

3DES_168

Тройное шифрование DES с тремя ключами (168 бит) (только для Oracle9*i*).

SQLNET.ENCRYPTION_TYPES_SERVER = (метод [,метод...])

Параметр указывает методы шифрования, которые может применять сервер при приеме зашифрованного сеанса Oracle Net Services от клиента. По умолчанию разрешено выбирать из всех возможных алгоритмов. Ключи размером более 40 бит разрешены только в версии Oracle Net Services для США и Канады. Возможные значения совпадают с перечисленными для параметра SQLNET.ENCRYPTION_TYPES_CLIENT.

SQLNET.EXPIRE_TIME = период_времени

Устанавливает время жизни сеансов Oracle Net Services. Измеряется в секундах для Oracle9*i* и в минутах для более ранних версий. Отсчет времени начинается в момент открытия соединения. Через указанный интервал времени Oracle Net Services проверяет, активно ли еще соединение. В случае отрицательного ответа сеанс Oracle Net Services завершается.

SQLNET.IDENTIX_FINGERPRINT_DATABASE = имя_сетевой_службы

В случае применения дактилоскопической аутентификации Identix данный параметр указывает имя сетевой службы для базы данных, содержащей отпечатки пальцев. Параметр не поддерживается в Oracle9*i*.

SQLNET.IDENTIX_FINGERPRINT_DATABASE_METHOD = ORACLE

Указывает тип базы данных для хранения дактилоскопической информации. В настоящее время поддерживается только один метод – ORACLE. Параметр не поддерживается в Oracle9*i*.

SQLNET.IDENTIX_FINGERPRINT_DATABASE_PASSWORD = пароль

Определяет пароль, вводимый при подключении к дактилоскопической базе данных для проверки отпечатка. Параметр не поддерживается в Oracle9*i*.

SQLNET.IDENTIX_FINGERPRINT_DATABASE_USER = имя_пользователя

Определяет имя пользователя, вводимое при подключении к дактилоскопической базе данных для проверки отпечатка. Параметр не поддерживается в Oracle9*i*.

SQLNET.KERBEROS5_CC_NAME = путь_и_имя_файла

Указывает полный путь и имя файла кэша удостоверений Kerberos. Параметр действует, только если применяется аутентификация Kerberos (см. описание параметра SQLNET.AUTHENTICATION_SERVICES). В Unix по умолчанию задается значение */usr/tmp/krbcache*, а при работе в Windows – *C:\temp\krbcache*.

SQLNET.KERBEROS5_CLOCKSKEW = секунды

Указывает количество секунд, в течение которых удостоверение Kerberos считается действительным. По истечении указанного времени удостоверение теряет силу. Значение по умолчанию – 300 секунд (т. е. 5 минут).

SQLNET.KERBEROS5_CONF = путь_и_имя_файла

Указывает полный путь и имя файла конфигурации Kerberos. В Unix по умолчанию задается значение */krb5/krb.conf*, при работе в Windows – *C:\krb5\krb.conf*.

SQLNET.KERBEROS5_KEYTAB = путь_и_имя_файла

Указывает полный путь и имя файла соответствия секретных ключей Kerberos. При работе в Unix по умолчанию задается значение */etc/v5srvtab*, а в Windows – *C:\krb5\v5srvtab*.

SQLNET.KERBEROS5_REALMS = путь_и_имя_файла

Указывает полный путь и имя файла преобразования областей Kerberos. При работе в Unix по умолчанию устанавливается значение */krb5/krb.realms*, а в Windows – *C:\krb\krb.realms*.

SQLNET.RADIUS_ALTERNATE = имя_хоста

Указывает имя хоста или числовой адрес (в зависимости от предпочтений пользователя) запасного сервера RADIUS, который будет задействован в случае недоступности основного сервера RADIUS. Значения по умолчанию не существует. Поддержка аутентификации RADIUS появилась в Oracle8*i*.

SQLNET.RADIUS_ALTERNATE_PORT = номер_порта

Определяет номер порта, указываемый при обращении к запасному серверу RADIUS. Номер порта по умолчанию равен 1645. Поддержка аутентификации RADIUS появилась в Oracle8*i*.

SQLNET.RADIUS_ALTERNATE_RETRIES = количество_попыток

Задает количество попыток установления соединения с запасным сервером RADIUS. По умолчанию дается 3 попытки. Поддержка аутентификации RADIUS появилась в Oracle8*i*.

SQLNET.RADIUS_AUTHENTICATION = имя_хоста

Указывает имя хоста или числовой адрес (в зависимости от предпочтений пользователя) основного сервера RADIUS. Значение по умолчанию – локальный хост. Поддержка аутентификации RADIUS появилась в Oracle8*i*.

SQLNET.RADIUS_AUTHENTICATION_INTERFACE = класс_интерфейса

Указывает класс Java, который определяет интерфейс для взаимодействия с пользователем в случае, если RADIUS работает в *режиме аутентификации с запросом и подтверждением* (*challenge-response mode*). Классом по умолчанию является DefaultRadiusInterface (см. также описание параметра SQLNET.RADIUS_CLASSPATH). Поддержка аутентификации RADIUS появилась в Oracle8*i*.

SQLNET.RADIUS_AUTHENTICATION_PORT = номер_порта

Определяет номер порта, указываемый при обращении к основному серверу RADIUS. Номер порта по умолчанию равен 1645. Поддержка аутентификации RADIUS появилась в Oracle8*i*.

SQLNET.RADIUS_AUTHENTICATION_RETRIES = количество_попыток

Указывает количество попыток установления соединения с основным сервером RADIUS. По умолчанию дается 3 попытки. Поддержка аутентификации RADIUS появилась в Oracle8*i*.

SQLNET.RADIUS_AUTHENTICATION_TIMEOUT = секунды

Задает период времени (в секундах), в течение которого следует ждать ответа при попытке соединения с сервером RADIUS. Значение по умолчанию – 5 секунд. Поддержка аутентификации RADIUS появилась в Oracle8*i*.

SQLNET.RADIUS_CHALLENGE_KEYWORD = ключевое_слово

Задает ключевое слово, которое используется для запроса отклика от сервера RADIUS. Ключевое слово по умолчанию – «challenge». Поддержка аутентификации RADIUS появилась в Oracle8*i*. Параметр не поддерживается в Oracle9*i*.

*SQLNET.RADIUS_CHALLENGE_RESPONSE = ON | **OFF***

Включает и выключает аутентификацию RADIUS с запросом и подтверждением. По умолчанию выключена. Поддержка аутентификации RADIUS появилась в Oracle8*i*.

SQLNET.RADIUS_CLASSPATH = путь_класса

Указывает путь к классам Java, реализующим интерфейс запроса/подтверждения (см. также описание параметра SQLNET.RADIUS_AUTHENTICATION_INTERFACE). Значение по умолчанию не определено. Поддержка аутентификации RADIUS появилась в Oracle8*i*. Параметр не поддерживается в Oracle9*i*.

SQLNET.RADIUS_SECRET = путь_и_имя_файла

Указывает полный путь и имя разделяемого секретного файла RADIUS. По умолчанию это файл *$ORACLE_HOME/network/security/radius.key*. Поддержка аутентификации RADIUS появилась в Oracle8*i*.

*SQLNET.RADIUS_SEND_ACCOUNTING = ON | **OFF***

Включает и выключает учет расходования ресурсов RADIUS. Если учет расходования ресурсов включен, то пакеты отсылаются на сервер RADIUS через порт с номером, на единицу превышающим номер порта, указанного в параметре SQLNET.RADIUS_AUTHENTICATION_PORT. Другими словами, если значение параметра SQLNET.RADIUS_AUTHENTICATION_PORT равно 1645, то при включении задействуется порт 1646. По умолчанию установлено значение OFF, т. е. учет расходования ресурсов RADIUS отключен. Появился в Oracle8*i*.

SSL_CIPHER_SUITES = (имя_комплекта [,имя_комплекта...])

Указывает список шифрокомплектов SSL, которые будут поддерживаться. Список возможных имен комплектов шифров приведен в документации Oracle.

*SSL_CLIENT_AUTHENTICATION = **TRUE** | FALSE*

Определяет, должен ли клиент аутентифицироваться посредством SSL. По умолчанию задано значение TRUE, но оно действует лишь в случае применения аутентификации SSL (см. описание параметра SQLNET.AUTHENTICATION_SERVICES).

*SSL_SERVER_DN_MATCH = {YES | ON | TRUE} | {**NO** | OFF | FALSE}*

Требует, чтобы отличительное имя (Distinguished Name) сервера базы данных соответствовало имени службы. Значение по умолчанию – NO. Появился в Oracle9*i*.

*SSL_VERSION = **UNDETERMINED** | 2.0 | 3.0*

Указывает, какая версия SSL должна применяться для SSL-зашифрованного соединения Oracle Net Services. По умолчанию установлено значение UNDETERMINED, т. е. версия определяется клиентом и сервером в момент установки соединения.

TCP.EXCLUDED_NODES = (имя_хоста | ip_адрес[, имя_хоста | ip_адрес] . .)

Указывает, для каких клиентов доступ к базе данных запрещен. Параметр появился в Oracle9*i* и предназначен для замены функциональности, относившейся к более не поддерживаемому файлу *protocol.ora*.

TCP.INVITED_NODES = (имя_хоста | ip_адрес[, имя_хоста | ip_адрес] . .)

Указывает, для каких клиентов доступ к базе данных разрешен. Если задан и этот параметр, и TCP.EXCLUDED_NODES, то более высокий приоритет имеет пара-

метр TCP.INVITED_NODES. Параметр появился в Oracle9*i* и предназначен для замены функциональности, относившейся к более не поддерживаемому файлу *protocol.ora*.

*TCP.NODELAY = YES | **NO***

Определяет, следует ли исключить задержку при сбросе буфера стека протоколов TCP/IP. Значение по умолчанию – NO. Параметр появился в Oracle9*i* и предназначен для замены функциональности, относившейся к более не поддерживаемому файлу *protocol.ora*.

TCP.VALIDNODE_CHECKING=YES | NO

Указывает, следует ли применять параметры TCP.EXCLUDED_NODES и TCP.INVITED_NODES для контроля клиентского доступа. Значение по умолчанию – NO. Параметр появился в Oracle9*i* и предназначен для замены функциональности, относившейся к более не поддерживаемому файлу *protocol.ora*.

TNSPING.TRACE_DIRECTORY = путь_к_каталогу

Указывает каталог для файлов трассировки *tnsping*. По умолчанию это каталог *$ORACLE_HOME/network/trace*.

*TNSPING.TRACE_LEVEL = **OFF** | USER | ADMIN | SUPPORT*

Определяет уровень детализации трассировочной информации, генерируемой утилитой *tnsping*. Значение по умолчанию – OFF. Значения ключевых слов приведены в описании параметра DAEMON.TRACE_LEVEL.

TRACE_DIRECTORY_CLIENT = путь_к_каталогу

Задает каталог, в который будут записываться файлы трассировки Oracle Net Services в случае, если Oracle Net Services действует как клиент. По умолчанию это каталог *$ORACLE_HOME/network/trace*.

TRACE_DIRECTORY_SERVER = путь_к_каталогу

Задает каталог, в который будут записываться файлы трассировки Oracle Net Services в случае, если Oracle Net Services действует как сервер. По умолчанию это каталог *$ORACLE_HOME/network/trace*.

TRACE_FILE_CLIENT = имя_файла

Определяет имя файла, которое будет использоваться клиентскими файлами трассировки. Имя трассировочного файла по умолчанию – *sqlnet.trc* или в некоторых случаях – *cli.trc*. В имена клиентских файлов может (необязательно) добавляться идентификатор процесса (см. описание параметра TRACE_UNIQUE_CLIENT).

TRACE_FILE_SERVER = имя_файла

Определяет имя файла, которое будет использоваться серверными файлами трассировки. Имя трассировочного файла по умолчанию – *svr_pid.trc*, где *pid* – идентификатор процесса.

TRACE_FILELEN_CLIENT = килобайты

Задает размер клиентского файла трассировки. Появился в Oracle9*i*.

TRACE_FILELEN_SERVER = килобайты

Задает размер серверного файла трассировки. Появился в Oracle9*i*.

TRACE_FILENO_CLIENT = количество

Указывает количество клиентских файлов трассировки. Если данный параметр задается в сочетании с параметром TRACE_FILELEN_CLIENT, то файлы трассировки используются циклически. Появился в Oracle9*i*.

TRACE_FILENO_server = количество

Указывает количество серверных файлов трассировки. Если данный параметр задается в сочетании с параметром TRACE_FILELEN_SERVER, то файлы трассировки используются циклически. Появился в Oracle9*i*.

*TRACE_LEVEL_CLIENT = **OFF** | USER | ADMIN | SUPPORT*

Включает трассировку Oracle Net Services на стороне клиента и определяет уровень детализации информации, записываемой в файл трассировки. Пояснения значений ключевых слов приведены в описании параметра DAEMON.TRACE_LEVEL.

*TRACE_LEVEL_SERVER = **OFF** | USER | ADMIN | SUPPORT*

Включает трассировку Oracle Net Services на стороне сервера и определяет уровень детализации данных, записываемых в файл трассировки. Пояснения значений ключевых слов приведены в описании параметра DAEMON.TRACE_LEVEL.

*TRACE_TIMESTAMP_CLIENT = **ON** | OFF*

Указывает, следует ли добавлять временную метку к каждому клиентскому событию трассировки. Значение по умолчанию – ON. Появился в Oracle9*i*.

*TRACE_TIMESTAMP_SERVER = **ON** | OFF*

Указывает, следует ли добавлять временную метку к каждому серверному событию трассировки. Значение по умолчанию – ON. Появился в Oracle9*i*.

*TRACE_UNIQUE_CLIENT = ON | **OFF***

Определяет, должны ли клиентские файлы трассировки иметь уникальные имена. Уникальность достигается за счет дополнения имени клиентского трассировочного файла идентификатором процесса или потока. Значение ON обеспечивает уникальность. По умолчанию равен OFF, т. е. идентификаторы процесса и потока не входят в имя файла (см. также описание параметра TRACE_FILE_CLIENT).

*USE_CMAN = TRUE | **FALSE***

Позволяет принудительно проводить все сеансы Oracle Net Services через диспетчер соединений Oracle. Значение по умолчанию – FALSE. Если параметр равен TRUE, то для того чтобы попасть на сервер, соединения Oracle Net Services автоматически пропускаются через диспетчер соединений.

*USE_DEDICATED_SERVER = **OFF** | ON*

Позволяет принудительно включить использование выделенных серверных процессов для всех соединений определенного клиента. Значение по умолчанию – OFF. Если параметр установлен в ON, то любое соединение с удаленной службой Oracle Net Services приводит к порождению нового выделенного серверного процесса.

WALLET_LOCATION

Задается при работе с компонентом Oracle Advanced Security для указания местоположения *накопителей* (*wallets*). Появился в Oracle9*i*. Дополнительную информацию об этом параметре и о применении накопителей можно найти в документации Oracle.

TNSNAMES.ORA

TNSNAMES.ORA – это файл конфигурации, в котором сетевое имя службы сопоставлено адресам и сетевым протоколам, применяемым при подключении к такой службе. Файл используется для *локального именования* (*local naming*): локальный файл идентифицирует и конфигурирует соединение с прослушивающим процессом.

Для удобства восприятия описание синтаксиса файла *TNSNAMES.ORA* будет поделено на части, по одной для каждой секции файла. После разделов с описанием синтаксиса будет предложен пример файла.

Синтаксис

В общем виде конструкция файла такова:

```
имя службы =
    (описательная_информация
        (информация_о_соединении)
    )
```

где *имя_службы* – это имя сетевой службы, *описательная_информация* – фраза, содержащая характеристики службы, а *информация_о_соединении* – фраза, содержащая данные о соединениях для данной службы.

Имя_службы должно состоять из имени, которое Oracle Net Services будет использовать для соединения, точки (.) и имени домена Oracle Net Services.

Файл *TNSNAMES.ORA* может включать в себя несколько записей, каждая из которых должна иметь такую форму. Обратите внимание, что наличие скобок необходимо. Приведенное форматирование не является обязательным, но большинство пользователей Oracle считают его наиболее удобочитаемым, т. к. в одной записи может присутствовать несколько пар скобок.

описательная_информация

```
DESCRIPTION_LIST=
    [(DESCRIPTION=
        [(ADDRESS LIST=]
            (ADDRESS=адресная информация)
            [(ADDRESS=адресная информация). . .]
        (параметры_адреса)
        [)]
    [(параметр_списка_описаний
    [) (параметр_списка_описаний])]
    [(параметр_описания
    [) (параметр_описания])]
    [)]
```

Применяется для задания информации о соединении. Для одной службы файл может содержать несколько секций *описательная_информация*. Если клиент хочет осуществить соединение со службой, то Oracle Net Services пытается подключиться, используя информацию из первой секции описания. В случае неудачи Oracle Net Services перейдет ко второй секции и т. д.

Ключевые слова и секции

DESCRIPTION LIST

Необязательное ключевое слово, употребляемое при необходимости указать один или несколько параметров описания. Если ключевое слово опускается, то не следует ставить и закрывающую скобку. Если параметр задан, то по умолчанию включается выравнивание клиентской нагрузки.

параметр_описания

Если задан параметр DESCRIPTION LIST, то можно указывать параметры описания, а именно:

*FAILOVER=({**ON** | *YES* | *TRUE*} | {*OFF* | *NO* | *FALSE*})*

Включает и выключает переключение при сбое в момент соединения для описаний из списка. По умолчанию переключение при сбое установлено (ON). Если переключение при сбое включено, то неудачная попытка соединения через одно описание приводит к тому, что Oracle Net Services пробует другое описание из списка, перебирая адреса последовательно, если только не применяется выравнивание нагрузки. Появился в Oracle8*i*.

*LOAD_BALANCE=({**ON** | *YES* | *TRUE*} | {*OFF* | *NO* | *FALSE*})*

Включает и выключает выравнивание нагрузки для клиента. Если задан параметр DESCRIPTION LIST, то выравнивание клиентской нагрузки по умолчанию включено. При осуществлении соединения Oracle Net Services случайным образом выбирает описание из списка. Если включено и переключение при сбое, то Oracle Net Services произвольно переходит от одного описания к другому до тех пор, пока не переберет все описания, или пока не будет установлено соединение. Появился в Oracle8*i*.

*SOURCE_ROUTE=({ON | YES} | {**OFF** | NO})*

Применяется для маршрутизации соединений через диспетчер соединений. Значение по умолчанию – OFF. Если маршрутизация включена, то DESCRIPTION LIST должен содержать два или более описаний. Первое описание должно указывать на экземпляр диспетчера соединений. Тогда Oracle Net Services осуществляет соединение, переходя от одного описания к другому до тех пор, пока не достигнет прослушивающего процесса базы данных. Появился в Oracle8*i*.

Каждый параметр описания должен быть заключен в скобки. Обычно каждый параметр занимает отдельную строку.

параметр_списка_описаний

Те же значения, что и для *параметра_описания*, за исключением FAILOVER и LOAD_BALANCE.

ADDRESS LIST

Необязательное ключевое слово, появившееся в Oracle8*i*. Если ключевое слово опускается, не следует ставить и закрывающую скобку. Запись может иметь несколько списков адресов. В этом случае Oracle Net Services выбирает произвольным образом один адрес из первого списка. В случае неудачи Oracle Net Services переходит ко второму списку и выбирает там произвольный адрес и так далее.

адресная информация

Указывает тип протокола, используемого для подключения к прослушивателю и определяет конфигурационную информацию для протокола. Перечислим поддерживаемые протоколы и приведем необходимую для каждого из них информацию:

BEQUEATH – не поддерживается в Oracle9i

```
(PROTOCOL=BEQ)(PROGRAM=имя_программы)
(ARGV0=имя_экземпляра_программы)(
        ARGS='(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))')
```

TCP/IP

```
(PROTOCOL = TCP)(HOST={имя_хоста | IP_адрес})
(PORT = номер_порта)
```

TPC/IP с SSL – появился в Oracle8i

```
(PROTOCOL = TCPS) (HOST = { имя_хоста | IP_адрес})
(PORT = номер_порта)
```

IPC

```
(PROTOCOL = IPC)(KEY = имя_ключа)
```

Named Pipes (Именованные каналы)

```
(PROTOCOL = NMP)(SERVER = имя_сервера)(PIPE = имя_канала)
```

SPX – не поддерживается в Oracle9i

```
(PROTOCOL = SPX)(SERVICE = имя_службы_spx)
```

LU6.2 – не поддерживается в Oracle9i

```
(PROTOCOL=LU6.2)(LU_NAME = уточненное_имя_сервера)
    (LLU_NAME | LOCAL_LLU_NAME = уточненный_локальный_псевдоним_LU)
    (MODE | MDN = режим_журнала)
    (PLU_LA | PARTNER_LU_LOCAL_ALIAS = LU_псевдоним партнера)
    (TP_NAME | TPN = программа_транзакции_хоста)
```

параметры_адреса

Большинство параметров такие же, как для *параметров_описания* (не действуют только FAILOVER и LOAD_BALANCE), а также следующие параметры:

SDU = число

Указывает размер пакетов данных, отправляемых по сети. Для наибольшей производительности SDU должен быть кратен размеру кадра протокола нижнего уровня, который можно установить при помощи параметра TDU. Параметр не может быть установлен для базы данных, динамически регистрирующейся при помощи прослушивающего процесса и выделенного серверного соединения. Значение по умолчанию – 2048 байт. Максимальное разрешенное значение – 32 768 байт (32K).

TYPE_OF_SERVICE

Указывает, устанавливается ли соединение с базой данных Oracle (ORACLE8_DATABASE, это значение по умолчанию) или же с базой данных Oracle Rdb (RDB_DATABASE). Параметр следует указывает только при подключении к базе данных Oracle Rdb.

информация о соединении

```
(CONNECT_DATA=
    [(параметр_соединения)]
    [(параметр_соединения)] . . .
)
```

Предоставляет информацию о соединении. Каждой службе соответствует секция с информацией о соединении.

Ключевые слова и инструкции

Приведем перечень параметров соединения (каждый из них должен быть заключен в скобки):

FAILOVER_MODE = инструкции_по_восстановлению

Применяется для технологии Transparent Application Failover (TAF). В качестве *инструкции_по_восстановлению* может выступать один или несколько перечисленных ниже параметров, при этом каждый из них должен быть заключен в скобки:

BACKUP = имя_резервной_сетевой_службы

Указывает имя сетевой службы для резервного соединения.

TYPE = SESSION | SELECT | NONE

Определяет, что будет делать Oracle Net Services в случае сбоя основного экземпляра. Значение SESSION означает открытие сеанса с резервным экземпляром. SELECT устанавливает сеанс с резервным экземпляром, заново открывает курсоры для ожидающих выполнения команд SELECT и устанавливает эти курсоры в такое положение, чтобы выборка продолжалась с того места, в котором клиент находился в момент сбоя основного экземпляра. NONE означает полное отсутствие переключения при сбое и дает возможность явно указать поведение по умолчанию.

METHOD = BASIC | PRECONNECT

BASIC устанавливает резервное соединение только в случае сбоя соединения с основным экземпляром. PRECONNECT устанавливает соединение с резервным экземпляром в момент совершения первого соединения.

RETRIES = количество_повторных_попыток

Указывает количество попыток установки соединения с резервным экземпляром в случае сбоя основного. По умолчанию обычно задается значение 0, но если задан параметр DELAY, то количество попыток по умолчанию устанавливается равным 5. Появился в Oracle8*i*.

DELAY = интервал_между_попытками

Указывает время ожидания между повторными попытками (в секундах). Интервал по умолчанию – 1 секунда. Появился в Oracle8*i*.

GLOBAL_NAME = глобальное_имя_базы_данных

Идентифицирует базы данных Oracle Rdb. Указывайте этот параметр только при подключении к БД Oracle Rdb.

HS = OK

Сообщает Oracle Net Services о том, что выполняется подключение к службе, которая не является базой данных Oracle или Oracle Rdb (HS – Heterogeneous Services, или гетерогенные сервисы).

INSTANCE_NAME = имя_экземпляра

Определяет конкретный экземпляр Oracle, к которому вы хотите подключиться. Это удобно, когда служба имеет несколько экземпляров, а подключиться надо к какому-то определенному. Имя экземпляра всегда должно указываться в сочетании с именем службы. Появился в Oracle8*i*.

RDB_DATABASE = имя_файла_rdb

Идентифицирует базу данных Oracle Rdb по имени файла. В имя файла можно включить путь и расширение.

SDU=число

Аналогичен параметру SDU из адресной секции.

SERVER = DEDICATED | SHARED

Определяет, устанавливается ли соединение с выделенным или же разделяемым серверным процессом. Выбирайте SHARED, только если сервер сконфигурирован для работы с разделяемыми серверами. Ключевое слово DEDICATED указывается для того, чтобы указать, что требуется установить выделенное соединение с базой данных, настроенной на работу с разделяемым сервером.

SERVICE_NAME = служба_базы данных

Определяет службу базы данных Oracle. Указанное имя должно совпадать с именем службы, записанным в параметре SERVICE_NAMES экземпляра базы данных, или же с глобальным именем базы данных.

SID = системный_идентификатор

Позволяет идентифицировать экземпляр по системному идентификатору. SID может применяться как вместо SERVICE_NAME, так и вместе с SERVICE_ NAME, если необходимо поддерживать средствами одного файла *TNSNAMES.ORA* как клиентов, работающих с Oracle8*i* и более поздними версиями, так и пользователей ранних версий сетевого программного обеспечения Oracle.

Пример

Приведем пример файла *TNSNAMES.ORA*, описывающего следующую ситуацию:

- Имя сетевой службы – Bailey в домене *greenie.org*.
- Применяется протокол TCP.
- В домене присутствуют хосты TREATS, BAGELS и FRIES.
- Для этих трех хостов действует выравнивание нагрузки.
- Для сетевой службы CHUCK в этом же домене действует переключение при сбое.
- Соединение для переключения на случай сбоя устанавливается в момент совершения основного соединения:

```
BAILEY.GREENIE.ORG =
(DESCRIPTION =
    (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL=TCP)(HOST=TREATS.GREENIE. ORG)(PORT=1521))
       (ADDRESS=(PROTOCOL=TCP)(HOST=BAGELS.GREENIE. ORG)(PORT=1521))
       (ADDRESS=(PROTOCOL=TCP)(HOST=FRIES.GREENIE. ORG)(PORT=1521))
       (LOAD_BALANCE=ON)
       (FAILOVER=ON)
    )
    (CONNECT_DATA =
        (SERVICE_NAME=CHUCK.GREENIE.ORG)
        (FAILOVER_MODE=
            (TYPE=SESSION)
            (METHOD=PRECONNECT))
    )
)
```

LDAP.ORA

LDAP.ORA – это файл конфигурации на клиентской машине, предназначенный для указания сервера LDAP, который будет применяться для определения имен сетевых служб. *LDAP.ORA* используется, если вы храните конфигурацию сетевых служб в LDAP-каталоге, а не применяете локальный метод, используя *TNSNAMES.ORA*.

Основная часть сетевой информации хранится в LDAP-каталоге, поэтому файл *LDAP.ORA* весьма невелик. Этот файл обычно создается утилитой Oracle Net Configuration Assistant, и корпорация Oracle не рекомендует вносить в него изменения.

Ключевые слова и инструкции

DIRECTORY_SERVERS = хост:порт[:ssl_порт][, хост:порт[:ssl_порт] . . .]

Перечисляет имена хостов, порты и SSL-порты для основного и резервного LDAP-серверов.

DIRECTORY_SERVER_TYPE = идентификатор_типа

Определяет тип LDAP-сервера. В версии Oracle9*i* единственными поддерживаемыми значениями для *идентификатора_типа* являются «*id*» для Oracle Internet Directory и «*ad*» для Microsoft Active Directory, который не поддерживался до выхода Oracle9*i*.

DEFAULT_ADMIN_CONTEXT = отличительное_имя

Указывает запись каталога, которая содержит контекст Oracle, из которого можно создавать, изменять или просматривать идентификаторы соединений.

LISTENER.ORA

LISTENER.ORA – это файл конфигурации, используемый для управления работой прослушивающего процесса Oracle на сервере. В одном файле *LISTENER.ORA* может быть определена конфигурация нескольких прослушивающих процессов, если каждый из них обладает уникальным именем.

Синтаксис *LISTENER.ORA* аналогичен синтаксису файла *TNSNAMES.ORA*.

Синтаксис

```
LISTENER=
    (DESCRIPTION_LIST
       (DESCRIPTION=
        (ADDRESS LIST=
            (ADDRESS=адресная_информация)
            [(ADDRESS= адресная_информация). . .]
            [(PROTOCOL_STACK=
            (PRESENTATION=GIOP)
            (SESSION=RAW))]
    )
SID_LIST_LISTENER=
    (SID_LIST=
        (SID_DESC=
            (параметры_sid)
            [(параметры_sid) . . .]
        )
        [(SID_DESC=
            (параметры_sid)
            [(параметры_sid) . . .]
        ) . . .]
(управляющие параметры)
[(управляющие параметры). . .]
```

Ключевые слова и инструкции

адресная информация

Синтаксически не отличается от *адресной_информации* в описанном ранее файле *TNSNAMES.ORA*. В дополнение к ключевым словам, указанным для каждого протокола, в *LISTENER.ORA* для TCP/IP также разрешено:

QUEUESIZE = число

Указывает количество параллельных запросов на соединение, которое прослушивающий процесс может принимать в точке прослушивания TCP/IP. Данное ключевое слово должно быть последним в разделе *адресная_информация*. Значение по умолчанию равно 5 для Sun SPARC Solaris и Windows NT 4.0 Workstation, а для Windows NT 4.0 Server значение по умолчанию равно 60.

```
(PROTOCOL_STACK =
  (PRESENTATION = GIOP)
  (SESSION = RAW))
```

Применяется для соединения по протоколу IIOP (Internet Intra-Orb Protocol), определяющему передачу сообщений между сетевыми объектами по TCP/IP, с Java-объектами базы данных. Параметр появился в Oracle8*i*.

SID_LIST_LISTENER

В Oracle8*i* и Oracle9*i* экземпляры при запуске автоматически регистрируют себя для своих прослушивающих процессов. При работе с Oracle8 и более ранними версиями, а также при вызове внешних процедур или гетерогенных сервисов, необходимо применять данную инструкцию для статического конфигурирования служб для прослушивающего процесса.

параметры_sid

Для статически конфигурируемого SID (Oracle system identifier), или системного идентификатора Oracle, можно задавать следующие параметры, при этом каждый из них должен быть заключен в скобки:

ENVS = "переменная = значение[*,переменная = значение . . .*]"

Позволяет задавать переменные окружения для прослушивающего процесса; они устанавливаются до выполнения выделенной серверной программы или программы, для которой задан параметр PROGRAM. Данный параметр не работает в Windows NT, т. к. каждый процесс, порожденный прослушивающим процессом, наследует его окружение и переменные.

GLOBAL_DBNAME = имя_базы_данных.домен_базы_данных

Задает глобальное имя базы данных, которое состоит из имени базы данных и домена базы данных. Это имя должно совпадать с одним из значений, указанных в параметре SERVICE_NAMES файла *INIT.ORA*.

ORACLE_HOME = путь_домашнего_каталога

Указывает домашний каталог Oracle для службы.

PROGRAM = имя_программы

Указывает имя исполняемой программы службы.

SID_NAME = имя_sid

Определяет имя SID для экземпляра Oracle. Значение должно совпадать со значением параметра INSTANCE_NAME файла *INIT.ORA*.

управляющие параметры

Управляет работой прослушивающего процесса. В списке приведены названия всех управляющих параметров, каждое из которых должно быть дополнено в конце именем прослушивающего процесса:

ADMIN_RESTRICTIONS_имя_прослушивателя = ON | OFF

Запрещает или повторно разрешает изменение параметров LISTENER.*ORA* во время исполнения. В данном параметре нет необходимости, если для обеспече-

ния безопасности прослушивающего процесса применяется пароль. Более подробная информация будет приведена далее в разделе «Утилита Listener Control».

LOG_DIRECTORY_имя_прослушивателя = путь_к_журналу_прослушивателя

Указывает каталог для файла журнала прослушивающего процесса.

LOG_FILE_имя_прослушивателя = имя_файла_журнала

Задает имя файла журнала прослушивающего процесса.

LOGGING_имя_прослушивателя = ON | OFF

Включает и выключает ведение журнала для указанного прослушивающего процесса.

PASSWORDS_имя_прослушивателя = (зашифрованный пароль)

Позволяет указать зашифрованный пароль для прослушивающего процесса. Действует так же, как команда CHANGE_PASSWORD утилиты Listener Control.

SAVE_CONFIG_ON_STOP_имя_прослушивателя = TRUE | FALSE

Определяет, следует ли сохранять изменения, сделанные в файле конфигурации прослушивающего процесса при помощи команд SET утилиты Listener Control.

SSL_CLIENT_AUTHENTICATION = TRUE | FALSE

Указывает, производится ли SSL-аутентификация клиента. Если значение параметра равно TRUE, то прослушивающий процесс пытается аутентифицировать пользователя.

STARTUP_WAIT_TIME_имя_прослушивателя = число

Задает период времени в секундах, в течение которого прослушивающий процесс ждет, прежде чем ответить на команду START утилиты Listener Control.

TRACE_DIRECTORY_имя_прослушивателя = каталог_файла_трассировки

Указывает каталог для файла трассировки прослушивающего процесса.

TRACE_FILE_имя_прослушивателя = имя_файла_трассировки

Указывает имя файла трассировки прослушивающего процесса. Имя файла по умолчанию – *listener.trc*.

TRACE_FILELEN_имя_прослушивателя = число

Определяет размер файлов трассировки прослушивающего процесса (в килобайтах). Когда указанный предельный размер достигнут, трассировочная информация начинает записываться в следующий файл. По умолчанию размер файла не ограничен.

TRACE_FILENO_имя_прослушивателя = число

Указывает количество файлов трассировки прослушивающего процесса. Если параметр указан, то в конец имени файла трассировки добавляется целое число.

TRACE_LEVEL_имя_прослушивателя = OFF | USER | ADMIN | SUPPORT

Определяет уровень детализации информации файла трассировки прослушивающего процесса.

TRACE_TIMESTAMP_имя_прослушивателя = ON | OFF

Указывает, следует ли добавлять в записи файла трассировки временную метку.

WALLET_LOCATION

Используется в сочетании с накопителями для компонента Oracle Advanced Security. Подробные сведения о данном параметре можно найти в документации Oracle.

Утилиты сетевого управления

В данном разделе описаны различные утилиты, применяемые для управления Oracle Net Services. Общая цель этих утилит заключается в упрощении реализации сетевого взаимодействия и управления им. Приведем описания утилит сетевого управления, чаще всего применяемых в Oracle9*i*, при этом для утилит с интерфейсом командной строки будут указаны имена исполняемых файлов:

Утилита Listener Control (lsnrctl)

Утилита с интерфейсом командной строки для управления прослушивающим процессом Oracle. В некоторых операционных системах (например, Windows) прослушивающий процесс можно запустить и автономно как сервис. В системах Unix и Linux команды *lsnrctl* можно встраивать в сценарий запуска.

Утилита Oracle Names Control (namesctl)

Утилита с интерфейсом командной строки для управления Oracle Names. Подробная информация о командах, используемых утилитой Oracle Names Control, приведена в документации Oracle.

Утилита Oracle Connection Manager Control (cmctl)

Утилита с интерфейсом командной строки для управления диспетчером соединений Oracle. Подробная информация о командах, используемых утилитой Oracle Connection Manager, приведена в документации Oracle.

Утилита TNSPing (tnsping)

Утилита с интерфейсом командной строки, применяемая для проверки соединения между клиентом Oracle Net Services и сервером.

Oracle Net Manager

Инструмент управления с графическим интерфейсом пользователя, который позволяет создавать, исследовать и тестировать конфигурации для прослушивающего процесса, соединения и Oracle Names.

Configuration Assistant

Инструмент управления с графическим интерфейсом пользователя, применяемый в процессе установки для конфигурирования имен прослушивающих процессов, адресов протоколов, методов именования, имен сетевых служб и использования каталогов.

Утилита Listener Control

Утилита Listener Control позволяет управлять прослушивающим процессом Oracle. Исполняемый файл утилиты обычно называется *lsnrctl*. Однако имейте в виду, что это имя может зависеть от платформы и версии. В последующих разделах будут рассмотрены команды утилиты Listener Control и их параметры.

Вызов утилиты

Утилита Listener Control обычно вызывается посредством ввода имени программы (обычно это *lsnrctl*) в командной строке операционной системы. Запустить Listener Control можно одной из приведенных ниже команд:

```
lsnrctl [команда] [прослушиватель]
```

или

```
lsnrctl @имя_файла
```

Задаются следующие необязательные параметры:

команда

Любая разрешенная команда Listener Control.

прослушиватель

Имя прослушивающего процесса. Если в команде не указано конкретное имя, то утилита Listener Control будет искать прослушивающий процесс с именем LISTENER.

@имя_файла

Выполняет команды, содержащиеся в указанном файле. Для того чтобы включить в файл комментарии, предварите их знаком (#) или командой REM.

Если утилита Listener Control вызывается без указания команды или файла, то утилита откроет собственный сеанс с приглашением на ввод команд вида `lsnrctl> prompt` .

Команды утилиты Listener Control

Для утилиты Listener Control можно указать следующие команды.

CHANGE_PASSWORD

```
CHANGE_PASSWORD [прослушиватель]
```

Заменяет пароль утилиты Listener Control на зашифрованный пароль. После ввода такой команды Listener Control запрашивает старый пароль пользователя (если он существовал) и новый пароль, не отображая их на экране. Обычно после изменения пароля выполняется команда SAVE_CONFIG.

EXIT

```
EXIT
```

Завершает работу утилиты Listener Control.

HELP

```
HELP [прослушиватель]
```

Выводит список команд утилиты Listener Control.

QUIT

```
QUIT
```

Завершает работу утилиты Listener Control. Идентична команде EXIT.

RELOAD

```
RELOAD [прослушиватель]
```

Перезагружает файл *LISTENER.ORA*, в результате чего применяются параметры, указанные в данном файле конфигурации. Часто выдача команды RELOAD позволяет реализовать изменения без выполнения остановки и перезапуска прослушивающего процесса.

SAVE_CONFIG

```
SAVE_CONFIG [прослушиватель]
```

Сохраняет все изменения конфигурации прослушивающего процесса в файл *LISTENER.ORA*.

SERVICES

```
SERVICES [прослушиватель]
```

Возвращает подробную информацию о службах базы данных, экземплярах, выделенных серверах и диспетчерах, с которыми взаимодействует прослушивающий процесс.

SET

```
SET команда
```

Позволяет установить значения параметров прослушивающего процесса. Если команда вводится без параметра, то она возвращает список параметров, которые могут быть установлены.

Ключевые слова

команда

Один из параметров списка:

CURRENT_LISTENER

Указывает прослушивающий процесс по умолчанию для всех последующих команд утилиты Listener Control.

*DISPLAYMODE = COMPAT | **NORMAL** | VERBOSE | RAW*

Определяет объем информации, отображаемой для команд SERVICE и STATUS:

COMPAT отображает вывод, который совместим со старыми версиями прослушивающего процесса.

VERBOSE отображает всю информацию о прослушивающем процессе.

RAW отображает информацию без форматирования и должен использоваться только по запросу службы технической поддержки Oracle.

LOG_DIRECTORY каталог

Указывает каталог для журнальных файлов утилиты Listener Control.

LOG_FILE имя_файла_журнала

Указывает имя для журнальных файлов утилиты Listener Control. Значение по умолчанию – *listener.log*.

*LOG_STATUS ON | **OFF***

Включает и выключает ведение журнала для утилиты Listener Control.

PASSWORD

Указывает пароль для пользователя. После выдачи такой команды утилита Listener Control запрашивает у пользователя пароль. Команда должна вводиться перед использованием защищенных паролем команд, таких как START, STOP и SAVE_CONFIG.

*SAVE_CONFIG_ON_STOP **ON** | OFF*

Указывает, должны ли все изменения файла конфигурации сохраняться в файле *LISTENER.ORA* при выходе из утилиты Listener Control.

STARTUP_WAITTIME секунды

Указывает период времени (в секундах), в течение которого прослушивающий процесс находится в состоянии ожидания после выдачи ему команды START. Команду не рекомендуется использовать в Oracle9*i*, в последующих версиях она не будет поддерживаться.

TRC_DIRECTORY каталог

Задает каталог файлов трассировки утилиты Listener Control.

TRC_FILE имя_файла

Задает имя файла трассировки утилиты Listener Control.

TRC_LEVEL ***OFF*** | *USER* | *ADMIN* | *SUPPORT*

Определяет уровень детализации трассировочной информации.

SHOW

```
SHOW [параметр]
```

Отображает значение параметра, а если параметр не указан – всех параметров.

SPAWN

```
SPAWN [прослушиватель] [псевдоним] [аргументы = 'арг1, арг2, . . .]
```

Запускает программу, указанную в параметре PROGRAM файла *LISTENER.ORA*. *Псевдоним* – это имя, указанное в параметре PROGRAM.

START

```
START [прослушиватель]
```

Запускает указанный прослушивающий процесс.

STATUS

```
STATUS [прослушиватель]
```

Выводит краткую информацию о прослушивающем процессе, включая параметры конфигурации, адреса протоколов и сводную информацию о зарегистрированных службах.

STOP

```
STOP [прослушиватель]
```

Останавливает прослушиватель.

TRACE

```
TRACE {OFF | USER | ADMIN | SUPPORT} [прослушиватель]
```

Управляет уровнем детализации трассировки для прослушивающего процесса. Действует аналогично команде SET TRC_LEVEL.

VERSION

```
VERSION
```

Выводит версию прослушивающего процесса.

TNSPing

```
tnsping имя_сетевой_службы [счетчик]
```

TNSPing – это утилита командной строки, применяемая для проверки соединения между клиентом Oracle Net Services и сервером. Утилита действует аналогично команде *ping* для протокола TCP.

Утилита запускается из командной строки, имя программы – *tnsping*.

Имя_сетевой_службы – это имя сетевой службы, определенное в файле *TNSNAMES.ORA*, или служба имен. *Счетчик* определяет, сколько раз программа TNSPing будет пытаться подключиться к серверу.

TNSPing выводит сообщение, содержащее информацию о соединении для доступа к сетевой службе, сообщение «OK» об успешном выполнении (в случае успеха) для каждой попытки соединения и время, потраченное на установку соединения.

Oracle Net Manager

Oracle Net Manager – это инструмент управления с графическим интерфейсом пользователя, позволяющим создавать, исследовать и тестировать конфигурации для прослушивающего процесса, соединения и Oracle Names. Oracle Net Manager может применяться автономно или в составе Oracle Enterprise Manager. Oracle Net Manager включает в себя несколько мастеров, позволяющих решать различные задачи конфигурирования, а именно:

Мастер имен сетевых служб – Net Service Name wizard

Применяется для создания файла *TNSNAME.ORA*

Мастер переноса на сервер каталогов – Directory Server Migration wizard

Применяется для переноса информации из файла *TNSNAMES.ORA* на сервер каталогов.

Мастер имен – Names wizard

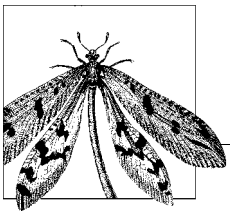
Предназначен для создания и конфигурирования сервера Oracle Names

В версии Oracle8*i* задачи Oracle Net Manager решаются с помощью Net8 Configuration Assistant. В Oracle8 они решались с помощью Oracle Net8 Assistant и Net8 Easy Configuration Tool.

Oracle Net Configuration Assistant

Инструмент управления Configuration Assistant, который назывался Net8 Configuration Assistant в Oracle8*i* и Net8 Easy Configuration Tool в Oracle8, применяется в процессе установки для конфигурирования имен прослушивающих процессов, адресов протоколов, методов именования, имен сетевых служб и использования каталогов.

Кроме того, в Oracle8*i* Database Configuration Assistant добавляет некоторую информацию о базе данных в файл *LISTENER.ORA*.



6

Словарь данных

Словарь данных (*data dictionary*) Oracle – это набор таблиц и связанных с ними представлений, который предоставляет пользователю возможность увидеть внутреннюю деятельность и структуру базы данных Oracle. Обращаясь к этим представлениям, вы можете получить информацию о любом объекте и пользователе базы данных. Все средства мониторинга Oracle обращаются за информацией к словарю данных и представляют ее в удобном формате.

Обычно словарь данных состоит из ряда представлений, принадлежащих пользователю SYS. Такие представления, называемые статическими представлениями словаря данных (*static data dictionary views*), отображают информацию, содержащуюся в таблицах, обновляемых в процессе выполнения сервером Oracle команд DDL. Таблицы и представления схемы SYS, а также набор открытых синонимов для представлений создаются сценарием *catalog.sql*. Кроме того, к созданию таблиц и представлений в схеме SYSTEM приводит установка некоторых дополнительных компонентов Oracle. Вообще говоря, таблицы и представления, принадлежащие пользователю SYSTEM существуют больше для поддержки функциональности хранимых процедур PL/SQL (о которых мы поговорим в главе 9), чем для поддержки базовой функциональности Oracle.

Второе множество представлений – динамические представления производительности (*dynamic performance data dictionary views*), которые часто называют V$-представлениями. В основе этих представлений лежат внутренние структуры памяти, поддерживаемые Oracle, например, виртуальные таблицы (их имена всегда начинаются с префикса «X$»). Статические представления словаря данных предоставляют информацию о базе данных, а V$-представления и лежащие в их основе X$-таблицы – об активном экземпляре.

Статические представления словаря данных

Статические представления словаря данных в их нынешнем виде появились в версии Oracle 6. Это были представления, принадлежащие схеме SYS, которые строились на основе принадлежащих ей таблиц и обеспечивали пользователей информацией об объектах базы данных.

Семейства представлений

Построение словаря данных во многом напоминает матрицу. Первым признаком, по которому можно классифицировать представления словаря данных, является широта охватываемой ими информации. Представления можно разделить на четыре группы:

*Представления USER_*

Представления, которые позволяют видеть принадлежащие пользователю объекты. Имена большей части таких представлений начинаются с USER_.

*Представления ALL_*

Представления, которые позволяют видеть объекты, принадлежащие пользователю или на которые ему были выданы привилегии. Имена большей части таких представлений начинаются с ALL_.

*Представления DBA_*

Представления, которые позволяют видеть все объекты базы данных. Они, главным образом, предназначены для администратора БД. Имена большей части таких представлений начинаются с DBA_.

Другие представления

Прочие представления, предлагающие общую информацию о базе данных.

В основе второго способа классификации представлений словаря данных лежит их содержимое. Многие представления USER_, ALL_ и DBA_ сгруппированы в соответствии с окончаниями своих имен (например, TABLES, COLUMNS и т.д.). Группы представлений предоставляют информацию о различных сущностях, таких как:

- Таблицы
- Распределение пространства
- Столбцы
- Представления
- Объекты
- Сетевые объекты

Представления ALL_ имеют такую же структуру, что и представления DBA_. Структура представлений USER_ также напоминает структуру DBA_, только в них отсутствует столбец OWNER. В этой главе мы поговорим о семействах представлений и о других важных представлениях. Для представлений, которые существуют в нескольких формах (т.е., ALL_, DBA_, USER_), используется обозначение *_имя_представления*. Например, если написано *_INDEXES, то подразумеваются три представления:

- ALL_INDEXES
- DBA_INDEXES
- USER_INDEXES

Если поддерживаются не все три разновидности представлений, то об этом будет специально упомянуто. Существуют представления, имеющие только одну форму или не соответствующие приведенному образцу наименования, – такие представления будут приведены без символа *.

Наиболее распространенные статические представления словаря данных

В разделе собраны наиболее часто используемые статические представления словаря данных Oracle. Представления разбиты на категории в зависимости от их функциональности и упорядочены по алфавиту внутри категорий.

Advanced Queuing

Следующие представления предлагают информацию об очередях сообщений:

**_ATTRIBUTE_TRANSFORMATIONS*

Перечисляет все функции для преобразования сообщений. Представления ALL_ATTRIBUTE_TRANSFORMATIONS не существует. Появилось в Oracle9*i*.

DBA_TRANSFORMATIONS

Предоставляет данные о преобразованиях сообщений Advanced Queuing. Появилось в Oracle9*i*.

**_QUEUE_SCHEDULES*

Показывает, когда будут доставлены определенные сообщения, стоящие в очереди.

**_QUEUE_TABLES*

Перечисляет таблицы, предназначенные для хранения очередей, определенных как часть компонента Advanced Queuing.

**_QUEUES*

Перечисляет очереди, определенные как часть компонента Advanced Queuing.

Журнал аудита

Представления в данном разделе предлагают информацию о статусе аудита и о журналах аудита:

ALL_DEF_AUDIT_OPTS

Содержит все принимаемые по умолчанию для новых объектов параметры аудита.

AUDIT_ACTIONS

Перечисляет коды аудита и соответствующие описания.

**_AUDIT_EXISTS*

Содержит записи журнала аудита, создаваемые командой AUDIT NOT EXISTS.

**_AUDIT_OBJECT*

Содержит информацию журнала аудита для объектов.

**_AUDIT_SESSION*

Содержит информацию журнала аудита для всех подключений и отключений.

**_AUDIT_STATEMENT*

Содержит информацию журнала аудита для всех отслеживаемых команд.

**_AUDIT_TRAIL*

Содержит всю информацию из журнала аудита. Все остальные представления *_AUDIT_* являются подмножествами данного.

**_OBJ_AUDIT_OPTS*

Содержит все действующие параметры аудита объектов.

**_PRIV_AUDIT_OPTS*

Перечисляет все действующие параметры аудита системных привилегий.

STMT_AUDIT_OPTION_MAP

Перечисляет команды SQL, для которых может выполняться аудит.

**_STMT_AUDIT_OPTS*

Перечисляет все действующие параметры аудита команд.

SYSTEM_PRIVILEGE_MAP

Перечисляет системные привилегии, для которых может проводиться аудит.

TABLE_PRIVILEGE_MAP

Перечисляет параметры аудита объектов, применяемые для аудита объектов схемы.

Технология захвата изменений данных

Oracle9*i* поддерживает возможность Change Data Capture (захват изменений данных), применяемую в основном в хранилищах данных и позволяющую пользователю создать множество *таблиц изменений* (*change tables*), которые затем будут использоваться для применения изменений к базовым таблицам.

**_SOURCE_TAB_COLUMNS*

Перечисляет столбцы исходных таблиц, которые содержатся в таблицах изменений.

**_SOURCE_TABLES*

Перечисляет связи между таблицами изменений и их исходными таблицами.

**_SUBSCRIBED_COLUMNS*

Перечисляет столбцы публикуемых таблиц, на которые производилась подписка.

**_SUBSCRIBED_TABLES*

Перечисляет все публикуемые таблицы, на которые проводилась подписка.

**_SUBSCRIPTIONS*

Перечисляет все подписки (subscriptions).

Ограничения целостности

Представления данного раздела предоставляют информацию об ограничениях целостности (constraints) и включенных в них столбцах:

**_CONS_COLUMNS*

Показывает, какие столбцы использованы в каждом из ограничений.

**_CONSTRAINTS*

Перечисляет все ограничения, заданные для базы данных.

Словарь данных

Представления этого раздела предлагают информацию об объектах словаря данных.

**_CATALOG*

Перечисляет все таблицы, представления, последовательности и синонимы базы данных.

**_DEPENDENCIES*

Перечисляет зависимости между объектами базы данных. Используется для того, чтобы определить, какие объекты становятся недействительными после изменения или удаления других объектов.

**_OBJECTS*

Перечисляет все объекты базы данных. Является предшественником компонента Oracle Objects Option и не ограничивается объектами, созданными при помощи Objects Option.

DICT_COLUMNS

Перечисляет все столбцы, определенные в представлениях словаря данных.

DICTIONARY

Перечисляет все представления словаря данных.

Внешние таблицы

В Oracle9*i* появилась возможность обращаться к данным, хранящимся в файлах вне базы данных Oracle. Такие файлы называют *внешними таблицами* (*external tables*). Следующие представления служат источником метаданных о внешних таблицах:

**_EXTERNAL_LOCATIONS*

Перечисляет источники внешних таблиц.

**_EXTERNAL_TABLES*

Описывает атрибуты внешних таблиц.

Индексы

Представления данного раздела предлагают информацию об индексах и индексированных столбцах.

**_IND_COLUMNS*

Перечисляет все проиндексированные столбцы.

**_INDEXES*

Перечисляет все индексы.

INDEX_HISTOGRAM

Содержит информацию о распределении ключей индекса внутри таблицы. Заполняется для каждого индекса при помощи команды ANALYZE INDEX ... VALIDATE STRUCTURE.

INDEX_STATS

Содержит информацию о структуре индекса. Заполняется для каждого индекса при помощи команды ANALYZE INDEX ... VALIDATE STRUCTURE.

Задания

Данные представления предлагают информацию об очередях заданий (*job queues*), которыми управляет встроенный пакет Oracle DBMS_JOBS. Эти очереди заданий используются при репликации и работе с Oracle Enterprise Manager, но с ними может работать любое приложение.

**_JOBS*

Перечисляет все задания, которые были определены.

**_JOBS_RUNNING*

Перечисляет все задания, выполняющиеся в текущий момент времени.

Большие объекты (LOB)

Информация о больших объектах (*Large Objects* – LOB) содержится в следующих представлениях:

**_DIRECTORIES*

Перечисляет все определенные внешние каталоги, в которых хранятся данные типа BFILE.

**_LOBS*

Перечиляет все большие объекты, которые были определены в базе данных.

Блокировки

Текущий статус блокировок в базе данных описывается следующими представлениями:

**_BLOCKERS*

Перечисляет все сеансы, удерживающие блокировки, освобождения которых ожидают другие сеансы.

DBMS_LOCK_ALLOCATED

Показывает, какие блокировки были выполнены текущим пользователем.

**_DDL_LOCKS*

Перечисляет все существующие DDL-блокировки.

**_DML_LOCKS*

Перечисляет все существующие DML-блокировки.

**_KGLLOCK*

Перечисляет все KGL-блокировки (Kernel Generic Library cache – кэш общей библиотеки ядра) для базы данных.

**_LOCK_INTERNAL*

Содержит внутреннюю информацию для каждой блокировки, определенной в *_LOCKS.

**_LOCKS*

Перечисляет все установленные и запрошенные блокировки базы данных.

**_WAITERS*

Перечисляет все сеансы, ожидающие освобождения блокировки, удерживаемой другим сеансом.

Журнальная группа

Журнальная группа предназначена для параллельной записи в несколько идентичных журнальных файлов, расположенных на разных носителях. Применение журнальных групп позволяют минимизировать влияние возможного сбоя дисковой системы на процесс восстановления:

**_LOG_COLUMNS*

Перечисляет столбцы, назначенные журнальным группам. Появилось в Oracle9*i*.

**_LOG_GROUPS*

Перечисляет таблицы, соответствующие журнальным группам, и указывает, всегда ли для таблицы используется журнальная группа или же только при изменении некоторых столбцов таблицы. Появилось в Oracle9*i*.

Материализованные представления

Материализованное представление – это представление, применяемое для увеличения скорости запросов к хранилищам данных за счет предварительного вычисления сводной информации. Сведения о материализованных представлениях в Oracle9*i* можно получить при помощи следующих представлений:

**_BASE_TABLE_MVIEWS*

Приводит информацию о существующих материализованных представлениях.

**_MVIEW_LOGS*

Приводит информацию о журналах материализованных представлений, которые отслеживают изменения в основных таблицах и могут использоваться для обновления материализованных представлений.

Работа в сети и распределенные транзакции

Представления, перечисленные в данном разделе, информируют о сетевой работе Oracle, удаленных базах данных и распределенных транзакциях в таких удаленных базах данных:

**_2PC_NEIGHBORS*

Содержит информацию о *точке фиксации* (commit *point*) распределенных транзакций, перечисленных в *_2PC_PENDING.

**_2PC_PENDING*

Приводит информацию о распределенных транзакциях, требующих восстановления.

**_DB_LINKS*

Перечисляет все связи базы данных.

GLOBAL_NAME

Указывает глобальное имя базы данных. Позволяет определить, к какой базе данных подключено приложение.

**_PENDING_TRANSACTIONS*

Содержит дополнительную информацию о незавершенных транзакциях (либо по причине сбоя, либо из-за того, что не было получено сообщение о фиксации или откате).

TRUSTED_SERVERS

Указывает, какие серверы были идентифицированы как доверенные.

Objects Option

Следующие представления предлагают информацию об объектах, созданных при помощи Oracle Objects Option:

**_COLL_TYPES*

Перечисляет созданные типы коллекций.

**_METHOD_PARAMS*

Перечисляет все параметры для методов, определенных в *_TYPE_METHODS.

**_METHOD_RESULTS*

Перечисляет все результаты методов, определенных в *_TYPE_METHODS.

_NESTED_TABLES

Перечисляет все вложенные таблицы, созданные средствами Objects Option.

_OBJECT_TABLES

Перечисляет все таблицы, созданные средствами Objects Option.

_REFS

Перечисляет столбцы REF и атрибуты объектов.

_TYPE_ATTRS

Перечисляет атрибуты всех типов.

_TYPE_METHODS

Перечисляет все методы, созданные для поддержки каждого типа, определенного в *_TYPES.

_TYPES

Перечисляет все созданные типы.

Секционирование

Данные о секционированных таблицах и индексах содержатся в следующих представлениях:

_IND_PARTITIONS

Перечисляет все секции (partitions) индекса. Каждой секции индекса соответствует одна строка.

_PART_COL_STATISTICS

Содержит статистическую информацию о проанализированных секционированных столбцах.

_PART_HISTOGRAMS

Содержит информацию о гистограммах, созданных для отдельных секций.

_PART_INDEXES

Перечисляет все секционированные индексы. Каждому секционированному индексу соответствует одна строка.

_PART_KEY_COLUMNS

Приводит столбцы ключей секционирования для всех секций.

_PART_TABLES

Перечисляет все секционированные таблицы. Каждой секционированной таблице соответствует одна строка.

_TAB_PARTITIONS

Перечисляет все секции таблицы. Каждой секции таблицы соответствует одна строка.

PL/SQL

Представления данного раздела предлагают информацию о хранимых программах PL/SQL, а именно о функциях, процедурах, пакетах и триггерах:

ALL_ARGUMENTS

Перечисляет все разрешенные аргументы хранимых процедур и функций.

**_ERRORS*

Перечисляет все текущие ошибки хранимых объектов.

**_LIBRARIES*

Перечисляет внешние библиотеки, которые могут вызываться из пакетов, процедур и функций PL/SQL.

**_OBJECT_SIZE*

Показывает размер скомпилированного кода для каждого пакета, процедуры, функции и триггера PL/SQL.

PUBLIC_DEPENDENCY

Перечисляет зависимости, используя только номера объектов.

**_SOURCE*

Приводит исходный текст PL/SQL для пакетов, процедур и функций.

**_TRIGGER_COLS*

Перечисляет столбцы, на которые имеются ссылки в триггерах.

**_TRIGGERS*

Приводит PL/SQL-код триггеров базы данных.

Репликация

Информация, используемая средствами репликации Oracle, предоставляется следующими представлениями (в настоящее время для получения этой информации корпорация Oracle рекомендует применять Replication Manager):

**_ANALYZE_OBJECTS*

Перечисляет объекты, которые были проанализированы. Не поддерживается в Oracle9*i*.

**_REGISTERED_SNAPSHOT_GROUPS, _REGISTERED_MVIEW_GROUPS*

Перечисляет зарегистрированные группы моментальных копий.

**_REPAUDIT_ATTRIBUTE*

Перечисляет атрибуты аудита репликации.

**_REPAUDIT_COLUMN*

Перечисляет столбцы аудита репликации.

**_REPCATLOG*

Приводит промежуточные статусы всех асинхронных административных запросов и всех генерируемых сообщений об ошибках.

**_REPCOLUMN*

Перечисляет реплицируемые столбцы для группы.

**_REPCOLUMN_GROUP*

Перечисляет группы столбцов, определенные для таблицы. Представления USER_REPCOLUMN_GROUP не существует.

**_REPCONFLICT*

Перечисляет таблицы с методами разрешения конфликтов репликации и методы для таблиц. Представления USER_ALL_REPCONFLICT не существует.

*_REPDDL

Приводит команды DDL для объектов репликации.

*_REPGENOBJECTS

Перечисляет объекты, сформированные для поддержки репликации.

*_REPGROUP

Описывает группы репликации, доступные пользователю.

*_REPGROUPED_COLUMN

Перечисляет столбцы в группах столбцов для каждой таблицы.

*_REPKEY_COLUMNS

Приводит информацию о столбцах первичных ключей для реплицируемых таблиц.

*_REPOBJECT

Перечисляет объекты в каждой реплицируемой группе объектов.

*_REPPARAMETER_COLUMN

Приводит информацию о столбцах, используемых для разрешения конфликтов.

*_REPPRIORITY

Перечисляет значение и уровень приоритета каждой группы приоритетов.

*_REPPRIORITY_GROUP

Перечисляет группы приоритетов и группы приоритетов узлов для группы реплицируемых объектов.

*_REPPROP

Перечисляет способы, применяемые для *распространения* (*propagation*) реплицируемого объекта.

*_REPRESOL_STATS_CONTROL

Описывает сбор статистики для разрешения конфликтов.

*_REPRESOLUTION

Перечисляет методы, используемые для разрешения конфликтов для указанной схемы.

*_REPRESOLUTION_METHOD

Перечисляет команды разрешения конфликтов.

*_REPRESOLUTION_STATISTICS

Приводит информацию о разрешенных конфликтах репликации.

*_REPSITES

Перечисляет элементы реплицируемой группы объектов.

DBA_REPEXTENSIONS

Перечисляет текущие операции, которые добавляют новые главные узлы в главную группу без ее *замораживания* (*quiescing*). Появилось в Oracle9*i*.

DBA_REPSITES_NEW

Перечисляет новые узлы репликации, которые вы планируете добавить в среду репликации. Появилось в Oracle9*i*.

DEFCALLDEST

Перечисляет назначение каждого отложенного вызова удаленной процедуры. Появилось в Oracle9*i*.

Безопасность

Представления раздела служат источником информации о пользователях, привилегиях и политике безопасности, реализующих детальный контроль доступа (Fine-Grained Access Control – FGAC):

**_APPLICATION_ROLES*

Описывает все роли приложения, для которых определена политика аутентификации. Представления ALL_ не существует. Появилось в Oracle9*i*.

**_COL_PRIVS*

Перечисляет все выданные в БД привилегии на столбцы.

**_GLOBAL_CONTEXT*

Перечисляет все глобальные контексты; это наборы определяемых приложением атрибутов, которые могут применяться для определения прав доступа, выдаваемых экземпляру. Представления USER_GLOBAL_CONTEXT не существует. Появилось в Oracle9*i*.

**_POLICY_CONTEXTS*

Перечисляет политики безопасности и соответствующие контексты. Появилось в Oracle9*i*.

**_POLICY_GROUPS*

Перечисляет различные группы для различных политик безопасности. Это представление появилось в Oracle9*i*.

**_PROFILES*

Перечисляет все определенные профили.

RESOURCE_COST

Выводит стоимость, присвоенную каждому ресурсу, для совокупных ограничений ресурсов.

RESOURCE_MAP

Сопоставляет номера профилей ресурсов их именам.

**_ROLES*

Перечисляет все роли.

**_ROLE_PRIVS*

Перечисляет все роли, выданные пользователям и другим ролям.

ROLE_ROLE_PRIVS

Перечисляет роли, выданные другим ролям. Подмножество *_ROLE_PRIVS.

ROLE_SYS_PRIVS

Перечисляет системные привилегии, выданные ролям. Подмножество *_SYS_PRIVS.

ROLE_TAB_PRIVS

Перечисляет привилегии на таблицы, выданные ролям. Подмножество *_TAB_PRIVS.

SESSION_PRIVS

Показывает, какие системные привилегии активны для текущего сеанса.

SESSION_ROLES

Показывает, какие роли активны для текущего сеанса.

**_SYS_PRIVS*

Показывает, каким пользователям какие системные привилегии были назначены.

**_TAB_PRIVS*

Выводит все привилегии на доступ к объектам. Включает привилегии не только для таблиц, но и для представлений, последовательностей, пакетов, процедур и функций.

USER_PASSWORD_LIMITS

Выводит действующие в текущем сеансе ограничения для паролей.

**_USERS*

Перечисляет всех пользователей.

Последовательности

Сведения о последовательностях предлагает такое представление:

**_SEQUENCES*

Перечисляет все последовательности базы данных.

Управление сервером

Представления этого раздела предлагают информацию о текущем статусе базы данных:

NLS_DATABASE_PARAMETERS

Выводит действующие параметры поддержки национальных языков (NLS) на уровне базы данных.

NLS_INSTANCE_PARAMETERS

Выводит параметры NLS, действующие на уровне экземпляра.

NLS_SESSION_PARAMETERS

Выводит параметры NLS, действующие на уровне сеанса.

PRODUCT_COMPONENT_VERSION

Выводит версии всех установленных компонентов Oracle.

SM$VERSION

Выводит версию Oracle для использования в Server Manager.

SQLJ

SQLJ – это способ встраивания статических команд SQL в программы Java (подробно мы поговорим об этом в главе 11). Команды SQLJ обращаются к объектам SQLJ, которые являются составными структурами данных, связанными с командами SQLJ. Объекты SQLJ хранятся в базе данных Oracle и описываются следующими представлениями:

**_SQLJ_TYPES*

Описывает типы объектов SQLJ. Появилось в Oracle9*i*.

**SQLJ_TYPE_ATTRS*

Описывает атрибуты, сопоставленные каждому типу объектов SQLJ. Появилось в Oracle9*i*.

**_SQLJ_TYPE_METHODS*

Описывает методы, сопоставленные каждому типу объектов SQLJ. Появилось в Oracle9*i*.

Распределение пространства

Представления данного раздела описывают внутренние хранилища базы данных, такие как файлы данных, табличные пространства, свободные экстенты, используемые экстенты и сегменты.

**_DATA_FILES*

Перечисляет все файлы данных, используемые базой данных.

**_EXTENTS*

Перечисляет все занятые экстенты для всех сегментов.

**_FREE_SPACE*

Перечисляет все свободные экстенты. Данное представление в сочетании с представлением *_EXTENTS показывает полное пространство, имеющееся в *_DATA_FILES.

**_FREE_SPACE_COALESCED*

Перечисляет все экстенты, находящиеся в начале блоков свободных экстентов.

**_ROLLBACK_SEGS*

Перечисляет все сегменты отката.

**_SEGMENTS*

Перечисляет все сегменты.

**_TABLESPACES*

Перечисляет все табличные пространства.

**_TS_QUOTAS*

Показывает выделенную пользователю квоту и использованную область в табличных пространствах.

Синонимы

Информацию о синонимах предлагает следующее представление:

**_SYNONYMS*

Перечисляет все синонимы в базе данных.

Таблицы, кластеры и представления

Представления данного раздела информируют о таблицах, кластерах и представлениях:

**_ALL_TABLES*

Перечисляет все объекты и реляционные таблицы.

**_CLU_COLUMNS*

Перечисляет все ключи кластеров.

**_CLUSTER_HASH_EXPRESSIONS*

Перечисляет значения хеша, используемые для необязательных кластерных хеш-индексов.

**_CLUSTERS*

Перечисляет все кластеры базы данных.

**_COL_COMMENTS*

Выводит комментарии для всех столбцов таблиц и представлений.

**_TAB_COL_STATISTICS*

Выводит информацию об анализируемых столбцах.

**_TAB_COLUMNS*

Выводит все столбцы таблиц и представлений.

**_TAB_COMMENTS*

Выводит комментарии для всех таблиц и представлений.

**_TAB_HISTOGRAMS*

Выводит гистограммы всех таблиц.

**_TABLES*

Выводит все реляционные таблицы.

**_UPDATABLE_COLUMNS*

Выводит столбцы в представлениях с соединениями, которые могут быть обновлены.

**_VIEWS*

Выводит все представления.

Прочие представления

С перечисленными ниже представлениями и таблицами работают отдельные пользователи:

DBA_UNDO_EXTENTS

Выводит время фиксации каждого экстента в табличном пространстве UNDO. Появилось в Oracle9*i*.

CHAINED_ROWS

Заполняется командой ANALYZE TABLE и описывает все *расщепленные строки* (*chained rows*) таблицы. Создается при помощи сценария *utlchain.sql*.

EXCEPTIONS

Содержит список всех строк с *нарушением целостности* (*constraint violation*). Заполняется при попытке создания или использования ограничения. Создается при помощи сценария *utlexcpt.sql*.

**_JOIN_IND_COLUMNS*

Описывает столбцы, используемые в соединении, которому сопоставлен *битовый индекс* (*bitmapped index*). Появилось в Oracle9*i*.

PLAN_TABLE

Используется процессом EXPLAIN_PLAN для вывода плана исполнения команды SQL. Создается при помощи сценария *utlxplan.sql*.

*_PROCEDURES

Выводит информацию о процедурах, существующих в базе данных. Например, о том, являются ли они *агрегатными функциями* (*aggregate functions*), *конвейерными табличными функциями* (*pipelined table functions*) или же *параллельно разрешенными функциями* (*parallel-enabled functions*). Появилось в Oracle9*i*.

*_PROXIES

Приводит информацию обо всех прокси-соединениях в системе. Представления ALL_PROXIES не существует. Появилось в Oracle9*i*.

*_RESUMABLE

Перечисляет все возобновляемые (RESUMABLE) команды. Когда базе данных не хватает пространства, такие команды позволяют DBA приостановить выполнение, увеличить доступное пространство и возобновить работу. Представления ALL_RESUMABLE не существует. Появилось в Oracle9*i*.

*_STORED_SETTINGS

Предоставляет информацию о значении постоянных параметров для хранимых PL/SQL-программ. Появилось в Oracle9*i*.

Другие статические представления словаря данных

Приведенные далее представления выводят важную информацию о структуре базы данных, но обычно они не применяются напрямую. Перечислим их для полноты картины.

Экспорт

Следующие представления обеспечивают информацию для утилит Export и Import:

*_EXP_FILES

*_EXP_OBJECTS

*_EXP_VERSION

Шлюзы данных

Информацию, необходимую для поддержки *внешних источников данных* (Foreign Data Sources – FDS) или *шлюзов данных* (*data gateways*) предлагают следующие представления:

HS_ALL_CAPS

HS_ALL_DD

HS_ALL_INITS

HS_BASE_CAPS

HS_BASE_DD

HS_CLASS_CAPS

HS_CLASS_DD

HS_CLASS_INIT

HS_EXTERNAL_OBJECT_PRIVILEGES

HS_EXTERNAL_OBJECTS

HS_EXTERNAL_USER_PRIVILEGES

HS_FDS_CLASS

HS_FDS_INST
HS_INST_CAPS
HS_INST_DD
HS_INST_INIT

Oracle Parallel Server/Real Application Clusters

Сведения о состоянии среды Oracle Parallel Server/Real Application Clusters можно получить из приведенных ниже представлений. Однако имейте в виду, что при стандартной работе с Real Application Clusters эти данные не требуются:

FILE_LOCK
FILE_PING

Вызовы удаленных процедур

Следующие представления предлагают информацию о статусе вызовов удаленных процедур (Remote Procedure Calls – RPC):

DEFCALL
DEFDEFAULTDEST
DEFERRCOUNT
DEFERROR
DEFLOB
DEFPROPAGATOR
DEFSCHEDULE
DEFTRAN
DEFTRANDEST
ORA_KGLR7_DB_LINKS
ORA_KGLR7_DEPENDENCIES
ORA_KGLR7_IDL_CHAR
ORA_KGLR7_IDL_SB4
ORA_KGLR7_IDL_UB1
ORA_KGLR7_IDL_UB2

Моментальные копии

Сведения о моментальных копиях или о пришедших им на смену материализованных представлениях (MVIEWS) можно получить из следующих представлений:

DBA_REGISTERED_SNAPSHOT_GROUPS
DBA_SNAPSHOT_LOG_FILTER_COLS
*_RCHILD
*_REFRESH
*_REFRESH_CHILDREN
*_REGISTERED_SNAPSHOTS/_REGISTERED_MVIEWS
*_RGROUP
*_SNAPSHOT_LOGS/MVIEW_LOGS
*_SNAPSHOT_REFRESH_TIMES/MVIEW_REFRESH_TIMES
*_SNAPSHOTS/MVIEWS

SQL*Loader

Информация, используемая компонентом прямого маршрута SQL*Loader, обеспечивается такими представлениями:

LOADER_CONSTRAINT_INFO

LOADER_FILE_TS

LOADER_PARAM_INFO

LOADER_PART_INFO

LOADER_TAB_INFO

LOADER_TRIGGER_INFO

Восстановление табличного пространства на заданный момент времени

Следующие представления предлагают информацию, необходимую для *восстановления табличного пространства на указанный момент времени* (Tablespace Point-In-Time Recovery – PITR):

STRADDLING_RS_OBJECTS

TS_PITR_CHECK

TS_PITR_OBJECTS_TO_BE_DROPPED

Динамические представления производительности

Динамические представления производительности словаря данных (представления V$) содержат информацию в основном об экземпляре Oracle, а также информацию о базе данных, которую хранит экземпляр хранит. Такие представления считаются динамическими, поскольку их содержимое зависит от того, как работает экземпляр. Содержимое относится ко всему объему работ, а не к какой-то отдельной команде SQL.

Доступность динамических представлений производительности

Доступность конкретного динамического представления производительности словаря данных зависит от статуса экземпляра:

- Динамические представления производительности словаря данных, содержащие информацию о самом экземпляре (например, V$PARAMETER). Они доступны сразу же после запуска экземпляра.
- Динамические представления производительности словаря данных, содержащие информацию, хранящуюся в управляющих файлах. Они становятся доступными после того, как база данных смонтирована (что отмечено в описаниях представлений).
- Динамические представления производительности словаря данных, содержащие информацию о том, как ядро обрабатывает команды SQL. Такие представления доступны после того, как база данных открыта (отмечено в описании представлений).

Как строятся динамические представления производительности

В отличие от статических представлений словаря данных, которые являются представлениями существующих таблиц, динамические представления производительности – это представления для набора таблиц, которые физически не существуют в базе данных; они являются представлениями для X$ таблиц, которые в свою очередь представляют внутренние структуры памяти экземпляра Oracle, например:

- V$DATABASE – это открытый синоним для представления SYS.V_$DATABASE.
- SYS.V_$DATABASE – это представление для SYS.V$DATABASE.
- V$DATABASE – это представление структуры памяти X$KCCDI.

Точное описание устройства представлений V$ встроено в ядро Oracle. Представление V$FIXED_VIEW_DEFINITION определяет все представления V$ как построенные на основе таблиц X$.

Знания о структуре таких представлений помогают понять, как они работают. Будучи изначально определенными внутри ядра Oracle, эти V$-таблицы фиксированной структуры доступны начиная с момента запуска экземпляра или монтирования базы данных. При открытии базы данных вступает в действие обычный механизм обработки команд SQL и используются открытые синонимы этих представлений. В случае применения открытых синонимов одно и то же имя доступно, если вы подключаетесь как INTERNAL до открытия базы данных и если вы подключаетесь как пользователь с привилегиями DBA после открытия базы данных.

Некоторые представления V$, доступные только при открытии базы данных, являются настоящими представлениями, основанными на таблицах X$ или других таблицах V$.

Глобальные динамические представления производительности (GV$-представления)

Начиная с версии Oracle8 в дополнение к динамическим представлениям производительности (V$-представления) появилось добавочное множество глобальных представлений производительности словаря данных (GV$-представления). Представления V$ предлагают информацию об экземпляре, к которому вы подключены, и о его управлении базой данных. Представления GV$ предоставляют ту же самую информацию для всех остальных экземпляров, в которых смонтирована та же база данных. Особый интерес такие представления имеют при работе в среде Real Applications Cluster и Oracle Parallel.

В глобальные динамические представления производительности словаря данных добавлен столбец INST_ID с именами, позволяющий определить, к какому именно экземпляру относится информация.

Динамические представления

В разделе собраны динамические представления производительности словаря данных. Для удобства они разделены на семейства по функциональности, а внутри семейств упорядочены по алфавиту. Для большей части представлений, описанных в последующих разделах, применяется обозначение *$имя_представления*. Символ (*) означает, что на представление можно ссылаться и как на V*$имя_представления*, и как на GV*$имя_представления*.

Advanced Queuing

Представление предлагает статистику производительности для Advanced Queuing:

**$AQ*

Предоставляют статистику сообщений для каждой очереди сообщений. Доступно после открытия базы данных.

Конфигурация

Информацию о текущей конфигурации среды Oracle обеспечивают следующие представления:

**$COMPATIBILITY*

Перечисляет задействуемые текущим экземпляром возможности, которые препятствуют переходу на более раннюю версию программного обеспечения Oracle. Так как эти возможности используются экземпляром, некоторые из них могут исчезнуть при обычном выключении базы данных.

**$COMPATSEG*

Перечисляет постоянные свойства базы данных, которые препятствуют переходу на более раннюю версию программного обеспечения Oracle.

**$EVENT_NAME*

Содержит описательную информацию обо всех возможных событиях ожидания.

**$LICENSE*

Содержит одну строку, указывающую максимально возможное количество одновременных и именованных пользователей, а также максимальное количество одновременных пользователей с момента запуска экземпляра.

**$MLS_PARAMETERS*

Приводит параметры инициализации и их текущие значения для Trusted Oracle. Формат совпадает с форматом *$PARAMETER. Доступно после открытия базы данных.

**$NLS_PARAMETERS*

Содержит текущие значения каждого из параметров поддержки национальных языков (NLS).

**$NLS_VALID_VALUES*

Перечисляет разрешенные значения для каждого из параметров NLS.

**$OPTION*

Перечисляет компоненты, установленные вместе с сервером Oracle.

**$PARAMETER*

Перечисляет все параметры инициализации и их текущие значения. Кроме того, указывает, задано ли текущее значение в файле инициализации или же речь идет о значении по умолчанию, а также можно ли изменять параметр при помощи команды ALTER SYSTEM или ALTER SESSION.

**$RMAN_CONFIGURATION*

Перечисляет все постоянные параметры конфигурации RMAN, стандартной утилиты резервного копирования и восстановления Oracle (она будет рассмотрена в главе 15). Появилось в Oracle9*i*.

$SPPARAMETER

Выводит содержимое *SPFILE*. Появилось в Oracle9*i*.

$STATNAME

Перечисляет имена всех статистик, хранящихся в *$SYSSTAT и *$SESSTAT.

$SYSTEM_PARAMETER

Перечисляет все параметры инициализации и их текущие значения. Кроме того, указывает, задано ли текущее значение в файле инициализации или же речь идет о значении по умолчанию, а также можно ли изменять параметр при помощи команды ALTER SYSTEM или ALTER SESSION.

$TIMEZONE_NAMES

Приводит имена и разрешенные аббревиатуры, используемые в базе данных Oracle для часовых поясов. Появилось в Oracle9*i*.

$VERSION

Перечисляет текущие номера версий для библиотечных компонентов ядра Oracle.

Кэш словаря данных

Представления этого раздела содержат информацию о том, как ядро Oracle управляет кэшем словаря данных и библиотечным кэшем.

$DB_OBJECT_CACHE

Перечисляет таблицы, индексы, кластеры, синонимы, процедуры, пакеты и триггеры PL/SQL, которые содержатся в библиотечном кэше.

$LIBRARYCACHE

Содержит статистическую информацию о производительности библиотечного кэша.

$ROWCACHE

Содержит статистическую информацию о производительности кэша словаря данных.

$SUBCACHE

Перечисляет все подчиненные кэши библиотечного кэша.

База данных

Сведения о физической базе данных содержатся в следующих представлениях:

$CONTROLFILE

Приводит имена всех управляющих файлов.

$CONTROLFILE_RECORD_SECTION

Предоставляет информацию об объеме данных, хранящихся в каждой секции управляющего файла. Доступно после монтирования базы данных.

$DATABASE

Предоставляет сведения о базе данных, которые хранятся в управляющем файле. Доступно после монтирования базы данных.

$DATAFILE

Содержит информацию о каждом файле данных, полученную из управляющего файла. Доступно после монтирования базы данных.

**$DATAFILE_HEADER*

Содержит информацию о каждом файле данных, полученную из заголовка файла данных. Доступно после монтирования базы данных.

**$DBFILE*

Содержит имена всех файлов данных. Поддерживается для обеспечения совместимости версий. Oracle рекомендует использовать вместо него *$DATAFILE. Доступно после монтирования базы данных.

**$OFFLINE_RANGE*

Предоставляет информацию об автономном состоянии файлов данных. Информация берется из управляющего файла. Доступно после монтирования базы данных.

**$TABLESPACE*

Предоставляет информацию о табличных пространствах на основе информации из управляющего файла. Доступно после монтирования базы данных.

**$TYPE_SIZE*

Указывает размер в байтах для различных компонентов блока данных или индексов Oracle.

**$UNDOSTAT*

Приводит разнообразную информацию об использовании базой данных пространства отката. Такие сведения могут применяться для оценки объема пространства отката, необходимого базе данных, и на их основе БД настраивает работу с пространством отката. Появилось в Oracle9*i*.

V$FILESTAT

Предоставляет информацию, связанную с активностью ввода/вывода, для каждого файла, используемого базой данных. Появилось в Oracle9*i*. Доступно после монтирования базы данных.

Экземпляр

Следующие представления содержат информацию о состоянии экземпляра:

**$BGPROCESS*

Предоставляет информацию о каждом фоновом процессе.

**$INSTANCE*

Предоставляет сведения о статусе текущего экземпляра.

Блокировки и защелки

Сведения о статусах блокировок и замков для экземпляра содержатся в следующих представлениях:

**$ACCESS*

Перечисляет все заблокированные объекты БД и все сеансы, обращающиеся к таким объектам.

**$BUFFER_POOL*

Приводит данные о доступных буферных пулах. Количество буферных пулов регулируется параметром инициализации DB_BLOCK_LRU_LATCHES.

**$ENQUEUE_LOCK*

Перечисляет все блокировки, принадлежащие объектам постановки в очередь.

*$LATCH

Приводит статистическую информацию обо всех защелках. Если защелка является родительской, то приводится суммарная статистика на каждую из дочерних защелок. Доступно после открытия базы данных.

*$LATCH_CHILDREN

Приводит статистику для всех дочерних защелок.

*$LATCH_MISSES

Приводит статистику обо всех неудачных попытках получения защелки.

*$LATCH_PARENT

Приводит статистику для всех родительских защелок.

*$LATCHHOLDER

Предоставляет информацию о текущих держателях защелок.

*$LATCHNAME

Приводит декодированные имена всех защелок из *$LATCH. Доступно после монтирования базы данных.

*$LOCK

Перечисляет все удерживаемые защелки и все ожидающие выполнения запросы на блокировки и защелки.

*$LOCKED_OBJECT

Перечисляет все объекты системы, в настоящий момент времени заблокированные транзакциями.

*$RESOURCE

Содержит имена, типы и адреса всех ресурсов системы.

Многопоточный сервер/Разделяемый сервер

Следующие представления содержат информацию о конфигурации и работе систем, использующих многопоточный сервер (Multi-Threaded Server – MTS) или разделяемый сервер (Shared Server, для Oracle9*i*):

*$CIRCUIT

Содержит информацию о виртуальных каналах, применяемых для подключения пользователей к экземпляру.

*$DISPATCHER

Предлагает информацию о различных конфигурируемых диспетчерских процессах.

*$DISPATCHER_RATE

Приводит статистическую информацию о пропускной способности диспетчерских процессов.

*$MTS

Приводит данные об общей активности MTS/Shared Server.

*$QUEUE

Приводит статистическую информацию об очереди сообщений MTS.

*$REQDIST

Предоставляет 12-столбцовую гистограмму распределения времени обслуживания запросов. Гистограмма сбалансирована по ширине и показывает количество попаданий в 12 смежных значениях времени. Размеры столбца могут со временем меняться.

*$SHARED_SERVER

Выводит статус каждого из разделяемых серверов.

Oracle Parallel Server/Real Application Clusters

Представления данного раздела присутствуют только в среде Oracle Parallel Server/Real Application Clusters:

*$ACTIVE_INSTANCES

Перечисляет все текущие экземпляры, для которых смонтирована база данных.

*$BH

Выводит статус и результат опроса каждого буфера данных в System Global Area (SGA). Доступно после открытия базы данных.

*$CACHE

Предоставляет информацию о заголовке блока каждого объекта текущего экземпляра в SGA. Доступно после открытия базы данных.

*$CACHE_LOCK

Предоставляет статус блокирования каждого блока данных текущего экземпляра в SGA. Доступно после открытия базы данных.

*$CLASS_PING

Показывает статистику по количеству опросов на класс блоков данных.

*$FALSE_PING

Перечисляет буферы, для которых проводятся излишние опросы из-за того, что на них распространяется другая блокировка. Если такие буферы обнаружены, то положение можно изменить с помощью параметра инициализации GC_FILES_TO_LOCKS. Доступно после открытия базы данных.

*$ENQUEUE_STAT

Выводит статистику для каждого из различных типов запросов на постановку в очередь для некоторого экземпляра базы данных. Постановка в очередь применяется для того, чтобы предотвратить одновременную запись в один блок данных несколькими пользователями или процессами. Появилось в Oracle9*i*.

*$FILES_CACHE_TRANSFER

Выводит количество опросов блоков для файла данных. До выхода Oracle9*i* это представление называлось *$FILE_PING. Доступно после монтирования базы данных.

*$GCSHVMASTER

Отслеживает распределение ресурсов Global Cache Service, за исключением тех, которые находятся в файлах, сопоставленных конкретному экземпляру. Появилось в Oracle9*i*.

**$GCSPFMASTER*

Отслеживает распределение ресурсов Global Cache Service, которые находятся в файлах, сопоставленных конкретному экземпляру. Появилось в Oracle9*i*.

**$GES_CONVERT_LOCAL*

Выводит время, использованное для преобразования локальных блокировок DLM. До выхода Oracle9*i* это представление называлось *$DLM_CONVERT_LOCAL.

**$GES_CONVERT_REMOTE*

Выводит время, использованное для преобразования удаленных блокировок DLM. До выхода Oracle9*i* это представление называлось *$DLM_CONVERT_REMOTE.

**$GES_LATCH*

Выводит общее и текущее количество непосредственно полученных DLM-защелок по их типам. До выхода Oracle9*i* это представление называлось *$DLM_LATCH. Доступно после открытия базы данных.

**$GES_LOCKS*

Выводит все DLM-блокировки и запросы на блокировки, которые заблокированы или блокируют запросы на другие блокировки. До выхода Oracle9*i* это представление называлось *$DLM_LOCKS.

**$GES_MISC*

Приводит статистическую информацию для различных параметров DLM. До выхода Oracle9*i* это представление называлось *$DLM_MISC.

**$HVMASTER*

Отслеживает распределение ресурсов Global Enqueue Service. Появилось в Oracle9*i*.

**$LOCK_ACTIVITY*

Дает общее представление о DLM-блокировках в текущем экземпляре.

**$LOCK_ELEMENT*

Предоставляет информацию обо всех PCM-блокировках в буферах данных.

**$LOCKS_WITH_COLLISIONS*

Выводит блокировки с большим количеством ложных опросов.

**$PING*

Подмножество *$CACHE; выводит только те буферы, которые опрашивались хотя бы один раз. Доступно после открытия базы данных.

Parallel Query

Информацию о поддерживаемых в Parallel Query операциях обеспечивают следующие представления:

**$EXECUTION*

Предлагает информацию о каждом исполнении Parallel Query.

**$PQ_SESSTAT*

Предоставляет статистическую информацию об активности Parallel Query в текущем сеансе.

**$PQ_SLAVE*

Предоставляет статистическую информацию о каждом входящем в систему сервере Parallel Query.

**$PQ_SYSSTAT*

Предлагает итоговую статистическую информацию Parallel Query для всей системы.

**$PQ_TQSTAT*

Предоставляет статистическую информацию для всех активных сеансов Parallel Query.

Восстановление

Следующие представления предлагают информацию о текущем статусе оперативного и автономного журналов, а также о процессах резервного копирования, управляемых диспетчером восстановления (Recovery Manager – RMAN):

**$ARCHIVE*

Перечисляет журналы, подлежащие архивированию. Доступно после монтирования базы данных.

**$ARCHIVE_DEST*

Содержит статус всех местоположений архивирования журналов, указанных для экземпляра. См. описание параметров инициализации LOG_ARCHIVE_DEST, LOG_ARCHIVE_DUPLEX_DEST и LOG_ARCHIVE_MIN_SUCCEED_DEST в главе 2.

**$ARCHIVE_DEST_STATUS*

Содержит информацию времени выполнения о текущем состоянии местоположений архивирования журнальных файлов. Может применяться для отслеживания степени выполнения архивирования нескольких журнальных файлов по разным направлениям. Появилось в Oracle9*i*.

**$ARCHIVE_GAP*

Содержит информацию об имеющихся пропусках в архивных журналах, которые могут помешать выполнению операций восстановления базы данных. Появилось в Oracle9*i*.

**$ARCHIVED_LOG*

Содержит информацию из управляющего файла для всех архивных журналов, основанную на представлении журнальных файлов. Доступно после монтирования базы данных.

**$BACKUP*

Содержит статус резервного копирования для всех оперативных файлов данных, управляемых процессом RMAN. Доступно после монтирования базы данных.

**$BACKUP_CORRUPTION*

Описывает все искажения файлов данных, обнаруженные в ходе резервного копирования, выполняемого процессом RMAN. Доступно после монтирования базы данных.

**$BACKUP_DATAFILE*

Показывает местоположение файла резервного копирования, используемого процессом RMAN. Доступно после монтирования базы данных.

**$BACKUP_DEVICE*

Предлагает список доступных устройств резервного копирования, поддерживаемых процессом RMAN.

$BACKUP_PIECE

Содержит информацию о каждом элементе резервного копирования (подмножество набора резервного копирования RMAN). Доступно после монтирования базы данных.

$BACKUP_REDOLOG

Содержит информацию об архивных журналах, резервное копирование которых было выполнено под управлением процесса RMAN. Доступно после монтирования базы данных.

$BACKUP_SET

Содержит информацию обо всех наборах резервного копирования, обслуживаемых процессом RMAN. Доступно после монтирования базы данных.

$COPY_CORRUPTION

Описывает все искажения файлов данных, обнаруженные в ходе копирования файла данных, проводимого процессом RMAN. Доступно после монтирования базы данных.

$DATAFILE_COPY

Содержит информацию о копиях файлов данных, которая содержится в управляющем файле. Эта информация поддерживается процессом RMAN. Доступно после монтирования базы данных.

$DELETED_OBJECT

Содержит информацию об архивных журналах, элементах файлов данных и копиях файлов данных, которые были удалены из управляющего файла. Доступно после монтирования базы данных.

$LOG

Содержит информацию о журналах из управляющего файла. Доступно после монтирования базы данных.

$LOG_HISTORY

Содержит информацию об архивных журналах из управляющего файла. Включает SCN-номера архивных журнальных файлов. Доступно после монтирования базы данных.

$LOGFILE

Выводит текущий статус всех журналов. Доступно после монтирования базы данных.

$LOGHIST

Содержит историю журналов из управляющего файла. В настоящее время корпорация Oracle рекомендует применять представление *$LOG_HISTORY. Доступно после монтирования базы данных.

$MANAGED_STANDBY

Представления, содержащие информацию о различных процессах Data Guard. Также могут применяться для наблюдения за процессом восстановления при помощи Data Guard. Появилось в Oracle9*i*.

$RECOVER_FILE

Перечисляет файлы данных, используемые при восстановлении носителя. Доступно после монтирования базы данных.

$RECOVERY_FILE_STATUS

Содержит информацию, относящуюся к процессу восстановления текущего файла. Доступно только процессу Recovery Manager, но не пользователям. Доступно после монтирования базы данных.

$RECOVERY_LOG

Содержит производную информацию из *$LOG_HISTORY, необходимую процессу Recovery Manager. Будучи запрошено пользователями, не возвращает строк. Доступно после монтирования базы данных.

$RECOVERY_PROGRESS

Содержит подмножество *$SESSION_LONGOPS, которое выводит текущий статус операций восстановления.

$RECOVERY_STATUS

Хранит текущую статистическую информацию для процесса восстановления. Информация доступна только процессу Recovery Manager. Представление не возвращает строк в ответ на запрос пользователей.

$STANDBY_LOG

Приводит информацию о журналах для резервной базы данных. Появилось в Oracle9*i*.

$THREAD

Содержит информацию обо всех текущих потоках журнала из управляющего файла. Доступно после монтирования базы данных.

Репликация

Следующие представления позволяют следить за процессом репликации в базе данных:

$MVREFRESH

Содержит информацию обо всех материализованных представлениях, которые обновляются в настоящий момент. Появилось в Oracle9*i*.

$REPLPROP

Содержит информацию о работе распространения в ходе репликации. Появилось в Oracle9*i*.

$REPLQUEUE

Содержит информацию обо всех отложенных транзакциях репликации. Появилось в Oracle9*i*.

Выделение ресурсов

Диспетчер ресурсов базы данных (Database Resource Manager – DRM) может применяться для определения правил распределения ресурсов между различными группами пользователей. В Oracle9*i* два динамических представления позволяют увидеть доступные методы выделения ресурсов:

ACTIVE_SESSION_POOL_MTH

Перечисляет все доступные в настоящее время методы выделения ресурсов для пула активных сеансов. Появилось в Oracle9*i*.

**QUEUEING_MTH*

Перечисляет все доступные в настоящее время методы выделения ресурсов, поставленные в очередь. Появилось в Oracle9*i*.

Безопасность

Информацию о привилегиях предлагают следующие представления:

**$ENABLEDPRIVS*

Перечисляет все системные привилегии, разрешенные для текущего сеанса. Включает явно выданные привилегии и привилегии, доступные через роль.

**$PWFILE_USERS*

Перечисляет всех пользователей, которые определены в файле паролей как имеющие привилегии SYSDBA или SYSOPER.

Сеанс

Представления этого раздела предлагают информацию о текущем сеансе Oracle:

**$MYSTAT*

Содержит информацию о статистиках текущего сеанса. Каждой записи в *$STATNAME соответствует одна строка.

**$PROCESS*

Перечисляет все процессы экземпляра. Для получения более полной информации можно соединить это представление с *$SESSION.

**$SESS_IO*

Содержит последнюю информацию о вводе/выводе для каждого сеанса базы данных. Доступно после открытия базы данных.

**$SESSION*

Перечисляет все сеансы экземпляра.

**$SESSION_CONNECT_INFO*

Приводит информацию о сетевом соединении для текущего сеанса.

**$SESSION_CURSOR_CACHE*

Приводит информацию об использовании курсора текущего сеанса.

**$SESSION_EVENT*

Содержит информацию о том, сколько времени каждый сеанс провел в ожидании каждого события, указанного в *$EVENT_NAME.

**$SESSION_LONGOPS*

Приводит информацию о статусе долго выполняющихся операций для сеансов. Указывает количество уже выполненных единиц работы и предполагаемый объем работы, оставшейся до завершения операции.

**$SESSION_OBJECT_CACHE*

Приводит статистическую информацию для кэша объекта в рамках текущего сеанса текущего экземпляра.

**$SESSION_WAIT*

Указывает, какие ресурсы ожидает каждый из активных сеансов и как долго каждый сеанс ожидает каждый ресурс.

**$SESSTAT*

Приводит статистическую информацию для всех сеансов. Каждой статистике, указанной в *$STATNAME, соответствует одна строка.

SGA

Представления этого раздела содержат сведения о SGA:

**$CURRENT_BUCKET*

Приводит возросшее количество непопаданий в буфер , если значение параметра инициализации DB_BLOCK_BUFFERS следует уменьшить. Устарело после Oracle 8.1. Доступно после открытия базы данных.

**$PGASTAT*

Приводит статистики расходования памяти, на основании которых Oracle Memory Manager выделяет максимальный объем памяти, доступной единовременно отдельной рабочей области. Появилось в Oracle9*i*.

**$RECENT_BUCKET*

Приводит возросшее количество попаданий в буфер, если значение параметра инициализации DB_BLOCK_BUFFERS должно быть увеличено (см. описание параметра инициализации DB_BLOCK_LRU_EXTENDED_STATISTICS в главе 2). Доступно после открытия базы данных.

**$SGA*

Содержит информацию о размере (в байтах) каждого из компонентов SGA.

**$SGASTAT*

Приводит более подробную информацию об использовании SGA, чем *$SGA. Показывает разбиение на области SHARED_POOL и LARGE_POOL.

**$SHARED_POOL_RESERVED*

Содержит статистическую информацию об области SHARED_POOL. Некоторые столбцы имеют смысл, только если был установлен параметр инициализации SHARED_POOL_RESERVED_SIZE.

**$VPD_POLICY*

Перечисляет все политики безопасности и предикаты, сопоставленные курсорам, находящимся в настоящий момент в библиотечном кэше (VPD – это сокращение для Virtual Private Database, виртуальных частных баз данных, о которых мы говорили в главе 4. VPD могут применяться для реализации абсолютно отдельных логических баз данных внутри одной базы данных Oracle с помощью политик безопасности на основе детального контроля доступа).

SQL

Обработку всех команд SQL для экземпляра описывают следующие представления:

**$OBJECT_DEPENDENCY*

Перечисляет все объекты, от которых зависят пакет, процедура или курсор в SGA. Соединив это представление с представлениями *$SQL и *$SESSION, можно получить список всех объектов, на которые ссылается пользователь.

**$OPEN_CURSOR*

Перечисляет все открытые курсоры системы.

*$SORT_SEGMENT

Предоставляет информацию обо всех сегментах сортировки табличных пространств, описанных как TEMPORARY.

*$SORT_USAGE

Предоставляет информацию о сегментах сортировки всех табличных пространств.

*$SQL

Предоставляет информацию обо всех командах SQL в разделяемой области SQL.

*$SQL_BIND_DATA

Предоставляет информацию о переменных связывания для всех команд SQL.

*$SQL_BIND_METADATA

Предоставляет метаданные для всех переменных связывания, используемых в командах SQL.

*$SQL_CURSOR

Предоставляет отладочную информацию о каждом курсоре в разделяемой области SQL.

*$SQL_PLAN

Выводит план выполнения последних команд SQL. Информация аналогична получаемой от команды EXPLAIN PLAN, только в данном случае выводится фактически применяемый план, а не описание плана, который, вероятно, будет использован. Появилось в Oracle9*i*.

*SQL_REDIRECTION

Перечисляет команды SQL, которые были перенаправлены, а также причины такого перенаправления (перезапись запроса с материализованными представлениями или ссылка на неразрешенный объект). Появилось в Oracle9*i*.

*$SQL_SHARED_MEMORY

Предоставляет информацию о выделении памяти для каждого курсора в разделяемой области SQL.

*$SQL_WORKAREA

Предоставляет информацию о расходовании ресурсов для рабочих областей, используемых курсорами SQL. Появилось в Oracle9*i*.

*$SQL_WORKAREA_ACTIVE

Предоставляет мгновенную информацию об активных в настоящий момент рабочих областях. Сведения, предлагаемые данным представлением, несколько отличаются от информации предыдущего представления. Появилось в Oracle9*i*.

*$SQLAREA

Предоставляет информацию обо всех командах SQL в разделяемой области SQL. Доступно после открытия базы данных.

*$SQLTEXT

Приводит текст всех команд SQL в разделяемой области SQL. Все символы табуляции и разделители строк заменяются на пробелы.

*$SQLTEXT_WITH_NEWLINES

Приводит текст всех команд SQL в разделяемой области SQL, при этом сохраняются все исходные символы табуляции и разделители строк.

Прямой маршрут SQL*Loader

Следующие представления содержат информацию о текущей работе прямого маршрута SQL*Loader:

**$LOADCSTAT*

Представление, в котором сервер Oracle хранит статистическую информацию о количестве строк, обработанных в процессе текущей загрузки. Однако, т. к. обратиться к представлению в процессе загрузки невозможно, ответом на любой запрос представления будет «данных не найдено».

**$LOADPSTAT*

Представление, с помощью которого SQL*Loader отслеживает статистическую информацию для текущей прямой загрузки. Поскольку информация относится лишь к текущему сеансу, то в ответ на запрос пользователя всегда будет возвращено ноль строк.

**$LOADTSTAT*

Представление, в котором сервер Oracle хранит дополнительные статистические данные о количестве строк, отброшенных при текущей загрузке. Однако обратиться к представлению в процессе загрузки невозможно, поэтому ответом на любой запрос представления будет «данных не найдено».

Системное окружение

Представления этого раздела предлагают разнообразные сведения о текущем системном окружении:

**$DB_PIPES*

Приводит данные о каналах базы данных, в настоящее время определенных в БД.

**$DBLINK*

Приводит информацию обо всех открытых в настоящий момент связях БД.

**$FILESTAT*

Предоставляет информацию о текущем статусе чтения/записи для всех файлов данных.

**$FIXED_TABLE*

Перечисляет все определенные в ядре таблицы *$ и X$.

**$FIXED_VIEW_DEFINITION*

Приводит определение каждого динамического представления производительности, построенного на основе таблиц X$.

**$GLOBAL_TRANSACTION*

Предоставляет информацию обо всех текущих глобальных транзакциях.

**$INDEXED_FIXED_COLUMN*

Перечисляет все столбцы индексов для таблиц, содержащихся в *$FIXED_TABLE.

**$RESOURCE_LIMIT*

Описывает текущее использование системных ресурсов, которое может быть указано в файле инициализации. Показывает исходное распределение, текущее использование и максимально разрешенные величины для каждого ресурса.

$ROLLNAME

Перечисляет имена всех сегментов отката. Доступно после открытия базы данных.

$ROLLSTAT

Приводит статистику для каждого сегмента отката.

$SYSSTAT

Приводит текущие значения всех системных статистик, определенных в *$STATNAME.

$SYSTEM_CURSOR_CACHE

Предоставляет информацию об использовании курсоров для всех сеансов.

$SYSTEM_EVENT

Приводит данные о времени, потраченном на ожидание каждого из системных событий, определенных в *$EVENT_NAME.

$TIMER

Предоставляет доступ к датчику времени, который увеличивается каждую сотую долю секунды.

$TRANSACTION

Перечисляет все активные транзакции системы.

$TRANSACTION_ENQUEUE

Перечисляет все постановки в очередь, удерживаемые активными транзакциями системы.

$WAITSTAT

Приводит информацию о том, сколько раз и как долго система ожидала блоки данных каждого из классов.

II

Языки

Эта часть книги посвящена синтаксису, применяемому для построения команд и функций SQL, программ PL/SQL и Java-интерфейсов для Oracle. Она состоит из пяти глав:

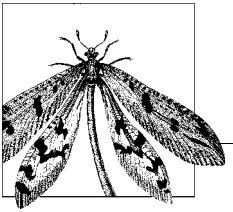
В главе 7 «SQL» описывается синтаксис Oracl-версии структурированного языка запросов (SQL – Structured Query Language).

Глава 8 «Функции» посвящена синтаксису функций, которые могут вызываться из SQL и PL/SQL.

В главе 9 «PL/SQL» рассмотрены возможности процедурного языка Oracle и описан формат всех его операторов.

Глава 10 «Пакеты PL/SQL» содержит перечень спецификаций заголовков всех процедур и функций встроенных пакетов Oracle, а также описания параметров.

В главе 11 «Java и Oracle» рассказывается о Java-интерфейсах к СУБД Oracle, в том числе о драйверах Java для Oracle, о сопоставлении типов данных Java и Oracle и о синтаксисе интерфейсов SQLJ и JDBC.



7
SQL

Структурированный язык запросов (Structured Query Language – SQL) за годы своего развития превратился в стандартный язык для работы с реляционными базами данных. В то время как большинство производителей СУБД реализуют собственные версии SQL, корпорации Oracle удается сохранять неплохую совместимость со стандартом ANSI SQL, хотя и она, безусловно, вносит свою долю расширений и усовершенствований с каждым новым выпуском своего продукта.

Команды SQL делятся на две категории: команды языка определения данных (Data Definition Language – DDL) и команды языка манипулирования данными (Data Manipulation Language – DML). Язык DDL применяется для управления структурой базы данных Oracle. Команды DDL позволяют создавать, изменять и удалять объекты любых типов, допустимых в базе данных. Многие из этих команд требуют специальных привилегий и обычно используются администраторами баз данных.

В отличие от них, команды DML позволяют обращаться к данным, хранимым в базе данных, и манипулировать ими. С их помощью можно добавлять, изменять, удалять и читать данные, а также контролировать то, как Oracle работает с данными.

Эта глава содержит краткий справочник по командам обоих типов с описанием их синтаксиса. Общие ключевые слова и идентификаторы, приведенные в первом разделе, относятся к обоим типам команд.

Общие ключевые слова и идентификаторы

Многие ключевые слова употребляются в различных командах SQL. Для экономии места такие ключевые слова собраны в одном разделе и не повторяются в описании каждой команды, где они могут встретиться.

Кроме того, имейте в виду, что описание типов данных приведено в приложении A; о выражениях, операторах и условиях речь пойдет в приложении B; элементам числового формата будет посвящено приложение C, а датам – приложение D.

псевдоним

Дополнительное имя, по которому можно ссылаться на объект, например таблицу.

столбец

Имя столбца таблицы или представления.

COMPRESS целое

Указывает, что при создании индекса разрешается *сжатие ключей* (*key compression*); параметр *целое* задает количество префиксных столбцов, которые следует сжать.

дата

Указывает дату в формате даты Oracle. Формат даты должен соответствовать установленному для базы данных формату даты по умолчанию (см. приложение А).

связь_БД

Имя связи базы данных (dblink), которая обеспечивает доступ к объектам схемы, расположенным в удаленной БД.

выражение

Любое разрешенное выражение Oracle, обычно включающее в себя один или несколько столбцов таблицы, представления или моментальной копии. Подробные сведения о выражениях приведены в приложении В.

имя_файла

Имя файла в формате, определяемом операционной системой.

имя_индекса

Имя индекса, перед которым может стоять имя владельца схемы.

целое

Любое число без десятичной точки и дробной части.

LOGGING

Указывает, что в процессе создания объекта будет вестись запись в журнал. Это значение устанавливается по умолчанию.

NOCOMPRESS

Указывает, что при создании индекса сжатие ключей будет отключено. Принимается по умолчанию.

NOLOGGING

Указывает, что в процессе создания объекта не будет вестись запись в журнал. Это ускоряет процесс создания, но в случае сбоя базы данных не будет возможности восстановить операцию создания при помощи журнальных файлов. Объекты придется создавать повторно.

NOPARALLEL

Указывает, что операция должна выполняться последовательно. Принимается по умолчанию.

OR REPLACE

Указывает, что если создаваемый объект уже существует, то его следует удалить, а затем создать повторно, не считая его наличие ошибкой.

PARALLEL [целое]

Указывает, что сервер Oracle будет использовать степень параллелизма, равную количеству процессоров, доступных всем существующим экземплярам, умноженному на значение параметра инициализации PARALLEL_THREADS_PER_CPU. Если задан параметр *целое*, то он определяет степень параллелизма.

имя_раздела

Имя раздела.

схема

Указывает имя схемы – учетную запись пользователя Oracle, владеющего объектом. Если при ссылке на объект имя схемы не указывается, то в качестве схемы по умолчанию будет выступать тот пользователь Oracle, под именем которого открыт текущий сеанс.

имя_подраздела

Имя подраздела.

имя_таблицы

Имя таблицы, перед которым может стоять имя владельца схемы.

табличное_пространство

Имя табличного пространства.

Общие инструкции SQL

Инструкция (*clause*) – это набор ключевых слов, связанных друг с другом и используемых как единое целое в ряде команд SQL. В разделе будут приведены наиболее распространенные инструкции SQL. Далее в главе, когда речь пойдет о командах, мы будем просто указывать имена инструкций и их синтаксическое расположение в описаниях форматов команд SQL.

Инструкция_авторасширения

```
AUTOEXTEND
{OFF
| ON [NEXT целое [K | M]] [MAXSIZE {целое [K | M] | UNLIMITED}]
}
```

Указывает, разрешено ли для файла увеличение его размера. Если разрешено, инструкция также может задавать параметры такого увеличения.

Ключевые слова

OFF Указывает, что авторасширение не поддерживается и увеличение размера файла не будет разрешено.

ON Указывает, что когда файл будет заполнен и потребуется дополнительное пространство, размер файла будет увеличен в соответствии со значением параметра NEXT до достижения предельного значения, определяемого параметром MAXSIZE.

NEXT Задает объем пространства в байтах, килобайтах (K) или мегабайтах (M), которое будет добавлено к файлу при его заполнении.

MAXSIZE Указывает максимальный размер файла в байтах, килобайтах (K) или мегабайтах (M). Увеличение размера файла сверх этого предела невозможно.

UNLIMITED Показывает, что максимальный размер файла определяется меньшей из следующих величин: доступный объем физического диска и максимальный размер файла, установленный в ОС.

Инструкция_ограничения_столбца

```
[CONSTRAINT имя_ограничения]
{ {NULL | NOT NULL}
| {UNIQUE | PRIMARY KEY (столбец[,столбец ...])}
| REFERENCES [схема.]имя_таблицы [(столбец[,столбец ...])
  [ON DELETE {CASCADE | SET NULL}]
| CHECK (условие)
}
[Инструкция_состояния_ограничения]
```

Задает ограничение целостности для столбца.

Ключевые слова

CONSTRAINT

Определяет имя ограничения. Если оно не указано, сервер Oracle присваивает ограничению имя в формате SYS_C*nnn*, где *nnn* – целое число.

NULL

Указывает, что столбец может содержать значение NULL.

NOT NULL

Указывает, что столбец не может содержать значение NULL.

UNIQUE

Указывает, что содержащиеся в данном столбце значения не могут дублироваться, т. е. значение данного столбца должно быть уникальным для всех строк таблицы. Однако обратите внимание, что в таком столбце несколько строк могут содержать значение NULL.

PRIMARY KEY

Означает, что столбец или комбинация столбцов служит первичным ключом той таблицы, в которой они определены. Столбец первичного ключа не может содержать значения NULL, и ни одно из значений первичного ключа не может дублироваться в другой строке таблицы (другими словами, значение ключа должно быть уникальным).

REFERENCES

Означает, что на таблицу и столбец (столбцы) ссылается внешний ключ. Комбинация столбцов, на которые ссылается внешний ключ, должна обладать ограничениями PRIMARY KEY или UNIQUE.

ON DELETE CASCADE

Указывает, что при удалении строки, содержащей уникальный или первичный ключ, строки, содержащие зависимые внешние ключи, также будут автоматически удалены.

ON DELETE SET NULL

Указывает, что при удалении строки, содержащей уникальный или первичный ключ, зависимым внешним ключам автоматически присваивается значение NULL.

CHECK

Задает условие, значение которого должно быть равно TRUE или NULL, для удовлетворения ограничения целостности.

Инструкция_состояния_ограничения

Определяет способ применения ограничения целостности к столбцу (см. следующий раздел).

Инструкция_состояния_ограничения

```
{DEFERRABLE [INITIALLY {IMMEDIATE | DEFERRED}
| NOT DEFERRABLE [INITIALLY IMMEDIATE ]
}
{ INITIALLY IMMEDIATE  [[NOT] DEFERRABLE]
| INITIALLY  DEFERRED
}
[RELY | NORELY]
[USING INDEX
  [INITRANS целое]
  [MAXTRANS целое]
  [PCTFREE  целое]
  [TABLESPACE табличное_пространство]
  [NOSORT]
  [LOGGING | NOLOGGING]
  [Инструкция_хранения]
  [ENABLE | DISABLE]
  [VALIDATE | NOVALIDATE]
  [EXCEPTIONS INTO [схема.]имя_таблицы]
]
```

Позволяет избирательно применять и не применять ограничение целостности.

Ключевые слова

DEFERRABLE

Указывает, что ограничение целостности можно отложить. Отложенное ограничение целостности будет наложено только после фиксации транзакции.

INITIALLY IMMEDIATE

Указывает, что изначально ограничение целостности должно вычисляться непосредственно после исполнения каждой команды DML. Такое поведение внутри транзакции можно изменить при помощи команды SET CONSTRAINTS.

INITIALLY DEFERRED

Указывает, что изначально ограничение целостности должно вычисляться только при фиксации транзакции. Такое поведение внутри транзакции можно изменить посредством команды SET CONSTRAINTS.

NOT DEFERRABLE

Означает, что для данного ограничения целостности проверка не может быть отложена.

RELY

Означает, что материализованное представление (или моментальная копия) будет считаться пригодным для перезаписи запроса, даже если соответствующее ограничение целостности не подтверждено. Ключевое слово действует только для материализованных представлений.

NORELY

Означает, что материализованное представление (или моментальная копия) не будет считаться пригодным для перезаписи запроса, если соответствующее ограничение целостности не подтверждено. Ключевое слово действует только для материализованных представлений.

USING INDEX

Указывает, что для проверки достоверности ограничения первичного и уникального ключа будет использоваться индекс.

INITRANS

Указывает начальное количество *входов транзакции* (*transaction entries*), выделенных в каждом блоке данных индекса. Вход необходим для каждой конкурентной транзакции, обновляющей блок.

MAXTRANS

Определяет наибольшее количество входов транзакции, которое в каждом блоке данных может быть выделено для данного индекса. Данное ключевое слово применяется для ограничения количества конкурентных транзакций, которые могут обновлять блок. Значение по умолчанию зависит от размера блока.

PCTFREE

Указывает, какая часть пространства каждого блока индекса (в процентах) зарезервирована для обновления значений индекса.

TABLESPACE

Определяет имя табличного пространства, в котором будет храниться индекс для ограничения целостности.

NOSORT

Указывает, что строки данных хранятся в базе данных в порядке возрастания, поэтому при создании индекса сортировка не требуется.

Инструкция_хранения

Указывает параметры хранения индекса, используемого для обеспечения выполнения ограничения целостности (см. далее раздел «Инструкция_хранения»).

ENABLE

Указывает, что ограничение целостности должно незамедлительно применяться ко всем новым данным таблицы или представления.

DISABLE

Указывает, что ограничение целостности не будет применяться к данным таблицы или представления.

VALIDATE

Указывает, что существующие данные таблицы или представления будут проверяться на соответствие ограничению целостности.

NOVALIDATE

Указывает, что существующие данные таблицы или представления не будут проверяться на соответствие ограничению целостности.

EXCEPTIONS INTO

Определяет таблицу, в которую сервер Oracle будет помещать идентификаторы строк, которые нарушают ограничение целостности. Эта таблица должна уже су-

ществовать (она может быть создана при помощи сценария *ultexcpt1.sql*). Имейте в виду, что если вы планируете использовать EXCEPTIONS INTO, необходимо задать и VALIDATE.

Инструкция_атрибутов_индекса

```
[Инструкция_физических_атрибутов]
[Инструкция_хранения]
[{LOGGING | NOLOGGING}]
[ONLINE]
[COMPUTE STATISTICS]
[TABLESPACE {табличное_пространство | DEFAULT}]
[{COMPRESS целое | NOCOMPRESS}]
[{NOSORT | REVERSE}]
[{PARALLEL | NOPARALLEL}]
```

Задает физические и логические характеристики индекса.

Ключевые слова

Инструкция_физических_атрибутов

Определяет физические атрибуты индекса (см. далее раздел «Инструкция_физических_атрибутов»).

Инструкция_хранения

Указывает параметры хранения индекса (см. далее раздел «Инструкция_хранения»).

ONLINE

Указывает, что в процессе создания индекса разрешено выполнение операций DML над индексируемой таблицей.

COMPUTE STATISTICS

Указывает, что в процессе создания индекса будет собираться статистическая информация.

TABLESPACE

Определяет имя табличного пространства, в котором будет храниться индекс. Если TABLESPACE не задано, то будет задействовано табличное пространство, определенное по умолчанию для владельца схемы. Можно использовать ключевое слово DEFAULT.

NOSORT

Указывает, что индексируемые строки были загружены в порядке возрастания и не требуют сортировки при создании индекса.

REVERSE

Указывает, что байты блока индекса (за исключением ROWID) будут храниться в обратном порядке. Данное ключевое слово нельзя употреблять вместе с NOSORT.

Инструкция_параметров_LOB

```
[TABLESPACE табличное_пространство]
[{ENABLE | DISABLE} STORAGE IN ROW]
[Инструкция_хранения]
```

```
[CHUNK целое]
[RETENTION]
[FREEPOOLS целое]
[PCTVERSION целое ]
[CACHE READS | NOCACHE [LOGGING | NOLOGGING]]
```

Определяет параметры хранения сегментов данных больших объектов (LOB).

TABLESPACE

Указывает имя табличного пространства, в котором будет храниться LOB.

ENABLE STORAGE IN ROW

Указывает, что данные LOB могут храниться в строке данных, если их размер не превышает приблизительно 4000 байт. Это принятое по умолчанию поведение.

DISABLE STORAGE IN ROW

Указывает, что данные LOB всегда хранятся вне строки данных.

Инструкция_хранения

Указывает параметры хранения сегмента LOB (см. далее раздел «Инструкция_хранения»).

CHUNK

Указывает, что для работы с LOB должно быть выделено указанное количество байт. Обратите внимание, что параметр *целое* будет округлен в сторону возрастания до числа, кратного размеру блока Oracle.

PCTVERSION

Указывает максимальный объем области хранения LOB (в процентах), резервируемой для создания новых версий LOB. Значение по умолчанию равно 10%.

CACHE READS

Указывает, что данные LOB будут удерживаться в оперативной памяти для ускорения доступа.

NOCACHE

Указывает, что данные LOB не будут удерживаться в оперативной памяти (поведение по умолчанию).

Инструкция_хранения_LOB

```
LOB
{(элемент_LOB[,элемент_LOB...]) STORE AS
 (Инструкция_параметров_LOB)
|(элемент_lob) STORE AS [(имя_сегмента_LOB)]
 (Инструкция_параметров_LOB)
}
```

Определяет, каким образом сегменты данных LOB будут сохраняться в таблице, разделе и подразделе.

Ключевые слова

LOB

Указывает, что параметры хранения задаются для перечисленных элементов LOB. Любой элемент LOB, не указанный в списке, будет использовать те же пара-

метры хранения, что и таблица, раздел или подраздел. Помните, что сервер Oracle автоматически создает индекс (управляемый системой) для каждого элемента LOB, который указан с данным ключевым словом.

STORE AS

Означает, что параметры хранения, требующие немедленного применения, указаны сразу после данного ключевого слова.

STORE AS имя_сегмента_LOB

Указывает имя сегмента LOB, которое может употребляться только при указании отдельного элемента LOB.

Инструкция_раздела

```
{PARTITION BY RANGE (столбец[,столбец...])
 (PARTITION [имя_раздела]
   VALUES LESS THAN ({значение | MAXVALUE}[,({значение | MAXVALUE}...])
   [Инструкция_хранения_LOB]
   [инструкция_физических_атрибутов]
   [TABLESPACE табличное_пространство]
   [LOGGING | NOLOGGING]
   [{ SUBPARTITIONS целое [STORE IN (табличное_пространство[,табличное_пространство...]])
    | (SUBPARTITION [имя_подраздела] [TABLESPACE табличное_пространство]
       [Инструкция_хранения_LOB]
       [, SUBPARTITION [имя_подраздела] [TABLESPACE табличное_пространство]
         [Инструкция_хранения_LOB] ...])
     }
   [,(PARTITION [имя_раздела]
     VALUES LESS THAN ({значение | MAXVALUE}[,({значение | MAXVALUE}...)
     [Инструкция_хранения_LOB]
     [Инструкция_физических_атрибутов]
     [TABLESPACE табличное_пространство]
     [LOGGING | NOLOGGING]
     [{ SUBPARTITIONS целое [STORE IN (табличное_пространство[,табличное_пространство ...])
      | (SUBPARTITION [имя_подраздела] [TABLESPACE табличное_пространство]
         [Инструкция_хранения_LOB]
         [, SUBPARTITION [имя_подраздела] [TABLESPACE табличное_пространство]
           [Инструкция_хранения_LOB] ...])
      }
| PARTITION BY LIST (столбец)
  (PARTITIONS имя_раздела VALUES ({значение | NULL}
    [,{значение | NULL}...]
  [Инструкция_описания_раздела]
  [,PARTITIONS имя_раздела VALUES ({значение | NULL}
    [,{значение | NULL}...]
  [Инструкция_описания_раздела]...])

| PARTITION BY HASH (столбец[,столбец ...])
  [{PARTITIONS целое [STORE IN (табличное_пространство[,табличное_пространство ...])
   | (PARTITION [имя_раздела] [TABLESPACE табличное_пространство]
      [Инструкция_хранения_LOB]
     [, PARTITION [имя_раздела] [TABLESPACE табличное_пространство]
      [Инструкция_хранения_LOB]
  ...]
   }
}
```

Задает параметры секционирования, кроме того применяется для определения *диапазонного секционирования* (*range partitioning*), *комбинированного секционирования* (*composite partitioning*), *спискового секционирования* (*list partitioning*) или *секционирования по хеш-ключу* (*hash partitioning*).

Ключевые слова

PARTITION BY RANGE

Указывает, что таблица должна быть секционирована по диапазонам значений перечисленных столбцов.

PARTITION ... VALUES LESS THAN

Указывает имя раздела и одно или несколько значений (в точном соответствии со списком столбцов, указанным для ключевого слова PARTITION BY RANGE), которые обозначают максимальные значения для включения в раздел. Список значений может содержать ключевое слово MAXVALUE, которое представляет собой максимально возможное значение для указанного столбца.

Инструкция_хранения_LOB

Указывает параметры хранения LOB для данного раздела (см. раздел «Инструкция_хранения_LOB»).

Инструкция_физических_атрибутов

Указывает физические параметры данного раздела (см. раздел «Инструкция_физических_атрибутов»).

TABLESPACE

Указывает имя табличного пространства, в котором будет храниться раздел или подраздел.

SUBPARTITIONS

Указывает, что для данного раздела должно быть создано столько подразделов, сколько указано в параметре *целое*.

STORE IN

Задает имя (имена) табличного пространства (пространств), в котором(ых) будут создаваться подразделы. Появилось в Oracle8*i*.

SUBPARTITION

Указывает подразделы по их именам.

PARTITION BY LIST

Определяет, что разделы должны создаваться на основе явного списка значений. Появилось в Oracle9*i*.

PARTITION BY HASH

Указывает, что на основе предоставленного списка столбцов должно быть выполнено секционирование по хеш-ключу. Появилось в Oracle8*i*.

PARTITIONS

Указывает, что следует создать количество разделов, указанное в параметре *целое*.

PARTITION

Задает имя раздела. Если параметр *имя* не указан, *сервер* Oracle присваивает разделу имя в формате SYS_P*nnn*, где *nnn* – это порядковый номер из последовательности, поддерживаемой сервером.

Секционирование – это отдельно оплачиваемая возможность, доступная только в рамках Oracle Enterprise Edition.

Инструкция_описания_раздела

```
[Инструкция_физических_атрибутов]
[TABLESPACE табличное_пространство]
[LOGGING | NOLOGGING]
[COMPRESS целое | NOCOMPRESS]
[OVERFLOW [Инструкция_физических_атрибутов]
  [TABLESPACE табличное_пространство]
  [LOGGING | NOLOGGING]]
[{Инструкция_хранения_LOB| Инструкция_хранения_Varray}
  [,{Инструкция_хранения_LOB| Инструкция_хранения_Varray}...]]
[{SUBPARTITIONS целое [STORE IN (табличное_пространство[,табличное_пространство ...])]
 | SUBPARTITION [имя_подраздела] [TABLESPACE табличное_пространство]
   [OVERFLOW [TABLESPACE табличное_пространство]]
   [{Инструкция_хранения_LOB| Инструкция_хранения_Varray}
     [,{Инструкция_хранения_LOB| Инструкция_хранения_Varray}...]]
    [,SUBPARTITION [имя_подраздела] [TABLESPACE табличное_пространство]
    [OVERFLOW [TABLESPACE табличное_пространство]]
    [{Инструкция_хранения_LOB| Инструкция_хранения_Varray}
     [,{Инструкция_хранения_LOB| Инструкция_хранения_Varray}...]
    ...)]
 }]
```

Задает различные физические параметры хранения разделов.

Ключевые слова

Инструкция_физических_атрибутов

Указывает физические параметры данного раздела (см. раздел «Инструкция_физических_атрибутов»).

Инструкция_хранения_LOB

Указывает параметры хранения LOB для данного раздела (см. раздел «Инструкция_хранения_LOB»).

OVERFLOW

Указывает, что строки индексированной таблицы, для которой превышено значение PCTTHRESHOLD, должны быть помещены в сегмент, описанный в данной инструкции.

Инструкция_физических_атрибутов

```
[INITRANS целое]
[MAXTRANS целое]
[PCTFREE целое]
[PCTUSED целое]
```

Определяет характеристики объектов схемы, влияющие на использование пространства в блоке Oracle.

Ключевые слова

INITTRANS

Указывает количество входов транзакции, выделенных для каждого блока объекта. Разрешен диапазон значений от 1 до 255, при этом обычно не рекомендуется изменять значение по умолчанию, равное 1 (хотя для OLTP-системы с большим количеством транзакций полезным может оказаться значение порядка 4–8).

MAXTRANS

Задает максимальное количество конкурентных транзакций, которые могут обновлять блок объекта. Разрешен диапазон значений от 1 до 255, при этом обычно не рекомендуется изменять значение по умолчанию. Значение по умолчанию зависит от размера блока Oracle.

PCTFREE

Указывает участок памяти блока данных (в процентах), резервируемый для последующих обновлений объекта. Разрешен диапазон значений от 0 до 99, значение по умолчанию равно 10.

PCTUSED

Указывает объем памяти блока данных (в процентах), который сервер Oracle стремится держать заполненным. Разрешен диапазон значений от 0 до 99, значение по умолчанию равно 40. Обратите внимание, что данное ключевое слово не применяется к индексам.

Инструкция_хранения

```
STORAGE (
  [INITIAL целое [K | M]]
  [NEXT целое [K | M]]
  [MINEXTENTS целое]
  [MAXEXTENTS [целое | UNLIMITED] ]
  [PCTINCREASE целое ]
  [FREELISTS целое ]
  [FREELIST GROUPS целое ]
  [OPTIMAL [целое [K | M]]]
  [BUFFER_POOL {KEEP | RECYCLE | DEFAULT}]
 )
```

Определяет метод выделения места в табличном пространстве Oracle для отдельного объекта.

Ключевые слова

STORAGE

Задает физические параметры хранения объекта базы данных.

INITIAL

Указывает размер первого экстента для объекта БД в килобайтах (K) или мегабайтах (M). Если этот размер не кратен размеру блока БД, то значение будет округлено в сторону увеличения до кратного размеру блока БД.

NEXT

Указывает размер следующего экстента для объекта БД в килобайтах (K) или мегабайтах (M). Если этот размер не кратен размеру блока БД, то значение будет округлено в сторону увеличения до кратного размеру блока БД.

MINEXTENTS

Указывает количество экстентов, выделяемых при создании объекта БД. Минимальным значением является 1 (это значение по умолчанию). Исключение составляют сегменты отката, для которых минимальное значение (и значение по умолчанию) равно 2.

MAXEXTENTS

Задает максимальное количество экстентов, которые могут быть выделены объекту базы данных. Значение по умолчанию определяется размером блока БД. Если указано ключевое слово UNLIMITED, то допустимое количество экстентов не ограничено.

PCTINCREASE

Указывает, на сколько (в процентах) каждый следующий экстент больше, чем предыдущий. Значение по умолчанию равно 50, т. е. размер каждого следующего экстента в полтора раза превышает размер предыдущего.

Если параметр PCTINCREASE установлен, то могут возникнуть сложности с обработкой все увеличивающихся экстентов, поэтому, вероятно, лучше всего установить PCTINCREASE в 0 и соответствующим образом задать значения INITIAL и NEXT.

FREELISTS

Указывает количество списков свободных блоков в группе для данного объекта БД. Значение по умолчанию равно 1, максимальное значение зависит от размера блока БД. Для систем с высоким коэффициентом вставки записей наилучшим будет значение FREELISTS в диапазоне от 2 до 4.

FREELIST GROUPS

Указывает количество групп списков свободных блоков данного объекта БД. Значение по умолчанию равно 1. Данный параметр используется только в среде Oracle Parallel Server и Real Application Clusters.

OPTIMAL

Только для сегментов отката. Параметр *целое* задает оптимальный размер, которого сервер Oracle будет стараться придерживаться за счет освобождения неиспользованных выделенных экстентов сегментов отката. Если размер не указан, то размер сегмента отката никогда не будет уменьшен. Если инструкция OPTIMAL не применяется, то по умолчанию размер сегментов отката никогда не будет уменьшен.

BUFFER_POOL

Указывает, как объекты схемы сопоставляются пулам буферов:

KEEP

Объект будет сопоставлен пулу KEEP и по возможности навсегда останется в оперативной памяти.

RECYCLE

Объект будет сопоставлен пулу RECYCLE и удален из оперативной памяти сразу же, как только станет ненужным.

DEFAULT

Объект будет приписан к буферному пулу DEFAULT, в котором для повторного использования буферов задействован стандартный алгоритм LRU.

Помните, с пулами KEEP и RECYCLE можно работать, только если они настроены администратором БД.

Инструкция_ограничения_таблицы

```
[CONSTRAINT имя_ограничения]
{{UNIQUE | PRIMARY KEY} (столбец[,столбец ...])
| CHECK (условие)
| FOREIGN KEY (столбец[,столбец ...])
   REFERENCES [схема.] имя_таблицы [(столбец[,столбец ...])]
   [ON DELETE {CASCADE | SET NULL}]
}
[Инструкция_состояния_ограничения]
```

Определяет ограничения целостности для таблицы.

Ключевые слова

CONSTRAINT

Определяет имя ограничения. Если не указано, то сервер Oracle присваивает ограничению имя в формате SYS_C*nnn*, где *nnn* – целое число.

UNIQUE

Указывает, что значения в указанных столбцах не могут дублироваться, т. е. значение данного столбца(ов) должно быть уникальным для всех строк таблицы. Однако обратите внимание, что несколько строк могут содержать в данном столбце значение NULL.

PRIMARY KEY

Означает, что столбец или комбинация столбцов служит первичным ключом той таблицы, в которой они определены. Столбец первичного ключа не может содержать NULL, и ни одно из значений первичного ключа не может повторно появиться в какой-то другой строке таблицы.

CHECK

Задает условие, принимающее значение TRUE или NULL, для удовлетворения ограничения целостности.

FOREIGN KEY

Указывает, что один или несколько столбцов таблицы участвуют в таком отношении ссылочной целостности как внешний ключ.

REFERENCES

Указывает таблицу и столбец (столбцы), на которые ссылается данное ограничение внешнего ключа.

ON DELETE CASCADE

Указывает, что при удалении строки, содержащей уникальный или первичный ключ, зависимые внешние ключи также будут автоматически удалены.

ON DELETE SET NULL

Указывает, что при удалении строки, содержащей уникальный или первичный ключ, зависимым внешним ключам автоматически присваивается значение NULL.

Инструкция_состояния_ограничения

Определяет способ применения ограничения целостности к столбцу (см. раздел «Инструкция_состояния_ограничения»).

Инструкция_хранения_Varray

```
VARRAY элемент_varray
{ {[ELEMENT] IS OF [TYPE] (ONLY тип)
  | [NOT] SUBSTITUTABLE AT ALL LEVELS
  }
| STORE AS LOB
  { имя_сегмента_LOB
    [([TABLESPACE табличное_пространство]
     [{ENABLE | DISABLE} STORAGE IN ROW]
     [Инструкция_хранения]
     [CHUNK целое]
     [PCTVERSION целое ]
     [CACHE | NOCACHE [LOGGING | NOLOGGING]]
    )]
  | ([TABLESPACE табличное_пространство]
     [{ENABLE | DISABLE} STORAGE IN ROW]
     [Инструкция_хранения]
     [CHUNK целое]
     [PCTVERSION целое ]
     [CACHE | NOCACHE [LOGGING | NOLOGGING]]
    )
  }
}
```

Определяет параметры хранения массивов переменной длины VARRAY. Появилась в Oracle8*i*.

элемент_varray

Задает имя элемента VARRAY.

ELEMENT

Указывает, что тип элемента столбца коллекции, или атрибута, должен быть подтипом объявленного *типа*.

IS OF TYPE

Указывает, что тип столбца объекта должен быть подтипом объявленного *типа*.

NOT SUBSTITUTABLE AT ALL LEVELS

Указывает, что столбец объекта не может содержать экземпляров, соответствующих любому из его подтипов и что замена запрещена для всех атрибутов встроенных объектов и элементов встроенных вложенных таблиц, а также объектов типа VARRAY. Значение по умолчанию равно SUBSTITUTABLE AT ALL LEVELS.

STORE AS LOB

Указывает, что VARRAY будет храниться как LOB. За ключевым словом следуют применяемые параметры хранения.

TABLESPACE

Указывает имя табличного пространства, в котором будет храниться LOB.

ENABLE STORAGE IN ROW

Указывает, что данные LOB могут храниться в строке данных, если их размер не превышает 4000 байт. Это поведение по умолчанию.

DISABLE STORAGE IN ROW

Указывает, что данные LOB всегда хранятся вне строки данных.

Инструкция_хранения

Указывает параметры хранения сегмента LOB (см. раздел «Инструкция_хранения»).

CHUNK

Указывает, что для работы с LOB должно быть выделено указанное количество байт. Обратите внимание, что значение параметра *целое* будет округлено до числа, кратного размеру блока Oracle.

PCTVERSION

Указывает максимальный объем (в процентах) области хранения LOB, используемой для создания новых версий LOB. Значение по умолчанию равно 10%.

CACHE

Указывает, что данные LOB для ускорения доступа будут удерживаться в оперативной памяти.

NOCACHE

Указывает, что данные LOB не будут удерживаться в оперативной памяти (поведение по умолчанию).

Команды языка определения данных

Команды языка определения данных (Data Definition Language – DDL) применяются для создания и изменения объектов базы данных Oracle. Команд DDL очень много, они могут выполнять множество разнообразных функций, к тому же многие из них предлагают большой набор параметров. В этом разделе приведена справочная информация о синтаксисе DDL-команд SQL. Команды языка манипулирования данными (Data Manipulation Language – DML) описаны отдельно далее в этой же главе.

Часто бывает так, что несколько команд DDL работает с одним и тем же типом объекта. Например, CREATE TABLE, ALTER TABLE и DROP TABLE – это родственные команды DDL, работающие с таблицами Oracle. Для удобства изложения такие взаимосвязанные команды были сгруппированы вместе. Поэтому заголовки типа CREATE/ALTER/DROP TABLE означают, что в разделе изложен синтаксис всех трех упомянутых команд.

Многие команды DDL обычно применяются только администраторами баз данных и выполняются от имени DBA. Поэтому для выдачи многих команд необходимы специальные привилегии. Кроме того, некоторые команды применимы, только если установлены соответствующие компоненты Oracle, например, Partition Option или Real Application Clusters.

ALTER RESOURCE COST

```
ALTER RESOURCE COST
[CPU_PER_SESSION вес]
[CONNECT_TIME вес]
[LOGICAL_READS_PER_SESSION вес]
[PRIVATE_SGA вес]
```

Изменяет формулу, по которой вычисляется общая стоимость ресурсов сеанса. Эта стоимость может быть затем ограничена посредством параметра COMPOSITE_LIMIT в профиле пользователя.

Общая стоимость ресурсов вычисляется следующим образом: объем каждого ресурса, расходуемый в рамках сеанса, умножается на вес данного ресурса, затем полученные произведения складываются для всех четырех ресурсов. Результат измеряется в сервисных единицах стоимости ресурсов. Для выдачи команды необходимо активировать привилегию ALTER RESOURCE COST.

Ключевые слова

CPU_PER_SESSION

Указывает количество процессорного времени, израсходованного в течение сеанса (в сотых долях секунды).

CONNECT_TIME

Указывает общее время сеанса (в минутах).

LOGICAL_READS_PER_SESSION

Указывает количество блоков БД, прочитанных в течение сеанса, при этом учитываются блоки, прочитанные из оперативной памяти и с диска.

PRIVATE_SGA

Указывает объем оперативной памяти, который сеанс может выделить в разделяемом пуле SGA (в байтах). Применяется только при работе с многопоточным/разделяемым сервером и при выделении для сеанса закрытой области в SGA.

вес

Вес каждого ресурса (целое число).

ALTER SYSTEM

```
ALTER SYSTEM
{SET
 {[RESOURCE_LIMIT = TRUE | FALSE] |
  [GLOBAL_NAMES = TRUE | FALSE] |
  [MTS_SERVERS = целое] |
  [MTS_DISPATCHERS = 'протокол, целое'] |
  [LICENSE_MAX_SESSIONS = целое] |
  [LICENSE_SESSIONS_WARNING = целое] |
  [LICENSE_MAX_USERS = целое] |
 }
  [COMMENT 'текст']
  [DEFERRED]
```

```
    [SCOPE = {MEMORY | SPFILE | BOTH}]
    [SID = {'sid' | .}]
  | [RESET
   {[RESOURCE_LIMIT = TRUE | FALSE] |
    [GLOBAL_NAMES = TRUE | FALSE] |
    [MTS_SERVERS = целое] |
    [MTS_DISPATCHERS = 'протокол, целое'] |
    [LICENSE_MAX_SESSIONS = целое] |
    [LICENSE_SESSIONS_WARNING = целое] |
    [LICENSE_MAX_USERS = целое]
   }]
    [SCOPE = {MEMORY | SPFILE | BOTH}]
    [SID = {'sid' | *}]
  | [{ENABLE | DISABLE} RESTRICTED SESSION]
  | [FLUSH SHARED POOL]
  | [CHECKPOINT [{GLOBAL | LOCAL}]]
  | [CHECK DATAFILES {GLOBAL | LOCAL}]
  | [SWITCH LOGFILE]
  | [{ENABLE | DISABLE} DISTRIBUTED RECOVERY]
  | [QUIESCE RESTRICTED | UNQUIESCE]
  | [SHUTDOWN [IMMEDIATE] имя_диспетчера]
  | [REGISTER]
  | [{SET | RESET} имя_параметра
       [SCOPE={MEMORY | SPFILE | BOTH}] [SID = 'sid'[,sid...]]]
  | [SUSPEND | RESUME]
  | [KILL SESSION 'sid_целое, сеанс_целое' [IMMEDIATE]]
  | [DISCONNECT SESSION 'sid_целое, сеанс_целое '
      [POST_TRANSACTION | IMMEDIATE]
  | [ARCHIVE_LOG [THREAD целое]
    {[START [TO 'место_назначения']]
    | [STOP]
    | [SEQUENCE целое [TO 'место_назначения']]
    | [CHANGE целое [TO 'место_назначения']]
    | [CURRENT [TO 'место_назначения']]
    | [GROUP целое [TO 'место_назначения']]
    | [LOGFILE 'имя_файла' [TO 'место_назначения']]
    | [NEXT [TO 'место_назначения']]
    | [ALL [TO 'место_назначения']]
    }
  }
```

Производит динамическое изменение экземпляра базы данных.

Во всех случаях, кроме специально отмеченных в описаниях соответствующих ключевых слов, для того чтобы можно быть применить команду ALTER SYSTEM, база данных должна быть смонтирована и открыта.

Ключевые слова

RESOURCE_LIMIT

Указывает, будут ли налагаться ограничения на ресурсы (TRUE или FALSE).

GLOBAL_NAMES

Указывает, будет ли вводиться глобальное именование (TRUE или FALSE).

MTS_SERVERS

Изменяет минимальное количество разделяемых процессов многопоточного/разделяемого сервера.

MTS_DISPATCHERS

Изменяет количество диспетчерских процессов для указанного протокола. Для выполнения этой команды база данных должна быть открыта.

LICENSE_MAX_SESSIONS

Указывает максимальное количество сеансов, разрешенных для экземпляра. Значение 0 означает отсутствие ограничения.

LICENSE_SESSIONS_WARNING

Указывает максимальное количество сеансов, разрешенных для экземпляра, после которого в журнал предупреждений записывается соответствующее сообщение. Значение 0 означает отсутствие ограничения.

LICENSE_MAX_USERS

Указывает максимальное количество пользователей БД. Значение 0 означает отсутствие ограничения.

COMMENT

Задает строку комментария, сопоставленную данному изменению значения параметра. Если задать SPFILE, то этот комментарий появится в файле параметров инициализации для указания последнего изменения параметра. Появилось в Oracle9*i*.

DEFERRED

Указывает, что изменения, выполненные для параметра, будут действовать для последующих сеансов, но текущий сеанс продолжает использовать старое значение. Появилось в Oracle9*i*.

SCOPE

Определяет, когда изменение вступит в силу в зависимости от того, была ли база данных запущена с использованием файла параметров (*INITsid_ORA*) или файла параметров сервера (*SPFILE*). Если для запуска базы данных использовался файл серверных параметров, то значение по умолчанию равно BOTH. Появилось в Oracle9*i*.

MEMORY

Указывает, что изменение выполняется в оперативной памяти, вступает в силу незамедлительно и действует до тех пор, пока не остановлена база данных. Если база данных запускается с использованием файла параметров, это единственный доступный вариант.

SPFILE

Указывает, что изменение выполняется в файле параметров сервера и новые установки вступят в силу после остановки и перезапуска базы данных. Необходимо указывать SPFILE при изменении значения статического параметра.

BOTH

Указывает, что изменение выполняется в оперативной памяти и в файле параметров сервера. Новые установки вступают в силу незамедлительно, действуют до остановки базы данных и продолжают действовать после перезапуска.

SID

Задает SID экземпляра, для которого будет действовать значение. Укажите SID = '*sid*', если необходимо, чтобы сервер Oracle изменил значение параметра только экземпляра с определенным *sid*, или SID = *, если необходимо изменить значение параметра для всех экземпляров. Эта инструкция применяется только при работе с компонентом Real Application Cluster и появилась в Oracle9*i*.

ENABLE RESTRICTED SESSION

Разрешает доступ к экземпляру только пользователям с привилегией RESTRICTED SESSION. При выдаче такой команды база данных может быть размонтирована, смонтирована, открыта или закрыта.

DISABLE RESTRICTED SESSION

Разрешает доступ к экземпляру любому пользователю с привилегией CREATE SESSION. При выдаче такой команды база данных может быть размонтирована, смонтирована, открыта или закрыта.

FLUSH SHARED POOL

Очищает все данные в разделяемом пуле экземпляра. При выдаче такой команды база данных может быть размонтирована, смонтирована, открыта или закрыта.

CHECKPOINT

Вынуждает сервер Oracle создать глобальную или локальную контрольную точку. GLOBAL создает контрольную точку для всех экземпляров, которые открыли базу данных; LOCAL – только для того экземпляра, к которому вы подключены. При выдаче такой команды база данных может быть открыта или закрыта.

CHECK DATAFILES

Проверяет доступ к оперативным файлам данных. Если задано значение GLOBAL, то проверяет, все ли экземпляры, открывшие базу данных, могут обращаться к файлам данных. Если задано значение LOCAL, то проверяет то же самое, но только для того экземпляра, к которому вы подключены. При выдаче такой команды база данных может быть открыта или закрыта.

SWITCH LOGFILE

Вызывает переключение группы журнальных файлов.

ENABLE DISTRIBUTED RECOVERY

Указывает, что разрешено распределенное восстановление; в однопроцессной среде применяется для инициирования распределенного восстановления.

DISABLE DISTRIBUTED RECOVERY

Указывает, что распределенное восстановление запрещено.

QUIESCE RESTRICTED

Указывает, что база данных должна быть приостановлена, для того чтобы администратор БД мог выполнить операции, которые невозможно осуществить при наличии конкурентных транзакций, запросов или операций PL/SQL. Появилось в Oracle9*i*.

UNQUIESCE

Указывает, что база данных должна быть выведена из пассивного состояния. Появилось в Oracle9*i*.

DISCONNECT SESSION

Отключает указанный сеанс от базы данных посредством уничтожения выделенного серверного процесса или виртуального канала MTS.

POST_TRANSACTION

Указывает, что уже начавшиеся транзакции должны быть завершены, прежде чем сеанс будет отключен.

IMMEDIATE

Указывает, что сеанс должен быть отключен незамедлительно, без ожидания завершения транзакций. Если задано значение POST_TRANSACTION, то данное ключевое слово будет проигнорировано.

KILL SESSION

Завершает сеанс, используя SID и SERIAL# из представления V$SESSION. Если сеанс ожидает завершения какого-то действия (например, операции над удаленной базой данных), то сервер Oracle даст ему возможность завершиться, если только не указано ключевое слово IMMEDIATE.

SUSPEND

Указывает, что все операции ввода/вывода для всех экземпляров должны быть приостановлены до тех пор, пока не будет выдана команда ALTER SYSTEM RESUME. Для выдачи команды необходимо, чтобы все табличные пространства находились в режиме *горячего резервирования* (*hot backup*). Появилось в Oracle8*i*.

RESUME

Указывает, что обычные операции ввода/вывода должны быть возобновлены после выдачи команды ALTER SYSTEM SUSPEND.

ARCHIVE_LOG

Архивирует группы журнальных файлов вручную или разрешает/запрещает автоматическое архивирование.

SET имя_параметра

Указывает, что задается динамически изменяемый параметр инициализации. (Подробная информация о параметрах инициализации приведена в главе 2.) Появилось в Oracle9*i*.

SCOPE

Определяет, каким образом обрабатываются изменения динамических параметров инициализации:

MEMORY

Задает параметр для работающего экземпляра, но не изменяет *SPFILE*.

SPFILE

Изменяет значение этого параметра в *SPFILE*, но не применяет его к работающему экземпляру.

BOTH

Задает параметр для работающего экземпляра и сохраняет его в *SPFILE*.

SID

Задает SID экземпляра, для которого будет действовать новое значение при работе в среде Parallel Server или Real Application Clusters. Для того чтобы сервер Oracle изменил значение параметра для всех экземпляров, следует задать SID = *.

THREAD

Задает поток, содержащий группу журнальных файлов, которая должна быть заархивирована. Этот параметр необходим только при использовании компонентов Parallel Server или Real Application Clusters в параллельном режиме.

START

Включает автоматическое архивирование журнальных групп.

STOP

Отключает автоматическое архивирование журнальных групп.

SEQUENCE

Задает номер последовательности журнала для журнальной группы, которая будет архивироваться вручную. Для применения SEQUENCE база данных должна быть смонтирована, но может быть как открыта, так и закрыта.

CHANGE

Принудительно архивирует группу оперативных журнальных файлов, которая содержит запись с системным номером изменения (System Change Number – SCN), равным значению параметра *целое*. Если SCN относится к текущей журнальной группе, производится переключение журнала. База данных должна быть открыта.

CURRENT

Инициирует переключение журнала и архивирует текущую журнальную группу. База данных должна быть открыта.

GROUP

Принудительно архивирует группу оперативных журнальных файлов с указанным значением GROUP, которое можно получить из представления словаря данных DBA_LOG_FILES. База данных должна быть смонтирована, но может быть как открыта, так и закрыта.

LOGFILE

Принудительно архивирует группу оперативных журнальных файлов, которая содержит элемент, идентифицируемый посредством *имени_файла*. База данных должна быть смонтирована, но может быть как открыта, так и закрыта.

NEXT

Принудительно архивирует следующую группу оперативных журнальных файлов, которая заполнена, но еще не заархивирована. База данных должна быть смонтирована, но может быть как открыта, так и закрыта.

ALL

Принудительно архивирует все группы оперативных журнальных файлов, которые уже заполнены, но еще не заархивированы. База данных должна быть смонтирована, но может быть как открыта, так и закрыта.

Общие ключевые слова и инструкции: *имя_файла*, *целое*.

ASSOCIATE STATISTICS

```
ASSOCIATE STATISTICS WITH
{COLUMNS [схема.] имя_таблицы.столбец[, [схема.] имя_таблицы.столбец...]
| FUNCTIONS [схема.]функция[,[схема.]функция...]
```

```
| PACKAGES [схема.]пакет[,[схема.]пакет...]
| INDEXES [схема.]индекс[,[схема.]индекс...]
}
{USING [схема.]тип_статистики
| DEFAULT COST (стоимость_процессора,стоимость_ввода-вывода,стоимость_сети)
| DEFAULT SELECTIVITY избирательность_по_умолчанию
| TYPES [схема.]тип[,[схема.]тип...]
}
```

Определяет, как будет вычисляться статистика для указанных объектов базы данных. Для удаления статистики или разрыва связи с объектом предназначена команда DISASSOCIATE STATISTICS. Появилось в Oracle8*i*.

Ключевые слова

COLUMNS

Означает, что будет приведен список столбцов.

FUNCTIONS

Означает, что будет указана одна или несколько функций.

PACKAGES

Означает, что будет указан один или несколько пакетов.

INDEXES

Означает, что будет указан один или несколько индексов.

USING тип_статистики

Указывает тип назначаемой статистики.

DEFAULT COST

Указывает, что будут предоставлены значения по умолчанию для стоимостей процессорного времени, ввода/вывода и сетевых операций. Данное ключевое слово не действует, если использовано ключевое слово COLUMNS.

стоимость_процессора

Целое число, задающее стоимость процессорного времени для однократного выполнения или обращения.

стоимость_ввода/вывода

Целое число, задающее стоимость операции ввода/вывода для однократного выполнения или обращения.

стоимость_сети

Целое число, задающее стоимость сетевой операции для однократного выполнения или обращения.

DEFAULT SELECTIVITY избирательность_по_умолчанию

Задает целое число в диапазоне от 1 до 100, которое представляет собой избирательность по умолчанию (в процентах). Данное ключевое слово не действует, если указано ключевое слово COLUMNS.

TYPES

Указывает, что будет назначен один или несколько типов.

Общие ключевые слова и инструкции: *схема, имя_таблицы*.

AUDIT (Объекты схемы)

```
AUDIT {параметр_объекта[,параметр_объекта ...] | ALL}
ON {[схема.]имя_объекта | DIRECTORY имя_каталога | DEFAULT}
[BY SESSION [WHENEVER [NOT] SUCCESSFUL]
[BY ACCESS [WHENEVER [NOT] SUCCESSFUL]
```

Задает аудит для определенного объекта схемы.

Ключевые слова

параметр_объекта

Означает, что аудит будет проводиться для конкретной операции. Допустимы следующие операции: ALTER, AUDIT, COMMENT, DELETE, EXECUTE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT и UPDATE. Ключевое слово ALL подразумевает все перечисленные операции.

имя_объекта

Имя объекта схемы, для которого будет проводиться аудит.

DIRECTORY имя_каталога

Имя каталога для аудита.

DEFAULT

Устанавливает указанный параметр объекта в качестве значения по умолчанию для объектов, которые еще не были созданы.

BY SESSION

Приводит к тому, что сервер Oracle вносит одну запись для всех команд SQL одного типа, выданных в рамках одного сеанса.

BY ACCESS

Приводит к тому, что сервер Oracle вносит одну запись для каждой отслеживаемой команды.

WHENEVER SUCCESSFUL

Задает применение аудита только для успешно завершенных команд SQL.

WHENEVER NOT SUCCESSFUL

Задает применение аудита только для команд SQL, завершившихся со сбоем или с выдачей ошибки.

Общие ключевые слова и инструкции: *схема*.

AUDIT (команды SQL)

```
AUDIT {системный_параметр | параметр_ sql}[,{системный_параметр | параметр_sql ...}]
{[BY имя_пользователя[,имя_пользователя ...]]
| [BY прокси_сервер [ON BEHALF OF {ANY | имя_пользователя[,имя_пользователя]] }
}
{[BY SESSION] [WHENEVER [NOT] SUCCESSFUL]
| [BY ACCESS] [WHENEVER [NOT] SUCCESSFUL]
}
```

Задает параметры проведения аудита для отдельных команд SQL в последующих пользовательских сеансах. Записи аудита помещаются в журнал аудита, представляющий

собой таблицу базы данных, к которой можно обратиться, используя представления словаря данных. Аудит включается при помощи параметра AUDIT_TRAIL в файле инициализации (подробную информацию о параметрах аудита можно найти в главе 2).

Ключевые слова

системный_параметр

Указывает, что аудит должен проводиться для команд SQL, разрешенных указанной системной привилегией (см. главу 4).

параметр_sql

Определяет набор команд SQL, для которых будет проводиться аудит (см. главу 4).

BY имя_пользователя

Указывает, что аудит должен проводиться для команд SQL, выданных указанным пользователем.

BY прокси_сервер

Указывает, что аудит должен проводиться для команд SQL, выданных указанным прокси-сервером.

ON BEHALF OF ANY

Указывает, что аудит должен проводиться для команд SQL, выданных от имени любого пользователя.

ON BEHALF OF имя_пользователя

Указывает пользователя, от имени которого прокси-сервер выполняет указанную команду.

BY SESSION

Приводит к тому, что сервер Oracle вносит одну запись для всех команд SQL одного типа, выданных в рамках одного сеанса.

BY ACCESS

Приводит к тому, что сервер Oracle вносит одну запись для каждой отслеживаемой команды.

WHENEVER SUCCESSFUL

Задает применение аудита только для успешно завершенных команд SQL.

WHENEVER NOT SUCCESSFUL

Задает применение аудита только для команд SQL, завершившихся со сбоем или с выдачей ошибки.

CALL

```
CALL [схема.] [пакет.]{функция | процедура}[@связь_БД]
  (выражение[,выражение ...]) [INTO : переменная_базового_языка [[INDICATOR]
:индикаторная_переменная]]
```

Исполняет хранимую функцию или процедуру PL/SQL.

Ключевые слова

пакет	Имя пакета, содержащего функцию или процедуру.
функция	Имя функции, которая должна быть выполнена.

процедура	Имя процедуры, которая должна быть выполнена.
выражение	Аргумент для функции или процедуры.
INTO	Указывает (для функции) имя переменной базового языка, которая будет хранить возвращаемое значение.
INDICATOR	Указывает имя переменной, которая задает условие переменной базового языка.

Общие ключевые слова и инструкции: *связь_БД*, *выражение*, *схема*.

COMMENT

```
COMMENT ON
{TABLE [схема.]{ имя_таблицы | представление | моментальная_копия}
| COLUMN [схема.]{ имя_таблицы | представление | моментальная_копия}.столбец
}
IS 'текст'
```

Добавляет комментарий для таблицы, представления, моментальной копии или столбца словаря данных.

Ключевые слова

TABLE	Указывает, что комментарий будет сопоставлен таблице, представлению или моментальному снимку.
представление	Имя представления, для которого назначается комментарий.
моментальная_ копия	Имя моментальной копии, для которой назначается комментарий.
COLUMN	Указывает, что комментарий будет назначен столбцу.
текст	Фактический текст комментария. Этот текст будет записан в словарь данных.

Общие ключевые слова и инструкции: *схема*, *имя_таблицы*.

CREATE/ALTER/DROP CLUSTER

Синтаксис CREATE:

```
CREATE CLUSTER [схема.]имя_кластера
(столбец тип_данных[, столбец тип_данных ...])
[Инструкция_физических_атрибутов]
[SIZE целое [K | M]]
[TABLESPACE табличное_пространство]
[Инструкция_хранения]
[INDEX]
[[SINGLE TABLE] HASHKEYS целое HASH IS выражение]
[{PARALLEL [целое] | NOPARALLEL}]
[CACHE | NOCACHE]
```

Синтаксис ALTER:

```
ALTER CLUSTER [схема.]имя_кластера
[SIZE целое [K | M]]
[Инструкция_физических_атрибутов]
[Инструкция_хранения]
```

```
[PARALLEL [целое] | NOPARALLEL]
[ALLOCATE EXTENT
  (EXTSIZE целое [K | M] [DATAFILE 'имя_файла'] [INSTANCE целое])]
```

Синтаксис DROP:

```
DROP CLUSTER [схема.]имя_кластера
[INCLUDING TABLES]
[CASCADE CONSTRAINTS]
```

Создает, изменяет или удаляет кластер – объект схемы, который содержит одну или несколько таблиц, имеющих несколько общих столбцов. Кластеризация может повысить производительность и эффективность работы базы данных. При кластеризации общие столбцы сохраняются лишь единожды, а данные из всех таблиц обычно размещаются в смежных блоках.

Ключевые слова

имя_кластера

Имя кластера.

тип_данных

Тип данных столбца.

TABLESPACE

Указывает имя табличного пространства, в котором будет храниться кластер. Если параметр отсутствует, то используется табличное пространство по умолчанию для владельца схемы.

INDEX

Указывает, что будет создан индексированный кластер. Данное ключевое слово не действует для хеш-кластера.

SINGLE TABLE

Показывает, что указанный кластер относится к особому типу – он содержит всего одну таблицу. Появилось в Oracle8*i*.

HASHKEYS целое

Указывает, что хеш-кластер должен быть создан с количеством ключей, указанным в параметре *целое*.

HASH IS выражение

Задает выражение, выступающее в качестве функции хеширования для хеш-кластера.

CACHE

Указывает, что блоки, извлекаемые для данной таблицы, помещаются в ту часть кэша буферов, где при выполнении полного просмотра таблицы находятся последние по времени использования элементы.

NOCACHE

Указывает, что блоки, извлекаемые для данной таблицы, помещаются в ту часть кэша буферов, где при выполнении полного просмотра таблицы находятся последние по времени использования элементы. Это поведение по умолчанию.

EXTSIZE целое

Задает размер нового экстента в байтах, килобайтах (K) или мегабайтах (M).

DATAFILE

Указывает имя файла данных операционной системы для табличного пространства, содержащего данный кластер, который предназначен для хранения нового экстента. Если параметр отсутствует, файл данных выберет сервер Oracle.

INSTANCE

Делает новый экстент доступным указанному экземпляру, который задается параметром инициализации INSTANCE_NUMBER. Данный параметр может применяться только при работе в параллельном режиме.

INCLUDING TABLES

Указывает, что должны быть удалены все таблицы, принадлежащие кластеру.

CASCADE CONSTRAINTS

Указывает, что из таблиц, находящихся вне кластера, должны быть удалены все ограничения ссылочной целостности, которые ссылаются на первичные и уникальные ключи таблиц кластера.

Общие ключевые слова и инструкции: *столбец*, *целое*, *Инструкция_физических_атрибутов*, *Инструкция_хранения*.

CREATE/DROP CONTEXT

Синтаксис CREATE:

```
CREATE [OR REPLACE] CONTEXT пространство_имен USING [схема .] пакет
[INITIALIZED {EXTERNALLY | GLOBALLY } | ACCESSED GLOBALLY ]
```

Синтаксис DROP:

```
DROP CONTEXT пространство_имен
```

Создает или удаляет пространство имен для контекста. Появилась в Oracle8*i*.

Ключевые слова

OR REPLACE

Указывает, что существующее пространство имен контекста должно быть заменено.

пространство_имен

Имя создаваемого пространства имен.

пакет

Имя пакета PL/SQL, который определяет атрибуты контекста.

INITIALIZED

Указывает, что пространство имен контекста может быть инициализировано не сервером Oracle.

EXTERNALLY

Означает, что пространство имен может быть инициализировано при помощи интерфейса OCI при установке соединения с базой данных.

GLOBALLY

Означает, что пространство имен может быть инициализировано при помощи каталога LDAP при подключении к базе данных глобального пользователя.

ACCESSED GLOBALLY

Указывает, что любой контекст приложения, определенный в пространстве имен, доступен в рамках всего экземпляра. Такая настройка позволяет нескольким сеансам совместно использовать атрибуты приложения.

Общие ключевые слова и инструкции: *схема*.

CREATE CONTROLFILE

```
CREATE CONTROLFILE [REUSE] [SET] DATABASE имя_БД
LOGFILE [GROUP целое] имя_файла[,[GROUP целое] имя_файла ...]
{RESETLOGS | NORESETLOGS}
[MAXLOGFILES целое]
[MAXLOGMEMBERS целое]
[MAXLOGHISTORY целое]
[MAXDATAFILES целое]
[MAXINSTANCES целое]
[ARCHIVELOG | NOARCHIVELOG]
DATAFILE (имя_файла[,имя_файла ...])
CHARACTER SET набор_символов
```

Заново создает управляющий файл, позволяя изменить некоторые параметры.

Ключевые слова

REUSE

Указывает, что один или более управляющих файлов, определенных в файле инициализации, могут быть повторно использованы и перезаписаны. Если данное ключевое слово не указано или существует какой-то из управляющих файлов, имена которых приведены в *INIT.ORA* или *SPFILE*, возникает ошибка.

SET

Означает, что указанное *имя_БД* будет новым именем базы данных. Длина имени не должна превышать восемь символов.

DATABASE

Указывает имя базы данных. До тех пор пока не выдана команда SET, это должно быть текущее имя базы данных.

LOGFILE

Указывает элементы всех журнальных групп, при этом все они должны существовать.

RESETLOGS

Означает, что содержимое журнальных файлов, имена которых перечислены в инструкции LOGFILE, будет проигнорировано. Для каждого файла, приведенного в инструкции LOGFILE, должен быть указан параметр SIZE.

NORESETLOGS

Указывает, что все файлы, перечисленные в инструкции LOGFILE (которые должны представлять собой текущие журнальные файлы, а не восстановленные с резервных копий), будут использованы повторно с сохранением их исходных размеров.

MAXLOGFILES

Определяет максимальное количество журнальных групп, которые могут быть созданы для базы данных. Значение по умолчанию и максимальное значение за-

висят от операционной системы. Значение обязательно должно быть не меньше 2, желательно – не меньше 3.

MAXLOGMEMBERS

Определяет максимальное количество копий журнальных групп, которые могут существовать в базе данных. Минимальное количество равно 1, максимальное и значение по умолчанию зависят от операционной системы.

MAXLOGHISTORY

Определяет максимальное количество архивных журнальных групп для автоматического восстановления носителя в режиме Parallel Server или Real Application Clusters. Минимальное количество равно 1, максимальное и значение по умолчанию зависят от операционной системы.

MAXDATAFILES

Определяет максимальное количество файлов данных, которые могут быть созданы для базы данных. Минимальное количество равно 1, но ни в коем случае не должно быть меньше наибольшего количества файлов данных, созданных в БД.

MAXINSTANCES

Указывает максимальное количество одновременно работающих экземпляров, для которых база данных может быть смонтирована и открыта. Параметр действует только в среде Parallel Server/Real Application Clusters.

ARCHIVELOG

Указывает, что база данных будет работать в режиме ARCHIVELOG.

NOARCHIVELOG

Указывает, что база данных не будет работать в режиме ARCHIVELOG и что оперативные журнальные файлы будут использоваться повторно. Это поведение по умолчанию.

DATAFILE

Указывает имена всех файлов данных в БД, при этом все они должны существовать.

набор_символов

Указывает имя набора символов, используемого для создания базы данных (если он отличен от набора, определенного по умолчанию).

Общие ключевые слова и инструкции: *имя_файла, целое.*

CREATE/ALTER DATABASE

Синтаксис CREATE:

```
CREATE DATABASE [имя_БД]
CONTROLFILE [REUSE]
LOGFILE [GROUP целое] (имя_файла[,[GROUP целое] имя_файла ...])
[MAXLOGFILES целое]
[MAXLOGMEMBERS целое]
[MAXLOGHISTORY целое]
[MAXDATAFILES целое]
[MAXINSTANCES целое]
[ARCHIVELOG | NOARCHIVELOG]
[CHARACTER SET набор_символов]
[NATIONAL CHARACTER SET набор_символов]
```

```
[SET TIME_ZONE = '{{+ | -} hh:mi | регион_часовой_пояс}']
[USER SYS IDENTIFIED BY пароль]
[USER SYSTEM IDENTIFIED BY пароль]
[DEFAULT TEMPORARY TABLESPACE табличное_пространство [TEMPFILE имя_файла]
  [EXTENT MANAGEMENT LOCAL] [UNIFORM [SIZE целое [ K | M ]]]]
[UNDO TABLESPACE табличное_пространство [DATAFILE имя_файла [инструкция_авторасширения]
DATAFILE (имя_файла[,имя_файла ...])[ инструкция_авторасширения]]]
[EXTENT MANAGEMENT LOCAL]
```

Синтаксис ALTER :

```
ALTER DATABASE [имя_БД]
{ ARCHIVELOG | NOARCHIVELOG
| MOUNT [[STANDBY | CLONE] DATABASE]
| CONVERT
| OPEN
  [{READ WRITE [{RESETLOGS | NORESETLOGS}] [MIGRATE]
  | READ ONLY
  }]
| ACTIVATE [PHYSICAL | LOGICAL] STANDBY DATABASE
  [SKIP [STANDBY LOGFILE]]
| SET STANDBY DATABASE TO MAXIMIZE
  {PROTECTION | AVAILABILITY | PERFORMANCE}
| REGISTER [OR REPLACE] {PHYSICAL | LOGICAL }
  LOGFILE дескриптор_файла_журнала[,дескриптор_файла_журнала ...]
| START LOGICAL STANDBY APPLY [NEW PRIMARY dblink | INITIAL значение_SCN]
| {STOP | ABORT} LOGICAL STANDBY APPLY
| COMMIT TO SWITCHOVER TO {PHYSICAL | LOGICAL} {PRIMARY | STANDBY}
  [{WITH | WITHOUT} SESSION SHUTDOWN] [WAIT | NOWAIT]
}
{ RESET COMPATIBILITY
| CONVERT
| ENABLE [PUBLIC] THREAD целое
| DISABLE THREAD целое
| GUARD {ALL | STANDBY | NONE}
| RENAME GLOBAL_NAME TO база_данных[.домен[.домен ...]]
| CHARACTER SET набор_символов
| NATIONAL CHARACTER SET набор_символов
| DEFAULT TEMPORARY TABLESPACE табличное_пространство
| SET TIME_ZONE = '{{+ | -} hh:mi | регион_часовой_пояс}'
}
{ CREATE DATAFILE 'имя_файла'[,'имя_файла' ...] [AS имя_файла]
| DATAFILE 'имя_файла'[,'имя_файла' ...]
  {ONLINE | OFFLINE [DROP}
  | RESIZE целое [K | M]
  | END BACKUP
  | инструкция_авторасширения
  }
| TEMPFILE 'имя_файла'[,'имя_файла' ...]
  {ONLINE | OFFLINE [DROP]
  | RESIZE целое [K | M]
  | инструкция_авторасширения
  }
| RENAME FILE 'имя_файла' [,'имя_файла' ...] TO 'имя_файла' [,'имя_файла' ...]
{ ADD [STANDBY] LOGFILE [THREAD целое]
  [GROUP целое] имя_файла[,[GROUP целое] имя_файла ...]
| ADD [STANDBY] LOGFILE MEMBER 'имя_файла' [RESUSE] [,'имя_файла' [RESUSE] ...]
  TO дескриптор_файла_журнала[,дескриптор_файла_журнала ...]
| ADD SUPPLEMENTAL LOG DATA ({PRIMARY KEY | UNIQUE INDEX}
```

```
  [,{PRIMARY KEY | UNIQUE INDEX...]) COLUMNS
| DROP {GROUP целое | 'имя_файла'|('имя_файла','имя_файла'[,'имя_файла' ...])
  }
| DROP [STANDBY] LOGFILE MEMBER 'имя_файла'[,'имя_файла' ...]
| CLEAR [UNARCHIVED] LOGFILE дескриптор_файла_журнала[,дескриптор_файла_журнала ...]
  UNRECOVERABLE DATAFILE
}
{ CREATE STANDBY CONTROLFILE AS 'имя_файла' [REUSE]
| BACKUP CONTROLFILE TO {'имя_файла' [REUSE] | TRACE}
  [{RESETLOGS | NORESETLOGS}]
}
{ RECOVER [AUTOMATIC] [FROM 'местоположение']
  {[STANDBY] DATABASE
    [{ UNTIL {CANCEL | TIME дата | CHANGE целое }
     | USING BACKUP CONTROLFILE}
  | TABLESPACE табличное_пространство [, табличное_пространство]...
  | DATAFILE 'имя_файла' [, 'имя_файла']...
  | STANDBY
    {TABLESPACE табличное_пространство [, табличное_пространство]...
    | DATAFILE имя_файла' [, 'имя_файла']...
    }
    UNTIL [CONSISTENT  WITH] CONTROLFILE
  | LOGFILE 'имя_файла'
  }
    TEST
    ALLOW целое CORRUPTION
    NOPARALLEL | PARALLEL [целое]} |
    CONTINUE [DEFAULT] | CANCEL}
  | CONTINUE [DEFAULT]
  | CANCEL
  }
| RECOVER MANAGED STANDBY DATABASE
  { {[DISCONNECT [FROM SESSION]] [FINISH [NOWAIT]]
    | [TIMEOUT целое | NOTIMEOUT]
  }
| {NODELAY | DELAY целое | DEFAULT DELAY}
| [NEXT целое] |
| [EXPIRE целое | NO EXPIRE]
| [PARALLEL целое | NOPARALLEL]
| [THROUGH [THREAD целое] SEQUENCE целое]
  | [ALL ARCHIVELOG]
  | [{ALL | LAST | NEXT} SWITCHOVER]]
  }
  | CANCEL [IMMEDIATE] [NOWAIT] |
  | [DISCONNECT [FROM SESSION]]
      [PARALLEL целое | NOPARALLEL]
      [FINISH [SKIP [STANDBY LOG FILE]][{WAIT | NOWAIT}]]
 }
| END BACKUP
}
```

Создает или изменяет базы данных и задает соответствующие параметры.

Ключевые слова

имя_БД

Имя базы данных. Может быть длиной от одного до восьми символов и не должно совпадать с зарезервированным словом.

CONTROLFILE REUSE

Указывает, что один или более существующих управляющих файлов, определенных в файле инициализации, могут быть повторно использованы и перезаписаны. Если данное ключевое слово не указано и существует какой-то из управляющих файлов, перечисленных в *INIT.ORA* или *SPFILE*, генерируется ошибка. Если указанные параметры требуют, чтобы управляющий файл имел больший размер, чем текущий, команда не выполнится. При создании новой базы данных этот параметр обычно не используется.

LOGFILE

Задает имена для одного или нескольких создаваемых журнальных файлов.

MAXLOGFILES

Определяет максимальное количество журнальных групп, которые могут быть созданы для базы данных. Значение по умолчанию и максимальное значение зависят от операционной системы. Значение обязательно должно быть не меньше 2, желательно – не меньше 3.

MAXLOGMEMBERS

Определяет максимальное количество копий журнальных групп, которые могут существовать в базе данных. Минимальное количество равно 1, максимальное и значение по умолчанию зависят от операционной системы.

MAXLOGHISTORY

Определяет максимальное количество архивных журнальных групп для автоматического восстановления носителя в режиме Parallel Server или Real Application Clusters. Минимальное количество равно 1, максимальное и значение по умолчанию зависят от операционной системы.

MAXDATAFILES

Определяет максимальное количество файлов данных, которые могут быть созданы для базы данных. Минимальное количество равно 1, но ни в коем случае не должно быть меньше наибольшего количества файлов данных, когда-либо созданных в БД.

MAXINSTANCES

Указывает максимальное количество одновременно работающих экземпляров, для которых база данных может быть смонтирована и открыта. Параметр действует только в среде Parallel Server/Real Application Clusters.

ARCHIVELOG

Указывает, что база данных будет работать в режиме ARCHIVELOG, что означает, что журнальная группа должна быть заархивирована, прежде чем ее можно будет повторно использовать. Если группа не заархивирована, база данных будет остановлена до тех пор, пока архивирование не будет успешно выполнено. Такой режим необходим для восстановления носителя.

NOARCHIVELOG

Указывает, что журнальные группы не архивируются и могут быть сразу же повторно использованы сервером Oracle. Это поведение по умолчанию.

CHARACTER SET

Указывает, какой набор символов база данных будет использовать для хранения данных (например, US7ASCII или JA16SJIS). Набор символов не может быть из-

менен после создания базы данных. Возможные варианты и значение по умолчанию зависят от операционной системы.

NATIONAL CHARACTER SET

Определяет набор национальных символов, который будет использоваться для специальным образом обозначенных столбцов. Если не указан, то будет взят набор символов базы данных по умолчанию.

SET TIME_ZONE

Определяет часовой пояс базы данных. Величина *hh:mm* задает сдвиг (положительный или отрицательный) относительно всеобщего скоординированного времени (Universal Coordinated Time – UTC), которое является эквивалентом времени по Гринвичу (GMT). Существует и другая возможность: указать имя региона. Данное ключевое слово появилось в Oracle9*i*.

Для того чтобы увидеть список допустимых имен регионов, обратитесь к столбцу TZNAME динамического представления V$TIMEZONE_NAMES.

DEFAULT TEMPORARY TABLESPACE

Определяет временное табличное пространство, по умолчанию назначаемое пользователем. Если инструкция не указана, то в качестве табличного пространства по умолчанию будет выступать SYSTEM. Инструкция появилась в Oracle9*i*.

TEMPFILE имя_файла

Задает имя файла данных для временного табличного пространства. Инструкция появилась в Oracle9*i* Release 2.

EXTENT MANAGEMENT

Определяет, каким образом сервер Oracle будет заниматься распределением временного табличного пространства. Появилось в Oracle9*i*.

LOCAL

Указывает, что некоторая часть табличного пространства должна быть выделена для битового отображения. Все временные табличные пространства имеют локально управляемые экстенты, так что эта инструкция не обязательна.

UNIFORM

Задает размер экстентов временного табличного пространства (в байтах) и указывает, что все экстенты временного табличного пространства имеют одинаковый размер. По умолчанию все экстенты имеют размер 1 Мбайт.

UNDO TABLESPACE

Означает, что сервер Oracle создаст табличное пространство отката с именем *табличное_пространство*. Oracle будет работать с данными отката, используя данное табличное пространство отката. Появилось в Oracle9*i* Release 2.

DATAFILE

Задает имена всех файлов данных БД. Если не указано, то по умолчанию будет создан один файл данных для табличного пространства SYSTEM. В версии Oracle9*i* Release 2 можно применить команду EXTENT MANAGEMENT LOCAL для создания локально управляемого табличного пространства SYSTEM.

Дополнительные ключевые слова для ALTER

MOUNT STANDBY DATABASE

Указывает, что будет смонтирована резервная база данных. Появилось в Oracle9*i*.

MOUNT CLONE DATABASE

Указывает, что должен быть смонтирован клон базы данных. Появилось в Oracle9*i*.

CONVERT

Указывает, что словарь данных БД должен быть преобразован из Oracle7 в Oracle8 или Oracle8*i*.

OPEN READ WRITE

Указывает, что база данных открывается в режиме чтения/записи. Это поведение по умолчанию.

RESETLOGS

Заново устанавливает номер последовательности журнала в 1 и делает недействительными все записи журнала в существующих оперативных и архивных журнальных файлах. Этот параметр применяется только после выполнения неполного восстановления носителя или при открытии БД после выполнения восстановления носителя с помощью резервного управляющего файла. В противном случае применяется NORESETLOGS. Если база данных открывается с указанием ключевого слова RESETLOGS, следует незамедлительно выполнить полное резервное копирование БД.

NORESETLOGS

Не изменяет статус текущего номера последовательности журнала и записей журнала.

MIGRATE

Указывает, что должна быть проведена миграция базы данных с версии Release 7.3.4 на текущую версию Oracle. Если миграция осуществляется не с версии 7.3.4, можно применить команду SQL*Plus STARTUP MIGRATE. Появилось в Oracle9*i*.

READ ONLY

Указывает, что база данных будет открыта в режиме только для чтения, т. е. запросы к БД будут разрешены, а операции записи – запрещены. Появилось в Oracle8*i*.

ACTIVATE STANDBY DATABASE

Означает, что база данных переходит из резервного состояния в активное. Появилось в Oracle9*i*.

PHYSICAL

Указывает, что должна быть активирована физическая резервная база данных.

LOGICAL

Указывает, что должна быть активирована логическая резервная база данных.

SKIP

Означает, что сервер Oracle должен продолжать работу, даже если резервные журнальные файлы содержат информацию, которую можно восстановить при помощи команды RECOVER MANAGED STANDBY DATABASE.

SET STANDBY DATABASE TO MAXIMIZE

Задает уровень защиты базы данных. Появилось в Oracle9*i*.

PROTECTION

Устанавливает режим максимальной защиты.

AVAILABILITY

Устанавливает режим максимальной доступности.

PERFORMANCE

Устанавливает режим максимальной производительности.

REGISTER LOGFILE

Указывает, что журнальные файлы основной базы данных, вышедшей из строя, должны быть зарегистрированы вручную. В случае логической резервной БД эта команда позволяет установить исходную отправную точку для новой логической резервной БД. С этой целью в команде ALTER DATABASE START LOGICAL STANDBY APPLY INITIAL в качестве отправной точки используется самый поздний зарегистрированный журнальный файл. Появилось в Oracle9*i*.

START LOGICAL STANDBY

Указывает, что сервер Oracle начинает применять журналы к логической резервной базе данных.

NEW PRIMARY

Определяет новую основную базу данных после выполнения команды ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY или в случае, если резервная база данных завершила обработку журналов основной БД, то резервная БД становится основной.

INITIAL

Указывает, что журналы впервые применяются к резервной базе данных.

COMMIT TO SWITCHOVER

Указывает, что сервер Oracle выполняет *постепенное переключение* (*graceful switchover*), при котором текущая основная база данных получает статус резервной, а резервная становится основной. В среде Real Application Clusters все экземпляры, отличные от того, в котором выдается данная команда, обычно должны быть остановлены.

PHYSICAL

Подготавливает базу данных к работе в качестве физической резервной БД.

LOGICAL

Подготавливает базу данных к работе в качестве логической резервной БД. Если вы указываете LOGICAL, то затем необходимо выполнить команду ALTER DATABASE START LOGICAL STANDBY APPLY.

WITH SESSION SHUTDOWN

Указывает, что сервер Oracle во время выполнения данной команды должен остановить все открытые сеансы приложений и откатить все незавершенные транзакции.

WITHOUT SESSION SHUTDOWN

Указывает, что выполнить команду не удастся, если существуют открытые сеансы приложений. Это поведение по умолчанию.

RESET COMPATIBILITY

Указывает, что должна быть обеспечена совместимость с БД указанной версии. Изменение вступает в силу при следующем запуске БД.

ENABLE THREAD

Означает, что поток журнала разрешен в среде Parallel Server/Real Application Clusters. Если указано ключевое слово PUBLIC, то разрешенный поток доступен всем экземплярам. В противном случае он доступен лишь тому экземпляру, который специально его запросит.

DISABLE THREAD

Означает, что поток журнала не разрешен и не доступен ни одному экземпляру среды Parallel Server/Real Application Clusters.

GUARD

Указывает, что данные БД защищены от изменения. Значение ALL не разрешает пользователям, отличным от SYS, изменять любые данные в базе данных. Если значение равно STANDBY, то пользователи, отличные от SYS, не могут выполнять какие бы то ни было изменения в объекте БД, поддерживаемом логической резервной БД. Такая установка удобна, если требуется, чтобы операции построения отчетов могли изменять данные, не участвующие в логическом резервировании.

RENAME FILE 'имя_файла1' TO 'имя_файла2'

Означает, что в управляющем файле должно быть изменено имя файла данных, временного файла или файла журнала. Обратите внимание, что на файлы операционной системы данное ключевое слово не действует.

RENAME GLOBAL NAME TO

Указывает, что глобальное имя базы данных заменяется на предложенное значение, длина которого не должна превышать 8 символов.

CREATE DATAFILE

Указывает, что вместо старого (который мог быть утрачен без резервирования) должен быть создан новый пустой файл данных. Прежде чем использовать файл данных, необходимо выполнить восстановление носителя.

ONLINE

Указывает, что файл данных должен быть переведен в оперативный режим.

OFFLINE

Указывает, что файл данных должен быть переведен в автономный режим.

RESIZE

Указывает, что размер файла данных должен быть увеличен или уменьшен до заданного.

END BACKUP

Указывает, что восстановление носителя не будет выполняться в случае запуска базы данных после прерывания горячего резервирования. Появилось в Oracle9*i*.

TEMPFILE

Указывает, что временный файл данных будет изменен.

ONLINE

Файл данных должен быть переведен в оперативный режим.

OFFLINE

Файл данных должен быть переведен в автономный режим.

DROP

Для базы данных, работающей в режиме NOARCHIVELOG, файл данных должен быть переведен в автономный режим. Если база данных работает в режиме ARCHIVELOG, данное ключевое слово игнорируется.

Инструкция DROP не удаляет файл данных из БД. Для удаления файла необходимо удалить табличное пространство, в котором он хранится. Пока табличное пространство не удалено, файл данных остается в словаре данных.

ADD LOGFILE

Указывает, что должна быть добавлена одна или несколько групп журнальных файлов. В среде Parallel Server/Real Application Clusters можно указывать THREAD.

ADD LOGFILE MEMBER

Указывает, что в существующую журнальную группу будет добавлен новый элемент *имя_файла*. Если элемент с именем *имя_файла* уже существует и его надо создать заново, задайте значение REUSE.

дескриптор_файла_журнала

Определяет существующую журнальную группу как GROUP *целое* или в виде списка имен файлов.

DROP GROUP

Указывает, что вся журнальная группа должна быть удалена после выполнения команды ALTER SYSTEM SWITCH LOGFILE. Удаляемая группа может быть задана в виде GROUP *целое* или как список имен файлов.

DROP LOGFILE MEMBER 'имя_файла'

Указывает, что один или несколько отдельных элементов журнальной группы будут удалены.

CLEAR LOGFILE

Указывает, что оперативный журнал инициализируется повторно, возможно, без архивирования. Команда CLEAR LOGFILE аналогична добавлению и удалению журнала, за исключением того, что может быть выдана даже в том случае, если в потоке присутствуют всего два журнала, а также может выдаваться для текущего журнала закрытого потока. Если вы хотите повторно использовать журнал, который не был заархивирован, необходимо указать UNARCHIVED.

UNRECOVERABLE DATAFILE

Необходимо использовать, если файл данных был переведен в автономный режим, и база данных работает в режиме ARCHIVELOG (т. е. выдается ALTER DATABASE ... DATAFILE OFFLINE без ключевого слова DROP), и если незаархивированный журнал, который планируется очистить, необходим для восстановления файла данных перед переводом его обратно в оперативный режим. В этом случае необходимо удалить файл данных и все табличное пространство, как только завершится команда CLEAR LOGFILE.

CREATE STANDBY CONTROL FILE

Указывает, что для ведения резервной базы данных создается управляющий файл.

BACKUP CONTROL FILE TO

Указывает, что резервная копия текущего управляющего файла должна быть сохранена в файле *имя_файла*. Если указаны ключевые слова TO TRACE, то вместо создания резервной копии управляющего файла будет выполнена запись набора команд SQL в файл трассировки. Если указано ключевое слово RESETLOGS, то в число записанных команд будет входить ALTER DATABASE OPEN RESETLOGS. Если указано ключевое слово NORESETLOGS, то будет записана команда SQL ALTER DATABASE OPEN NORESETLOGS.

RECOVER FROM

Определяет, откуда будут считываться архивные журнальные файлы, необходимые для восстановления. Если указано ключевое слово AUTOMATIC, то имя следующего архивного журнального файла, необходимого для восстановления, будет сгенерировано при помощи параметров инициализации LOG_ARCHIVE_DEST и LOG_ARCHIVE_FORMAT.

DATABASE

Означает, что будет восстановлена вся база данных. Если указано ключевое слово STANDBY, то резервная база данных будет восстановлена при помощи управляющего файла и архивных журнальных файлов из основной БД. Можно указать ключевые слова UNTIL CANCELLED, которые означают, что база данных будет восстанавливаться до тех пор, пока операция не будет прервана инструкцией RECOVER CANCEL. Ключевое слово TIME задает восстановление с временным критерием – восстановление будет выполнено вплоть до указанного времени. Ключевое слово CHANGE показывает, что восстановление будет выполнено до указанного SCN (системного номера изменения).

USING BACKUP CONTROLFILE

Указывает, что вместо текущего будет использован резервный управляющий файл.

STANDBY TABLESPACE UNTIL CONSISTENT WITH CONTROLFILE

Указывает, что табличное пространство старой резервной базы данных будет восстановлено при помощи управляющего файла текущей резервной базы данных. Появилось в Oracle8*i*.

STANDBY DATAFILE UNTIL CONSISTENT WITH CONTROLFILE

Указывает, что файл данных старой резервной базы данных будет восстановлен при помощи управляющего файла текущей резервной базы данных. Появилось в Oracle8*i*.

TABLESPACE

Указывает, что должны быть восстановлены одно или несколько табличных пространств (все они должны быть автономными).

DATAFILE

Указывает, что должны быть восстановлены один или несколько файлов данных (все они должны быть автономными).

LOGFILE

Означает, что восстановление носителя должно продолжаться с использованием предложенного(ых) файла(ов) журнала.

TEST

Означает, что выполняется пробное восстановление.

ALLOW CORRUPTION

Задает допустимое количество поврежденных блоков в файле журнала. При превышении этого значения восстановление прерывается.

CONTINUE DEFAULT

Указывает, что восстановление должно продолжаться с использованием автоматически сгенерированного файла журнала. Данное ключевое слово эквивалентно RECOVER AUTOMATIC, за исключением того, что не предлагается ввести имя файла.

CONTINUE

Указывает, что восстановление нескольких экземпляров должно быть продолжено после остановки для отключения потока.

CANCEL

Означает завершение прерываемого восстановления.

RECOVER MANAGED STANDBY DATABASE

Задает режим *устойчивого восстановления* (*sustained recovery*) резервной базы данных. В этом режиме предполагается, что резервная база данных является активным компонентом. Появилось в Oracle8*i*.

DELAY

Задает период ожидания (в минутах) сервера Oracle перед применением архивных журналов. NODELAY означает, что журналы должны быть применены незамедлительно.

DEFAULT DELAY

Указывает, что период ожидания (в минутах) сервера Oracle перед применением архивных журналов равен значению параметра инициализации LOG_ARCHIVE_DEST_*n* для основной базы данных.

TIMEOUT

Задает период ожидания (в минутах) доступности архивного журнального файла для записи в резервную БД.

CANCEL

Прерывает устойчивое восстановление после применения текущего архивного файла журнала или после прочтения следующего (если указано ключевое слово IMMEDIATE) в зависимости от того, какое событие произойдет раньше.

DISCONNECT

Указывает, что в ходе управляемого восстановления архивные файлы журнала должны применяться отсоединенным фоновым процессом, не препятствуя выполнению других задач в текущем сеансе. (Ключевые слова FROM SESSION могут отсутствовать и применяются для большей понятности.) Появилось в Oracle9*i*.

NEXT

Указывает, что сервер Oracle должен применить заданное количество архивных журналов как можно скорее после их архивации. Параметр временно подменяет собой любые значения времени задержки, указанные в параметре инициализации LOG_ARCHIVE_DEST_*n* основной базы данных, а также любые значения DELAY из команды ALTER DATABASE ... MANAGED STANDBY RECOVERY. После того как заданное параметром *целое* число архивных жур-

налов обработано, параметры, определяющие время задержки, опять вступают в силу.

EXPIRE

Задает период времени в минутах начиная с текущего момента, по истечении которого управляемое восстановление будет завершено автоматически. Фактически процесс может завершиться по истечении данного периода времени, т. к. сервер Oracle доводит до конца обработку того журнального файла, который обрабатывался в момент окончания периода.

NOEXPIRE

Отменяет ранее выданную команду EXPIRE.

THROUGH

Указывает, когда следует завершить управляемое восстановление. Появилось в Oracle9*i*.

ALL ARCHIVELOG

Указывает, что сервер Oracle будет управлять процессом восстановления до тех пор, пока не будут применены все архивные журнальные файлы.

SWITCHOVER

Процесс управляемого восстановления обычно прекращается, встретив операцию *переключения резерва* (*switchover*), т. к. эта операция формирует индикатор окончания архивации журнала. Инструкция SWITCHOVER полезна в случае, если вы работаете с несколькими резервными БД, при этом все они, кроме одной, останутся резервными после переключения. Эта инструкция оставляет процесс архивного резервного восстановления работающим. В этом случае «вторичные» резервные базы данных ожидают получения потока журнала от новой первичной базы данных вместо того, чтобы остановить процесс восстановления, а затем запустить его вновь после активации новой основной базы данных.

Значение ALL сохраняет процесс управляемого восстановления работающим при всех операциях переключения. Значение LAST отменяет операции управляемого восстановления после последнего индикатора окончания архивации. Значение NEXT отменяет операции управляемого восстановления после того, как встречает следующий индикатор окончания архивации (поведение по умолчанию).

Общие ключевые слова и инструкции: *инструкция_авторасширения*, *имя_файла*, *целое*, *имя_табличного_пространства*.

CREATE/DROP DATABASE LINK

Синтаксис CREATE:

```
CREATE [SHARED] [PUBLIC] DATABASE LINK связь_БД
[CONNECT TO
  {CURRENT USER
  |  имя_пользователя IDENTIFIED BY пароль
    [AUTHENTICATED BY имя_пользователя IDENTIFIED BY пароль]
  }]
[USING 'строка_соединения']
```

Синтаксис DROP:

```
DROP [PUBLIC] DATABASE LINK связь_БД
```

Создает или удаляет связь базы данных, которая позволяет обращаться к объектам удаленной базы данных.

Ключевые слова

связь_БД

Определяет имя создаваемой связи БД. Это должно быть имя, разрешенное для объекта Oracle.

SHARED

Указывает, что при работе с многопоточным/разделяемым сервером одно сетевое соединение будет использоваться совместно.

PUBLIC

Указывает, что связь БД будет доступна всем пользователям. Если данное ключевое слово не указано, то связь является частной и доступна только одному пользователю.

CONNECT TO CURRENT USER

Указывает, что связь создается от имени текущего пользователя, который должен быть глобальным пользователем с действительной учетной записью на удаленной БД.

CONNECT TO имя_пользователя IDENTIFIED BY пароль

Указывает *имя_пользователя* и *пароль*, указываемые для подключения к удаленной БД.

AUTHENTICATED BY

Указывает имя пользователя и пароль доступа к удаленной БД, применяемые для аутентификации при указании ключевого слова SHARED.

USING

Указывает имя удаленной БД в формате Oracle Net Services (*строка_соединения*).

Если инструкция CONNECT TO пропущена, то при вызове связи БД будут указаны имя пользователя и пароль текущего пользователя, а не имя пользователя и пароль создателя.

CREATE/DROP DIMENSION

Синтаксис CREATE:

```
CREATE [FORCE | NOFORCE] DIMENSION [схема.]измерение
LEVEL уровень IS
  { таблица_уровня.столбец_уровня
  |(таблица_уровня.столбец_уровня, имя_таблицы _ имя_таблицы.столбец_уровня ...)
  }
HIERARCHY иерархия (уровень_потомка CHILD OF уровень_родителя)
[JOIN KEY {ключевой_столбец_потомка | (ключевой_столбец_потомка,ключевой_столбец_потомка
...)}
  REFERENCES уровень_родителя] |
ATTRIBUTE уровень DETERMINES
  { зависимый_столбец
  |( зависимый_столбец, зависимый_столбец...)
  }
```

Синтаксис DROP:

```
DROP DIMENSION [схема.]измерение
```

Создает или удаляет измерение, которое определяет отношение родитель-потомок между парами наборов столбцов. Появилось в Oracle9*i*.

Ключевые слова

FORCE

Означает, что измерение будет создано, даже если упоминаемые таблицы не существуют.

NOFORCE

Означает, что измерение будет создано, только если упоминаемые объекты уже существуют. Это поведение по умолчанию.

измерение

Имя измерения.

LEVEL

Задает имя уровня, определяющего иерархии и атрибуты измерения.

таблица_уровня.столбец_уровня

Указывает столбцы (до 32) для уровня.

HIERARCHY

Указывает имя иерархии.

уровень_потомка

Имя уровня, который находится с данным родительским уровнем в отношении «*n*-к-одному».

CHILD OF

Указывает имя родительского уровня.

JOIN KEY

Указывает имя столбца в условии JOIN для родительской таблицы.

REFERENCES

Задает имя родительского уровня.

ATTRIBUTE

Задает имя уровня или иерархии.

DETERMINES зависимый_столбец

Указывает перечень столбцов, определяемых для данного уровня иерархии.

Общие ключевые слова и инструкции: *столбец*, *схема*, *имя_таблицы*.

CREATE/DROP DIRECTORY

Синтаксис CREATE:

```
CREATE [OR REPLACE] DIRECTORY имя_каталога AS 'путь'
```

Синтаксис DROP:

```
DROP DIRECTORY имя_каталога
```

Создает или удаляет объект каталога, указывающий на каталог операционной системы для хранения объектов BFILE.

Сервер Oracle не проверяет, действительно ли каталог существует в операционной системе, так что будьте внимательны при указании пути.

Ключевые слова

OR REPLACE

Указывает, что данный объект каталога должен заменить любой существующий объект каталога с таким же именем.

имя_каталога

Имя каталога.

путь

Полное имя каталога операционной системы с учетом регистра.

CREATE/ALTER/DROP FUNCTION

Создает, изменяет или удаляет функцию PL/SQL в базе данных (сведения о синтаксисе и применении команды можно найти в главе 9).

CREATE/ALTER/DROP INDEX

Синтаксис CREATE:

```
CREATE [OR REPLACE] [UNIQUE | BITMAP] INDEX [схема.]имя_индекса ON
{ CLUSTER [схема.]кластер
     Инструкция_атрибутов_индекса
| [схема.] имя_таблицы [псевдоним] ({столбец | выражение_для_столбца} [ASC | DESC]
  [,{столбец|выражение_для_столбца} [ASC | DESC] ...])
  [{ { {GLOBAL PARTITION BY RANGE (список_столбцов)
          (PARTITION имя_раздела VALUES LESS THAN (значение[, значение...])
           [Инструкция_физических_атрибутов]
           [TABLESPACE табличное_пространство]
           [LOGGING | NOLOGGING]
       | LOCAL (PARTITION имя_раздела
          [Инструкция_физических_атрибутов]
          [TABLESPACE табличное_пространство]
          [LOGGING | NOLOGGING]
          [,PARTITION имя_раздела
           [Инструкция_физических_атрибутов]
           [TABLESPACE табличное_пространство]
           [LOGGING | NOLOGGING] ...])
       | LOCAL
         {STORE IN ({табличное_пространство[,табличное_пространство...]|DEFAULT})
         | (PARTITION [имя_раздела] [TABLESPACE табличное_пространство]
       [,PARTITION [имя_раздела] [TABLESPACE табличное_пространство] ...])
         }
       | LOCAL STORE IN ({табличное_пространство[,табличное_пространство ...] | DEFAULT})
         (PARTITION [имя_раздела]
          [Инструкция_атрибутов_индекса]
          { STORE IN { табличное_пространство[,табличное_пространство ...] | DEFAULT}
          | (SUBPARTITION имя_подраздела [TABLESPACE табличное_пространство]
            [,SUBPARTITION имя_подраздела [TABLESPACE табличное_пространство] ...])
```

```
            }
          }
        | Инструкция_атрибутов_индекса
        }
      | INDEXTYPE IS индексный_тип
        [PARALLEL | NOPARALLEL]
        [PARAMETERS('строка_параметров')]
      }
  | [схема.] имя_таблицы ([схема.]имя_таблицы столбец [{ASC | DESC}]
    [,[схема.]имя_таблицы столбец [{ASC | DESC}]...])
    FROM [схема.] имя_таблицы [,[схема.] имя_таблицы...]
    WHERE условие
    { LOCAL (PARTITION имя_раздела
      [Инструкция_физических_атрибутов]
      [TABLESPACE табличное_пространство]
      [LOGGING | NOLOGGING]
      [,PARTITION имя_раздела
       [Инструкция_физических_атрибутов]
       [TABLESPACE табличное_пространство]
       [LOGGING | NOLOGGING] ...])
    | LOCAL
      {STORE IN ({табличное_пространство[,табличное_пространство...]|DEFAULT})
      | (PARTITION [имя_раздела] [TABLESPACE табличное_пространство]
        [,PARTITION [имя_раздела] [TABLESPACE табличное_пространство] ...])
      }
    | LOCAL STORE IN ({табличное_пространство[,табличное_пространство ...] | DEFAULT})
      (PARTITION [имя_раздела]
       [Инструкция_атрибутов_индекса]
       { STORE IN { табличное_пространство[,табличное_пространство ...] | DEFAULT}
       | (SUBPARTITION имя_подраздела [TABLESPACE табличное_пространство]
         [,SUBPARTITION имя_подраздела [TABLESPACE табличное_пространство] ...])
       }
    }
  }
```

Синтаксис ALTER:

```
ALTER INDEX [схема.]имя_индекса
[DEALLOCATE UNUSED [KEEP целое [K | M]]]
[ALLOCATE EXTENT (
  [SIZE целое [K | M]]
  [DATAFILE 'имя_файла']
  [INSTANCE целое])]
[PARALLEL целое | NOPARALLEL]
[Инструкция_физических_атрибутов]
[Инструкция_хранения]
[LOGGING | NOLOGGING]
[REBUILD
  [{ {PARTITION имя_раздела | SUBPARTITION имя_подраздела}
    | REVERSE | NOREVERSE}
   }]
   [PARALLEL целое | NOPARALLEL]
   [TABLESPACE табличное_пространство]
   [ONLINE]
   [Инструкция_физических_атрибутов]
   [Инструкция_хранения]
   [{COMPRESS целое | NOCOMPRESS}]
   [{LOGGING | NOLOGGING}]
```

```
    [COMPUTE STATISTICS]
    [PARAMETERS ('параметры_перестраивания')]
[PARAMETERS ('параметры_изменения')]
[{ENABLE | DISABLE}]
[UNUSABLE]
[RENAME [PARTITION имя_раздела] TO новое_имя]
[COALESCE]
[{MONITORING | NOMONITORING} USAGE]
[Инструкция_раздела]
```

Синтаксис DROP:

```
DROP INDEX [схема.]имя_индекса
```

Создает, изменяет или удаляет индекс для одного или нескольких столбцов таблицы или для кластера.

Если *Инструкция_хранения* в команде CREATE INDEX или ALTER INDEX не указана, то сервер Oracle выделяет место для хранения индекса следующим образом:

- Если индексированная таблица не содержит строк, то сервер Oracle берет параметры хранения, заданные по умолчанию для табличного пространства.
- Если индексированная таблица содержит строки и результирующий индекс занимает не более 25 блоков данных, то сервер Oracle выделяет для такого индекса один экстент.
- Если индексированная таблица содержит строки и результирующий индекс занимает более 25 блоков данных, то сервер Oracle выделяет для такого индекса пять экстентов одинакового размера.

Ключевые слова

UNIQUE

Указывает, что значение столбца(ов), по которому(ым) строится индекс, должно быть уникальным.

BITMAP

Указывает, что должен быть создан битовый индекс, а не обычный индекс в виде двоичного дерева.

ON CLUSTER

Указывает, что индекс строится для кластера (это не может быть кластер с хеш-ключом), и сообщает имя кластера.

ON имя_таблицы

Указывает, что индекс строится для таблицы, и приводит имя таблицы.

INDEXTYPE IS индексный_тип

Указывает, что создается доменный индекс. Значение *индексный_тип* – это имя объекта индексного типа, который должен уже существовать. Появилось в Oracle8*i*.

PARAMETERS

Задает строку параметров, которая передается в функцию, реализующую *индексный_тип*.

псевдоним

Псевдоним для таблицы, по которой строится индекс. Необходим, если индекс ссылается на какие-либо атрибуты или методы объектного типа.

ASC

Указывает, что индекс должен строиться в порядке возрастания значений кодов символов из набора символов БД.

DESC

Указывает, что индекс должен строиться в порядке убывания значений кодов символов из набора символов БД.

ONLINE

Указывает, что операции DML для индексируемой таблицы могут выполняться в процессе создания индекса. Появилось в Oracle 8*i*.

COMPUTE STATISTICS

Указывает, что в процессе создания индекса необходимо вычислять и записывать статистические данные в словарь данных. Появилось в Oracle8*i*.

GLOBAL PARTITION BY RANGE

Означает, что глобальный индекс должен быть секционирован по диапазону значений указанных столбцов.

список_столбцов

Имя (имена) столбца (столбцов), по которым индекс секционируется. Начиная с версии Oracle8*i* этот список может содержать функции или столбцы.

LOCAL

Указывает, что индекс должен быть секционирован по тем же самым столбцам, что и таблица, для которой он построен, при этом количество разделов и их границы также должны совпадать.

PARTITION

Определяет имена отдельных разделов, количество которых должно совпадать с количеством разделов таблицы и которые должны быть приведены в том же порядке.

LOCAL STORE IN

Определяет, каким образом хеш-разделы или подразделы индекса будут распределены по табличным пространствам.

DEFAULT

Указывает, что для локального индекса, построенного для таблицы с комбинированным секционированием или секционированием хеш-ключом, табличное пространство, определенное на уровне индекса, будет подменено, а использоваться будет тот же раздел или подраздел, в котором расположена таблица.

SUBPARTITION

Задает имя подраздела.

Дополнительные ключевые слова ALTER

DEALLOCATE UNUSED

Указывает, что неиспользуемое пространство в конце индекса будет освобождено и станет доступным для других нужд базы данных.

KEEP

Указывает количество байт сверх *маркера максимального заполнения* (*highwater mark*), которые останутся в составе индекса после освобождения неиспользуемого пространства.

ALLOCATE EXTENT

Означает, что для данного индекса должен быть выделен новый экстент.

SIZE

Задает размер выделяемого индекса.

DATAFILE

Определяет имя файла данных, который будет содержать новый экстент. Если не указано, то сервер Oracle выберет файл данных из доступных для данного индекса.

INSTANCE

Указывает, что новый экстент будет доступен только данному экземпляру. Если ключевое слово не указано, то экстент будет доступен всем экземплярам.

REBUILD PARTITION

Указывает раздел индекса, который должен быть перестроен.

REBUILD SUBPARTITION

Указывает подраздел индекса, который должен быть перестроен.

REVERSE

Указывает, что байты блока индекса будут храниться в обратном порядке (за исключением ROWID) при перестраивании индекса.

NOREVERSE

Указывает, что байты блока индекса будут храниться без изменения порядка на обратный при перестраивании индекса.

параметры_перестраивания

Строка параметров, передаваемая в процедуру *индексный_тип* для перестраивания глобального индекса.

параметры_изменения

Строка параметров, передаваемая в процедуру *индексный_тип* при изменении глобального индекса.

ENABLE

Активирует отключенный *индекс по функции* (*function-based index*).

DISABLE

Отключает индекс по функции.

UNUSABLE

Указывает, что индекс помечается как неиспользуемый.

RENAME TO

Указывает, что индекс переименовывается.

COALESCE

Указывает, что содержимое блоков индекса объединяется, чтобы освободить блоки для повторного использования. Появилось в Oracle8*i*.

MONITOR

Указывает, что начинается мониторинг индекса. Появилось в Oracle9*i*.

NOMONITOR

Указывает, что завершается мониторинг индекса. Появилось в Oracle9*i*.

Общие ключевые слова и инструкции: *столбец*, *имя_файла*, *Инструкция_атрибутов_индекса*, *имя_индекса*, *целое*, LOGGING, NOLOGGING, NOPARALLEL, OR REPLACE, PARALLEL, *Инструкция_раздела*, *имя_раздела*, *Инструкция_физических_параметров*, *схема*, *Инструкция_хранения*, *имя_подраздела*, *имя_таблицы*, *табличное_пространство*.

CREATE/DROP INDEXTYPE

Синатксис CREATE:

```
CREATE INDEXTYPE [схема.]индексный_тип
FOR [схема.]оператор (тип_параметра[,тип_параметра...])[,[схема.]оператор...]
USING [схема.]тип_реализации
```

Синтаксис DROP

```
DROP INDEXTYPE [схема.]индексный_тип
```

Создает или удаляет индексный тип – объект, используемый для управления *прикладным индексом* (*domain index*). Появилось в Oracle8*i*.

Ключевые слова

индексный_тип

Имя объекта индексного типа.

FOR

Указывает список операторов, поддерживаемых данным *индексным_типом*.

оператор

Имя оператора.

тип_параметра

Указывает тип параметра для оператора.

USING

Указывает тип, обеспечивающий реализацию для *индексного_типа*.

тип_реализации

Имя типа, реализующего интерфейс Oracle Data Cartridge (ODCI).

Общие ключевые слова и инструкции: *схема*.

CREATE/ALTER/DROP JAVA

Синтаксис CREATE:

```
CREATE [OR REPLACE]  [AND {RESOLVE | COMPILE}] [NOFORCE]
JAVA
{ {SOURCE | RESOURCE} NAMED [схема.]имя_объекта
| CLASS [SCHEMA схема]
}
[AUTHID {CURRENT_USER | DEFINER}]
RESOLVER ((строка_совпадения[,]{имя_схемы | -})[(строка_совпадения...)])]
{ USING
  {BFILE (имя_каталога,имя_серверного_файла)
  | { BLOB подзапрос
    | CLOB
    | BFILE
    }
```

```
    | 'ключ_для_BLOB'
    }
  |
    AS исходный_текст
  }
```

Синтаксис ALTER:

```
ALTER JAVA
{SOURCE | CLASS} [схема.]имя_объекта
[RESOLVER ((строка_совпадения[,]{имя_схемы | -})[(строка_совпадения...)])]
{COMPILE | RESOLVE}
[AUTHID {CURRENT_USER | DEFINER}]
```

Синтаксис DROP:

```
DROP JAVA {SOURCE | CLASS | RESOURCE} [схема.]имя_объекта
```

Создает объект, содержащий исходный текст, класс или ресурс Java; или инициирует разрешение (resolution) Java-класса, являющегося объектом схемы, или компиляцию исходного объекта схемы Java; или удаляет объект Java из базы данных. Появилось в Oracle8*i*.

Ключевые слова

OR REPLACE

Означает, что объект Java должен быть заменен, если он уже существовал.

RESOLVE | COMPILE

Указывает, что сервер Oracle должен попытаться разрешить имя созданного объекта схемы Java. Ключевые слова имеют одинаковое значение и являются взаимозаменяемыми.

NOFORCE

Указывает, что для результатов этой команды будет выполнен откат, если указано ключевое слово RESOLVE или COMPILE и разрешить имя не удается.

SOURCE

Указывает, что будет загружен исходный файл Java.

RESOURCE

Указывает, что будет загружен файл ресурса Java.

имя_объекта

Имя объекта схемы, хранящего исходный текст или ресурс Java.

CLASS

Указывает, что будет загружен файл класса Java.

SCHEMA

Определяет схему, в которой хранится объект, содержащий файл Java. Если ключевое слово пропущено, то задействуется схема пользователя.

AUTHID CURRENT_USER

Указывает, что методы выполняются с привилегиями текущего пользователя.

DEFINER

Указывает, что методы выполняются с привилегиями создателя объектов схемы Java.

RESOLVER

Указывает, что объект схемы Java должен быть сопоставлен уточненному имени Java.

строка_совпадения

Полное уточненное имя Java или маска, которая соответствует одному или нескольким именам Java.

имя_схемы

Определяет, в какой схеме будет вестись поиск соответствия для объекта Java.

– (тире)

Означает, что если *строка_совпадения* соответствует корректному имени Java, то схема может остаться неразрешенной.

USING

Задает последовательность данных для класса или ресурса Java.

BFILE

Определяет каталог и имя файла операционной системы, содержащего последовательность Java.

BLOB

Определяет подзапрос, который возвращает одну строку типа BLOB.

CLOB

Определяет подзапрос, который возвращает одну строку типа CLOB.

BFILE

Определяет подзапрос, который возвращает одну строку типа BFILE.

'*ключ_для_BLOB*'

Указывает, что будет использован явный запрос к таблице CREATEJVATABLE текущей схемы.

AS исходный_текст

Определяет последовательность символов для Java или SQLJ.

Общие ключевые слова и инструкции: *схема.*

CREATE/DROP LIBRARY

Синтаксис CREATE :

```
CREATE [OR REPLACE]LIBRARY [схема.]имя_библиотеки
{IS | AS} 'имя_файла'
```

Синтаксис DROP:

```
DROP LIBRARY [схема.]имя_библиотеки
```

Создает или удаляет объект схемы, сопоставленный разделяемой библиотеке операционной системы.

Ключевые слова

OR REPLACE

Указывает, что представляющий библиотеку объект схемы должен быть заменен, если он уже присутствует в схеме.

имя_библиотеки

Имя библиотечного объекта схемы.

Общие ключевые слова и инструкции: *имя_файла, схема.*

CREATE/ALTER/DROP MATERIALIZED VIEW

Синтаксис CREATE:

```
CREATE MATERIALIZED VIEW [схема.]имя_материализованного_представления
[Инструкция_физических_атрибутов]
[TABLESPACE табличное_пространство]
[Инструкция_хранения]
[REFRESH [FAST | COMPLETE | FORCE]
  [START WITH дата] [NEXT дата]]
AS запрос_материализованного_представления
```

Синтаксис ALTER:

```
ALTER MATERIALIZED VIEW [схема.]имя_материализованного_представления
[Инструкция_физических_атрибутов]
[Инструкция_хранения]
[REFRESH [FAST | COMPLETE | FORCE][START WITH дата][NEXT дата]
```

Синтаксис DROP:

```
DROP MATERIALIZED VIEW [схема.]имя_материализованного_представления
```

Создает, изменяет или удаляет материализованное представление (называемое также моментальной копией), представляющее собой результат запроса к одной или нескольким таблицам или представлениям. Прежде чем вы попытаетесь создать материализованное представление, необходимо запустить сценарий *dbmssnap.sql* от имени SYS. Появилось в Oracle8*i*.

Ключевые слова

имя_материализованного_представления

Имя материализованного представления. При создании объектов материализованных представлений в схеме сервер Oracle добавляет в конец имени 7-символьный идентификатор, поэтому *имя_материализованного_представления* должно содержать не более 27 символов.

TABLESPACE

Указывает имя табличного пространства, в котором будет храниться материализованное представление. Если не указано иное, используется табличное пространство по умолчанию для владельца схемы.

REFRESH

Задает режим и время автоматических обновлений.

FAST

Использовать журнал материализованного представления, соответствующий главной таблице.

COMPLETE

Обновление за счет повторного выполнения запроса материализованного представления.

FORCE

Сервер Oracle определяет, возможно ли обновление FAST; если нет, то выполняется обновление COMPLETE. По умолчанию устанавливается режим FORCE.

START WITH

Определяет дату следующего автоматического обновления при помощи стандартного выражения для дат Oracle.

NEXT

Указывает выражение, возвращающее дату для вычисления интервала между автоматическими обновлениями.

AS запрос_материализованного_представления

Содержит фактический запрос SQL, используемый для наполнения материализованного представления, к которому применяются те же ограничения, что и к представлению.

Общие ключевые слова и инструкции: *дата*, *Инструкция_физических_атрибутов*, *схема*, *Инструкция_хранения*.

CREATE/ALTER/DROP MATERIALIZED VIEW LOG

Синтаксис CREATE:

```
CREATE MATERIALIZED VIEW LOG ON [схема.]имя_таблицы
[Инструкция_физических_атрибутов]
[TABLESPACE табличное_пространство]
[Инструкция_хранения]
```

Синтаксис ALTER:

```
ALTER MATERIALIZED VIEW LOG ON [схема.]имя_таблицы
[Инструкция_физических_атрибутов]
[Инструкция_хранения]
```

Синтаксис DROP:

```
DROP MATERIALIZED VIEW LOG ON [схема.]имя_таблицы
```

Создает, изменяет или удаляет журнал материализованного представления, который представляет собой таблицу, связанную с основной таблицей материализованного представления и используемую для управления обновлениями материализованного представления. Появилось в Oracle8*i*.

Ключевые слова

имя_таблицы

Определяет имя таблицы, для которой будет вестись журнал материализованного представления.

Общие ключевые слова и инструкции: *Инструкция_физических_атрибутов*, *схема*, *Инструкция_хранения*, *имя_таблицы*, *табличное_пространство*.

CREATE/ALTER/DROP OPERATOR

Синтаксис CREATE:

```
CREATE OPERATOR [схема.] оператор
BINDING (тип_параметра [, тип_параметра]... )
RETURN возвращаемый_тип
  {[ANCILLARY TO
    первичный_оператор (тип_параметра [,тип_параметра]... )
   [, первичный_оператор (тип_параметра [,тип_параметра]... )]...]
  | WITH INDEX CONTEXT, SCAN CONTEXT тип_реализации
```

```
    [COMPUTE ANCILLARY DATA]
  }
  USING [схема .] [ пакет . | тип . ] имя_функции]...
```

Синтаксис ALTER:

```
ALTER OPERATOR [схема.] оператор  COMPILE
```

Синтаксис DROP:

```
DROP OPERATOR оператор [FORCE]
```

Создает, изменяет или удаляет новый оператор и определяет его переменные связывания. Появилось в Oracle8*i*. Синтаксис ALTER приведен для Oracle9*i* Release 2.

Ключевые слова

оператор

Имя, которое будет присвоено данному оператору.

BINDING

Задает один или несколько типов параметров для связывания оператора с функцией.

RETURN

Определяет возвращаемый тип данных для связывания.

ANCILLARY TO

Указывает, что связывание оператора является дополнительным по отношению к указанному связыванию первичного оператора.

COMPUTE ANCILLARY DATA

Указывает, что связывание оператора вычисляет дополнительные данные.

имя_функции

Функция, которая содержит реализацию связывания. Функция, определяемая параметром *имя_функции*, может быть автономной функцией, функцией пакета, методом или синонимом для любой из упомянутых функций.

ALTER OPERATOR COMPILE

Указывает, что существующий оператор должен быть перекомпилирован. Появилось в Oracle9*i* Release 2.

FORCE

Указывает, что оператор должен быть удален, даже если на него в текущий момент ссылается один или несколько объектов схемы (индексные типы, пакеты, функции, процедуры и т. д.), и помечает эти зависимые объекты как INVALID. Если ключевое слово FORCE не указано, то пользователь не может удалить оператор, на который ссылается какой-то объект схемы.

Общие ключевые слова и инструкции: *схема*.

CREATE/ALTER/DROP OUTLINE

Синтаксис CREATE:

```
CREATE [{PUBLIC | PRIVATE}] OUTLINE [хранимый_план]
[FROM [{PUBLIC | PRIVATE}] исходный_хранимый_план]
[FOR CATEGORY категория] [ON команда]
```

Синтаксис ALTER:

```
ALTER OUTLINE [{PUBLIC | PRIVATE}] хранимый_план
{ REBUILD
```

```
| RENAME TO новый_хранимый_план
| CHANGE CATEGORY TO новое_имя_категории
}
```

Синтаксис DROP:

```
DROP OUTLINE хранимый_план
```

Создает, изменяет или удаляет набор атрибутов, используемых оптимизатором для формирования хранимого плана выполнения запроса. Появилась в Oracle8*i*.

Ключевые слова

PUBLIC

Означает, что *хранимый_план* доступен всем пользователям (по умолчанию). Возможность применения данного ключевого слова в команде ALTER OUTLINE появилась в Oracle9*i*.

PRIVATE

Указывает, что *хранимый_план* доступен только для текущего сеанса. Возможность применения данного ключевого слова в команде ALTER OUTLINE появилась в Oracle9*i*.

исходный_хранимый_план

Имя существующего *хранимого_плана*, с которого будет скопирован текущий.

категория

Имя категории для группировки *хранимых_планов*.

ON инструкция

Определяет инструкцию SQL, для которой сервер Oracle будет создавать *хранимый_план* при компиляции команды. Эта инструкция является необязательной, только если вы создаете копию существующего *хранимого_плана* при помощи инструкции FROM.

REBUILD

Указывает, что сервер Oracle повторно сформирует план выполнения запроса для *хранимого_плана*, используя текущие условия.

RENAME TO

Указывает, что *хранимый_план* должен быть переименован.

CHANGE CATEGORY TO

Указывает, что *хранимый_план* будет помещен в новую категорию.

CREATE/ALTER/DROP PACKAGE

Создает, изменяет или удаляет хранимый пакет функций, процедур и других программных объектов PL/SQL в БД (синтаксис и информация о применении приведены в главе 9).

CREATE/ALTER/DROP PACKAGE BODY

Создает, изменяет или удаляет тело хранимого пакета, определяющее набор процедур, функций и других программных объектов PL/SQL в БД (синтаксис и информация о применении приведены в главе 9).

CREATE/ALTER/DROP PROCEDURE

Создает, изменяет или удаляет процедуру PL/SQL в базе данных (синтаксис и информация о применении приведены в главе 9).

CREATE PFILE

```
CREATE PFILE [= 'имя_файла_параметров'] FROM SPFILE [= 'имя_файла_параметров_сервера']
```

Создает текстовый файл, содержащий параметры инициализации. Появилась в Oracle9*i* Release 2.

Ключевые слова

имя_файла_параметров

Указывает имя выходного текстового файла, который будет создан и будет содержать информацию о параметрах в текстовом виде. Если *имя_файла_параметров* не указано, сервер Oracle берет имя файла, определяемое платформой, обычно *INITsid.ORA*.

имя_файла_параметров_сервера

Указывает имя существующего двоичного файла параметров сервера, из которого будут взяты параметры. Если *имя_файла_параметров_сервера* не указано, сервер Oracle берет имя файла, определяемое платформой.

CREATE/ALTER/DROP PROFILE

Синтаксис CREATE:

```
CREATE PROFILE имя_профиля LIMIT
[SESSIONS_PER_USER  целое | UNLIMITED | DEFAULT]
[CPU_PER_SESSION  целое | UNLIMITED | DEFAULT]
[CPU_PER_CALL  целое | UNLIMITED | DEFAULT]
[CONNECT_TIME  целое | UNLIMITED | DEFAULT]
[IDLE_TIME  целое | UNLIMITED | DEFAULT]
[LOGICAL_READS_PER_SESSION  целое | UNLIMITED | DEFAULT]
[LOGICAL_READS_PER_CALL  целое | UNLIMITED | DEFAULT]
[PRIVATE_SGA  {целое [K | M] | UNLIMITED | DEFAULT}]
[COMPOSITE_LIMIT  {целое | UNLIMITED | DEFAULT}]
[FAILED_LOGIN_ATTEMPTS выражение | UNLIMITED | DEFAULT]
[PASSWORD_LIFE_TIME выражение | UNLIMITED | DEFAULT]
[PASSWORD_REUSE_TIME выражение| UNLIMITED | DEFAULT]
[PASSWORD_REUSE_MAX выражение| UNLIMITED | DEFAULT]
[PASSWORD_LOCK_TIME выражение | UNLIMITED | DEFAULT]
[PASSWORD_GRACE_TIME выражение | UNLIMITED | DEFAULT]
[PASSWORD_VERIFY_FUNCTION функция | NULL | DEFAULT]
```

Синтаксис ALTER:

```
ALTER PROFILE имя_профиля LIMIT
[SESSIONS_PER_USER  целое | UNLIMITED | DEFAULT]
[CPU_PER_SESSION  целое | UNLIMITED | DEFAULT]
[CPU_PER_CALL  целое | UNLIMITED | DEFAULT]
[CONNECT_TIME  целое | UNLIMITED | DEFAULT]
[IDLE_TIME  целое | UNLIMITED | DEFAULT]
[LOGICAL_READS_PER_SESSION  целое | UNLIMITED | DEFAULT]
[LOGICAL_READS_PER_CALL  целое | UNLIMITED | DEFAULT]
```

```
[PRIVATE_SGA  целое [K | M] | UNLIMITED | DEFAULT]
[COMPOSITE_LIMIT  целое | UNLIMITED | DEFAULT]
[FAILED_LOGIN_ATTEMPTS выражение | UNLIMITED | DEFAULT]
[PASSWORD_LIFE_TIME выражение | UNLIMITED | DEFAULT]
[PASSWORD_REUSE_TIME выражение | UNLIMITED | DEFAULT]
[PASSWORD_REUSE_MAX выражение | UNLIMITED | DEFAULT]
[PASSWORD_LOCK_TIME выражение | UNLIMITED | DEFAULT]
[PASSWORD_GRACE_TIME выражение | UNLIMITED | DEFAULT]
[PASSWORD_VERIFY_FUNCTION функция | NULL | DEFAULT]
```

Синтаксис DROP:

```
DROP PROFILE имя_профиля [CASCADE]
```

Создает, изменяет или удаляет профиль, налагающий ограничения на ресурсы БД.

Для того чтобы применить ограничения, сопоставленные профилю, к конкретному пользователю, необходимо назначить ему профиль при помощи команды CREATE USER или ALTER USER. Ограничения на ресурсы также могут быть введены посредством параметра инициализации RESOURCE_LIMIT или при помощи команды ALTER SYSTEM.

Ключевые слова

имя_профиля

Имя создаваемого профиля.

SESSIONS_PER_USER

Ограничивает количество параллельных сеансов для пользователя.

CPU_PER_SESSION

Ограничивает количество процессорного времени, которое может быть израсходовано сеансом (в сотых долях секунды).

CPU_PER_CALL

Ограничивает количество процессорного времени для вызова (синтаксического анализа, исполнения или выборки) (в сотых долях секунды).

CONNECT_TIME

Ограничивает общее время продолжительности сеанса (в минутах).

IDLE_TIME

Ограничивает период пассивного ожидания сеанса (в минутах).

LOGICAL_READS_PER_SESSION

Ограничивает количество блоков БД, считываемых в рамках сеанса, включая читаемые с диска и из оперативной памяти.

LOGICAL_READS_PER_CALL

Ограничивает количество блоков БД, считываемых вызовом (синтаксическим анализом, исполнением или выборкой).

PRIVATE_SGA

Ограничивает объем памяти, который сеанс может выделить в разделяемом пуле SGA (в байтах).

COMPOSITE_LIMIT

Ограничивает общую стоимость ресурсов для сеанса в сервисных единицах стоимости ресурсов. Дополнительную информацию можно найти в описании ALTER RESOURCE COST ранее в этом же разделе.

UNLIMITED

Если указано данное значение, это означает, что на данный ресурс не будет наложено ограничений.

DEFAULT

Если указано данное значение, то для данного ресурса будет действовать ограничение, указанное в профиле DEFAULT.

FAILED_LOGIN_ATTEMPTS

Определяет количество неудачных попыток входа в систему, после которого учетная запись блокируется.

PASSWORD_LIFE_TIME

Определяет срок действия пароля – количество дней, по истечении которого пароль теряет силу и должен быть изменен.

PASSWORD_REUSE_TIME

Указывает количество дней, через которое возможно повторное применение пароля, использовавшегося ранее. Если параметру присвоено значение, то PASSWORD_REUSE_MAX должен быть установлен в UNLIMITED.

PASSWORD_REUSE_MAX

Указывает количество изменений пароля, необходимых для того, чтобы можно было повторно использовать текущий пароль. Если параметру присвоено значение, то PASSWORD_REUSE_TIME должен быть установлен в UNLIMITED.

PASSWORD_LOCK_TIME

Определяет количество дней, в течение которых учетная запись будет оставаться заблокированной после превышения количества FAILED_LOGIN_ATTEMPTS.

PASSWORD_GRACE_TIME

Задает количество дней после истечения срока действия пароля, в течение которых вход в систему с выдачей соответствующего предупреждения будет разрешен.

PASSWORD_VERIFY_FUNCTION

Указывает имя функции PL/SQL, применяемой для проверки паролей. Установка этого параметра в NULL означает, что проверка производиться не будет.

Общие ключевые слова и инструкции: *выражение*.

CREATE/ALTER/DROP ROLE

Синтаксис CREATE :

```
CREATE ROLE имя_роли
{NOT IDENTIFIED
| IDENTIFIED
  { EXTERNALLY
  | GLOBALLY
  | BY пароль
  | USING [схема.]пакет
  }
}
```

Синтаксис ALTER :

```
ALTER ROLE имя_роли
{NOT IDENTIFIED
| IDENTIFIED
  { EXTERNALLY
  | GLOBALLY
  | BY пароль
  | USING [схема.]пакет
  }
}
```

Синтаксис DROP:

```
DROP ROLE имя_роли
```

Создает, изменяет или удаляет роль, представляющую собой набор привилегий, которые могут быть выданы пользователям. Создающему роль пользователю она автоматически выдается с параметром WITH ADMIN OPTION, что позволяет ему выдавать и отзывать эту роль, а также изменять ее при помощи команды ALTER ROLE.

Ключевые слова

имя_роли

Имя создаваемой роли.

NOT IDENTIFIED

Указывает, что пользователь, которому была выдана роль, не должен проходить проверку при ее включении.

IDENTIFIED BY пароль

Указывает, что для включения роли необходимо предоставить пароль.

IDENTIFIED EXTERNALLY

Означает, что операционная система проверяет пользователя, который включает роль.

IDENTIFIED GLOBALLY

Указывает, что создается глобальный пользователь. Глобальный пользователь должен получить разрешение использовать роль от службы каталогов предприятия, прежде чем роль будет включена командой SET ROLE или же при входе в систему.

IDENTIFIED USING

Указывает, что это роль приложения, т. е. роль, которая может быть включена только приложениями, использующими пакеты. Появилось в Oracle9*i*.

Общие ключевые слова и инструкции: *схема*.

CREATE/ALTER/DROP ROLLBACK SEGMENT

Синтаксис CREATE:

```
CREATE [PUBLIC] ROLLBACK SEGMENT имя_сегмента
TABLESPACE табличное_пространство
 [Инструкция_хранения]
```

Синтаксис ALTER :

```
ALTER ROLLBACK SEGMENT имя_сегмента
[Инструкция_хранения]
```

```
[ONLINE | OFFLINE]
[SHRINK]
```

Синтаксис DROP :

```
DROP ROLLBACK SEGMENT имя_сегмента
```

Создает, изменяет или удаляет сегмент отката, который сервер Oracle использует для хранения данных, необходимых для отката выполненных транзакциями изменений.

При создании сегмента отката он находится в автономном режиме. Для того чтобы перевести его в оперативный режим, следует выполнить команду ROLLBACK SEGMENT или перезапустить базы данных с сегментом отката, имя которого указано в файле инициализации.

Ключевые слова

PUBLIC	Указывает, что данный сегмент отката доступен всем экземплярам. Если ключевое слово пропущено, то сегмент отката будет доступен только тому экземпляру, который укажет его имя в параметре ROLLBACK_SEGMENTS файла инициализации.
имя_сегмента	Имя создаваемого сегмента отката.
TABLESPACE	Указывает имя табличного пространства, в котором будет храниться сегмент отката.
ONLINE	Указывает, что данный сегмент отката должен быть переведен в оперативный режим.
OFFLINE	Указывает, что данный сегмент отката должен быть переведен в автономный режим.
SHRINK	Означает, что размер данного сегмента отката должен быть уменьшен до заданного размера (или до размера OPTIMAL, если значение не указано).

Общие ключевые слова и инструкции: *Инструкция_хранения*, *табличное_пространство*.

CREATE SCHEMA

```
CREATE SCHEMA AUTHORIZATION схема
[CREATE TABLE команда]
[CREATE VIEW команда]
[GRANT команда]
```

Создает несколько таблиц и/или представлений и выдает права на них в одной инструкции.

Для выдачи данной команды пользователь должен обладать такими же привилегиями, как для выдачи команд CREATE TABLE, CREATE VIEW и GRANT. Отдельные команды внутри команды CREATE SCHEMA не должны оканчиваться завершающим символом SQL.

Ключевые слова

схема

Имя создаваемой схемы. Оно должно совпадать с вашим именем пользователя.

CREATE TABLE команда

Это инструкция CREATE TABLE, описанная в одноименном разделе.

CREATE VIEW команда

Это инструкция CREATE VIEW, описанная в одноименном разделе.

GRANT команда

Это инструкция GRANT, описанная в одноименном разделе.

CREATE/ALTER/DROP SEQUENCE

Синтаксис CREATE :

```
CREATE SEQUENCE [схема.]имя_последовательности
[INCREMENT BY целое]
[START WITH целое]
[MAXVALUE целое | NOMAXVALUE]
[MINVALUE целое | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE целое | NOCACHE]
[ORDER | NOORDER]
```

Синтаксис ALTER:

```
ALTER SEQUENCE [схема.]имя_последовательности
[INCREMENT BY целое]
[MAXVALUE целое | NOMAXVALUE]
[MINVALUE целое | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE целое | NOCACHE]
[ORDER | NOORDER]
```

Синтаксис DROP:

```
DROP SEQUENCE [схема.] имя_последовательности
```

Создает, изменяет или удаляет последовательность Oracle, которая может применяться для автоматического формирования последовательных номеров в процессе выполнения операций над БД. На формирование номера последовательности не влияет последующий откат транзакции. После того как номер создан, он не может стать доступным вновь, поэтому в случае откатов возможно возникновение разрывов в нумерации. Для обращения к последовательностям используют псевдостолбцы CURRVAL и NEXTVAL.

Для того чтобы заново начать последовательность с меньшего номера, следует последовательно выдать команды DROP SEQUENCE и CREATE SEQUENCE. Однако все разрешения, выданные на работу с последовательностью (GRANT), также должны быть пересозданы.

Ключевые слова

INCREMENT BY

Задает приращение для номеров последовательности; может быть положительным или отрицательным (но не 0). Значение по умолчанию равно 1.

START WITH

Задает первый номер последовательности. Значение по умолчанию равно MINVALUE для возрастающих последовательностей и MAXVALUE – для убывающих.

MAXVALUE

Определяет максимальное значение для последовательности. По умолчанию равно NOMAXVALUE, т. е. максимальным номером будет 10^{27}.

MINVALUE

Определяет минимальное значение для последовательности. По умолчанию равно NOMINVALUE, т. е. минимальным номером будет 1.

CYCLE

Означает, что после того как последовательность достигает MAXVALUE, она начинается заново со значения MINVALUE. По умолчанию равно NOCYCLE.

NOCYCLE

Указывает, что после достижения максимального значения номера последовательности больше не генерируются.

CACHE

Определяет, сколько номеров последовательности сервер Oracle будет формировать заранее и хранить в оперативной памяти. Имейте в виду, что при отключении БД неиспользованные номера последовательности, хранящиеся в кэше, будут утеряны. Значение по умолчанию равно 20.

NOCACHE

Указывает, что номера последовательности не будут формироваться заранее.

ORDER

Означает, что номера последовательности гарантированно выдаются в запрошенном порядке. Значение по умолчанию равно NOORDER.

NOORDER

Означает, что выдача номеров последовательности в запрошенном порядке не гарантируется.

Общие ключевые слова и инструкции: *целое*, *схема*.

CREATE/ALTER/DROP SNAPSHOT

Синтаксис CREATE:

```
CREATE SNAPSHOT [схема.]имя_моментальной_копии
[Инструкция_физических_атрибутов]
[TABLESPACE табличное_пространство]
[Инструкция_хранения]
[REFRESH [FAST | COMPLETE | FORCE]
[START WITH дата][NEXT дата]]
AS запрос_моментальной_копии
```

Синтаксис ALTER:

```
ALTER SNAPSHOT [схема.]имя_моментальной_копии
[Инструкция_физических_атрибутов] [Инструкция_хранения]
[REFRESH [FAST | COMPLETE | FORCE]
 [START WITH дата][NEXT дата]]
```

Синтаксис DROP:

```
DROP SNAPSHOT [схема.]имя_моментальной_копии
```

Создает, изменяет или удаляет моментальную копию, представляющую собой результат запроса, выполненного для одной или нескольких таблиц или представлений. Обратите внимание, что прежде чем пытаться создать моментальную копию, необходимо запустить сценарий *dbmssnap.sql* от имени SYS для создания встроенного пакета DBMS_SNAPSHOT.

Ключевые слова

имя_моментальной_копии

Имя моментальной копии. При создании объектов моментальных копий в схеме сервер Oracle добавляет в конец имени идентификатор длиной 7 символов, поэтому *имя_моментальной_копии* должно содержать не более 23 символов.

TABLESPACE

Указывает имя табличного пространства, в котором будет храниться моментальная копия. Если ключевое слово пропущено, то будет использовано табличное пространство по умолчанию для владельца схемы.

REFRESH

Задает режим и время автоматических обновлений.

FAST

Использовать журнал моментальной копии, соответствующий главной таблице.

COMPLETE

Обновление за счет повторного выполнения запроса моментальной копии.

FORCE

Сервер Oracle определяет, возможно ли обновление FAST, и если нет, то выполняется обновление COMPLETE. FORCE – это режим по умолчанию.

START WITH

Определяет дату следующего автоматического обновления при помощи стандартного выражения для дат Oracle.

NEXT

Указывает новое выражение, возвращающее дату для вычисления интервала между автоматическими обновлениями.

AS

Приводит фактический запрос SQL, используемый для наполнения моментальной копии, к которому применяются те же ограничения, что и к представлению.

Общие ключевые слова и инструкции: *дата*, *Инструкция_физических_атрибутов*, *схема*, *Инструкция_хранения*, *табличное_пространство*.

CREATE/ALTER/DROP SNAPSHOT LOG

Синтаксис CREATE:

```
CREATE SNAPSHOT LOG ON [схема.]имя_таблицы
[Инструкция_физических_атрибутов]
[TABLESPACE табличное_пространство]
[Инструкция_хранения]
```

Синтаксис ALTER:

```
ALTER SNAPSHOT LOG ON [схема.]имя_таблицы
[Инструкция_физических_атрибутов]
[Инструкция_хранения]
```

Синтаксис DROP:

```
DROP SNAPSHOT LOG ON [схема.]имя_таблицы
```

Создает, изменяет или удаляет журнал моментальной копии, который представляет собой таблицу, связанную с основной таблицей моментальной копии и используемую для управления обновлениями моментальной копии.

Ключевые слова

TABLESPACE

Указывает имя табличного пространства, в котором будет храниться моментальная копия. Если ключевое слово пропущено, то будет использовано табличное пространство по умолчанию для владельца схемы.

Общие ключевые слова и инструкции: *Инструкция_физических_атрибутов*, *Инструкция_хранения*, *схема*, *имя_таблицы*, *табличное_пространство*.

CREATE SPFILE

```
CREATE SPFILE [= 'имя_файла_параметров'] FROM PFILE [= 'имя_файла_параметров_сервера']
```

Создает двоичный файл параметров сервера путем импорта содержимого текстового файла параметров инициализации (*INIT.ORA* или *SPFILE*). Появилась в Oracle9*i* Release 2.

Ключевые слова

имя_файла_параметров_сервера

Указывает имя создаваемого двоичного файла параметров сервера. Если *имя_файла_параметров_сервера* не указано, сервер Oracle берет имя файла, определяемое платформой, обычно *INITsid.ORA*.

имя_файла_параметров

Указывает имя существующего текстового файла параметров инициализации. Если *имя_файла_параметров* не указано, сервер Oracle берет имя файла, определяемое платформой.

CREATE/DROP SYNONYM

Синтаксис CREATE:

```
CREATE [PUBLIC] SYNONYM имя_синонима
FOR [схема.]имя_объекта[@связь_БД]
```

Синтаксис DROP:

```
DROP [PUBLIC] SYNONYM [схема.]имя_синонима
```

Создает или удаляет общий или частный синоним для объекта базы данных.

Ключевые слова

PUBLIC

Указывает, что данный синоним будет доступен всем пользователям. Если ключевое слово пропущено, то синоним будет доступен только владельцу схемы.

имя_синонима

Имя нового синонима.

имя_объекта

Имя объекта, на который будет ссылаться синоним. Может включать в себя ссылку на удаленную базу данных (применяется конструкция *@связь_БД*).

Oracle сначала выполняет разрешение имен объектов в текущей схеме, поэтому синоним PUBLIC будет использован, только если перед именем объекта не указано имя схемы, после имени объекта не указана связь с удаленной БД и имя не существует в текущей схеме.

Общие ключевые слова и инструкции: *связь_БД*, *схема*.

CREATE/ALTER/DROP TABLE (реляционный синтаксис)

Синтаксис CREATE:

```
CREATE [GLOBAL] [TEMPORARY] TABLE [схема.]имя_таблицы
(столбец  тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца]
  [,столбец  тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца]...]
[Инструкция_ограничения_таблицы])
[ON COMMIT {DELETE | PRESERVE} ROWS]
[Инструкция_физических_атрибутов]
[TABLESPACE табличное_пространство]
[Инструкция_хранения]
[{LOGGING | NOLOGGING}]
[{COMPRESS | NOCOMPRESS}]
{ CLUSTER (столбец[,столбец ...])
| ORGANIZATION HEAP
    [Инструкция_физических_атрибутов]
    [TABLESPACE табличное_пространство]
    [Инструкция_хранения]
    [{LOGGING | NOLOGGING}]
    [{COMPRESS | NOCOMPRESS}]
|  ORGANIZATION INDEX
    [Инструкция_физических_атрибутов]
    [TABLESPACE табличное_пространство]
    [Инструкция_хранения]
    [{LOGGING | NOLOGGING}]
    { {MAPPING TABLE | NOMAPPING}
    | PCTTHRESHOLD целое
    | {COMPRESS целое | NOCOMPRESS}
    }
    [[INCLUDING столбец] OVERFLOW
    [Инструкция_физических_атрибутов]
    [TABLESPACE табличное_пространство]
    [Инструкция_хранения]
    [LOGGING | NOLOGGING]
|  ORGANIZATION EXTERNAL
   ([TYPE драйвер доступа]
     DEFAULT DIRECTORY каталог
     [ACCESS PARAMETERS {(непрозрачная_спецификация) | USING CLOB подзапрос}]
     LOCATION ([каталог:]'указатель_местоположения'
       [,[каталог:]'указатель_местоположения'])
```

```
      [REJECT LIMIT {целое | UNLIMITED}]
  }
  [Инструкция_раздела]
  {
    [ { Инструкция_хранения_varray
      | [LOB
        { (элемент_LOB[,элемент_LOB ...]) STORE AS
          ([TABLESPACE табличное_пространство]
          [{ENABLE | DISABLE} STORAGE IN ROW]
          [Инструкция_хранения]
          [CHUNK целое]
          [PCTVERSION целое]
          [CACHE | NOCACHE [LOGGING | NOLOGGING]])
        | (элемент_LOB) STORE AS [(имя_сегмента_LOB)]
          [([TABLESPACE табличное_пространство]
          [{ENABLE | DISABLE} STORAGE IN ROW]
          [Инструкция_хранения]
          [CHUNK целое]
          [PCTVERSION целое]
          [CACHE | NOCACHE [LOGGING | NOLOGGING]])
        }
      }]
  }
  [ENABLE | DISABLE ROW MOVEMENT]
  [CACHE | NOCACHE]
  [MONITORING | NOMONITORING]
  [ROWDEPENDENCIES | NOROWDEPENDENCIES]
  [PARALLEL целое | NOPARALLEL]
  [ENABLE | DISABLE [VALIDATE | NOVALIDATE]]
    {UNIQUE (столбец[, столбец ...]
    | PRIMARY KEY
    | CONSTRAINT имя_ограничения
    }
    [USING INDEX
      [TABLESPACE табличное_пространство]
      [Инструкция_физических_атрибутов]
      [Инструкция_хранения]
      [NOSORT]
      [LOGGING | NOLOGGING]]
    [EXCEPTIONS INTO [схема.]имя_таблицы]
    [CASCADE]
    [{KEEP | DROP} INDEX]
  [AS подзапрос]
```

Синтаксис ALTER:

```
  ALTER TABLE [схема.]имя_таблицы
  { Инструкция_физических_атрибутов
    [{LOGGING | NOLOGGING}]
    [{COMPRESS | NOCOMPRESS}]
    [ADD SUPPLEMENTAL LOG GROUP журнальная_группа (столбец[,столбец...]) [ALWAYS]
    | DROP SUPPLEMENTAL LOG GROUP журнальная_группа
    }
    [ALLOCATE EXTENT (
    [SIZE целое[K | M]]
    [DATAFILE 'имя_файла']
    [INSTANCE целое]
    )]
```

```
  [DEALLOCATE UNUSED [KEEP целое[K | M]]]
  [CACHE | NOCACHE]
  [MONITORING | NOMONITORING]
  [UPGRADE [[NOT] INCLUDING DATA]
    [Инструкция_свойств_столбца]
  [{MINIMIZE | NOMINIMIZE} RECORDS_PER_BLOCK]
  [PARALLEL целое | NOPARALLEL]
  [ENABLE | DISABLE ROW MOVEMENT]
  [RENAME TO новое_имя_таблицы]
| { ADD (тип_данных_столбца [DEFAULT выражение] [Инструкция_ограничения_столбца]
    [,столбец тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца] ...)]
    [Инструкция_свойств_столбца]
  | MODIFY
    { (столбец тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца]
       [,столбец тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца] ...)]
      [Инструкция_хранения_LOB]
    | COLUMN столбец [NOT] SUBSTITUTABLE AT ALL LEVELS [FORCE]
    }
  | DROP
    { {PRIMARY KEY | UNIQUE (столбец[,столбец...])}
    | CONSTRAINT имя_ограничения [CASCADE]
    }
  | RENAME COLUMN столбец to новое имя_столбца
  | MODIFY VARRAY элемент_varray ([Инструкция_хранения] [Инструкция_параметров_LOB])
  }
  | { ADD Инструкция_ограничения
    | MODIFY CONSTRAINT имя_ограничения состояние_ограничения
    | RENAME CONSTRAINT имя_ограничения TO новое_имя_ограничения
    | DROP
      { {PRIMARY KEY | UNIQUE (столбец[,столбец...])}
      | CONSTRAINT имя_ограничения [CASCADE]
      }
  }
| MODIFY DEFAULT ATTRIBUTES [FOR PARTITION имя_раздела]
    [TABLESPACE табличное_пространство]
    [LOGGING | NOLOGGING]
    [Инструкция_физических_атрибутов]
    [PCTTHRESHOLD целое]
    [{COMPRESS целое | NOCOMPRESS}]
    [OVERFLOW [Инструкция_физических_атрибутов]
    [ALLOCATE EXTENT ([SIZE целое [K | M]]
     [DATAFILE 'имя_файла']
     [INSTANCE целое])]
   [DEALLOCATE UNUSED [KEEP целое[K | M]]
| MODIFY PARTITION имя_раздела
    Инструкция_параметров_раздела
    [REBUILD] UNUSABLE LOCAL INDEXES]
    { ADD PARTITION имя_раздела
      Инструкция_раздела
      COALESCE SUBPARTITION {UPDATE | INVALIDATE} GLOBAL INDEXES
    | MAPPING TABLE
      {UPDATE BLOCK REFERENCES
      | ALLOCATE EXTENT ([SIZE целое [K | M]]
         [DATAFILE 'имя_файла']
         [INSTANCE целое])
      | DEALLOCATE UNUSED [KEEP целое[K | M]]
```

```
      }
    | {ADD | DROP} VALUES (значение[,значение...])
    }
| MODIFY SUBPARTITION имя_подраздела
    Инструкция_параметров_раздела
    [REBUILD] UNUSABLE LOCAL INDEXES]
    { ADD PARTITION имя_раздела
      Инструкция_раздела
      COALESCE SUBPARTITION {UPDATE | INVALIDATE} GLOBAL INDEXES
    | MAPPING TABLE
      {UPDATE BLOCK REFERENCES
      | ALLOCATE EXTENT ([SIZE целое [K | M]]
        [DATAFILE 'имя_файла']
        [INSTANCE целое])
      | DEALLOCATE UNUSED [KEEP целое[K | M]]
      }
    | {ADD | DROP} VALUES (значение[,значение...])
    }
| MOVE PARTITION имя_раздела [MAPPING TABLE]
    [Инструкция_описания_раздела]
    [{UPDATE | INVALIDATE} GLOBAL INDEXES]
    [PARALLEL целое | NOPARALLEL]
| MOVE SUBPARTITION имя_подраздела
    [{UPDATE | INVALIDATE} GLOBAL INDEXES]
      [PARALLEL целое | NOPARALLEL]
| ADD PARTITION имя_раздела Инструкция_раздела
  COALESCE SUBPARTITION {UPDATE | INVALIDATE} GLOBAL INDEXES
    [PARALLEL целое | NOPARALLEL]
| DROP {PARTITION имя_раздела | SUBPARTITION имя_подраздела}
    [{UPDATE | INVALIDATE} GLOBAL INDEXES]
      [PARALLEL целое | NOPARALLEL]
| RENAME
    { PARTITION имя_раздела TO новое_имя_раздела
    | SUBPARTITION имя_подраздела TO новое_имя_подраздела
    }
| TRUNCATE {PARTITION имя_раздела | SUBPARTITION имя_подраздела}
| SPLIT PARTITION имя_раздела {AT | VALUES} (значение[,значение...])
    [INTO (PARTITION имя_раздела Инструкция_раздела)]
    [{UPDATE | INVALIDATE} GLOBAL INDEXES]
    [PARALLEL целое | NOPARALLEL]
| SPLIT SUBPARTITION имя_подраздела
    VALUES ({значение | NULL}[,{значение | NULL ...])
    [INTO (PARTITION имя_подраздела Инструкция_раздела)]
    [{UPDATE | INVALIDATE} GLOBAL INDEXES]
    [PARALLEL целое | NOPARALLEL]
| MERGE PARTITIONS имя_раздела, имя_раздела
    [INTO (PARTITION имя_раздела Инструкция_раздела)]
    [{UPDATE | INVALIDATE} GLOBAL INDEXES]
    [PARALLEL целое | NOPARALLEL]
| MERGE SUBPARTITIONS имя_подраздела, имя_подраздела
    [INTO (PARTITION имя_подраздела Инструкция_раздела)]
    [{UPDATE | INVALIDATE} GLOBAL INDEXES]
    [PARALLEL целое | NOPARALLEL]
| EXCHANGE {PARTITION имя_раздела | имя_подраздела}
    WITH TABLE имя_таблицы
    [{INCLUDING | EXCLUDING} INDEXES]
```

```
    [{WITH | WITHOUT} VALIDATION]
    [EXCEPTIONS INTO [схема.]имя_таблицы]
    [{UPDATE | INVALIDATE} GLOBAL INDEXES]
}
```

Синтаксис DROP:

```
DROP TABLE [схема.]имя_таблицы
[CASCADE CONSTRAINTS]
```

Создает (за счет определения структуры или копирования существующей таблицы), изменяет или удаляет реляционную таблицу.

При удалении таблицы удаляются все ее строки. Все имеющиеся индексы для этой таблицы также удаляются вне зависимости от того, какой схемой они были созданы и какая схема является их владельцем в настоящее время. Если удаляемая таблица является базовой таблицей представления или на нее ссылается какая-то хранимая процедура, то представление или процедура помечаются как INVALID (но не удаляются). Если таблица является базовой таблицей для моментальной копии, то моментальная копия не удаляется. Аналогично, если для таблицы существует журнал моментальной копии, такой журнал не удаляется.

Ключевые слова

GLOBAL TEMPORARY

Указывает, что создаваемая таблица будет временной таблицей, структура которой будет видна всем сеансам, а данные – только создающему. Временная таблица должна быть создана во временном табличном пространстве. Появилось в Oracle9*i*.

тип_данных

Тип данных, сопоставляемый столбцам.

DEFAULT

Указывает значение по умолчанию для столбца, которое будет использовано, если во вставляемых строках значение для данного столбца отсутствует. Выражение должно соответствовать *типу_данных* столбца.

ON COMMIT

Указывает, доступны ли данные временной таблицы на время транзакции или же всего сеанса. Применяется только к временным таблицам. Появилось в Oracle9*i*.

TABLESPACE

Указывает имя табличного пространства, в котором будет храниться таблица. Если ключевое слово пропущено, то будет использовано табличное пространство по умолчанию для владельца схемы.

CLUSTER

Указывает, что таблица является частью кластера. Перечисленные в инструкции столбцы – это столбцы таблицы, соответствующие столбцам кластера.

ORGANIZATION HEAP

Указывает, что строки не хранятся в каком-то определенном порядке (по умолчанию).

ORGANIZATION INDEX

Указывает, что таблица создается как индексированная, т. е. строки данных фактически хранятся в индексе, определенном на первичном ключе таблицы.

MAPPING TABLE

Указывает, что сервер Oracle создаст отображение локальных ROWID на физические и сохранит их в «куче» (*heap-organized table*). Такое отображение необходимо для того, чтобы создать битовый индекс для индексированной таблицы. По умолчанию принимает значение NOMAPPING. Появилось в Oracle9*i*.

PCTTHRESHOLD

Указывает (в процентах) объем памяти в каждом блоке индекса, зарезервированный для строк данных. Любая часть строки данных, которая не поместится в отведенное пространство, будет помещена в сегмент переполнения. Появилось в Oracle9*i*.

INCLUDING

Определяет точку, которая делит строку таблицы на составляющие индекса и переполнения. Все последующие столбцы (за исключением столбцов первичного ключа) будут сохранены в сегменте переполнения.

OVERFLOW

Указывает, что строки индексированной таблицы, для которых превышено значение PCTTHRESHOLD, будут помещены в сегмент, описанный в данной инструкции.

ORGANIZATION EXTERNAL

Указывает, что сервер Oracle создаст внешнюю таблицу, которая будет доступна только для чтения, ее метаданные будут храниться в БД, а данные – вне БД. Появилось в Oracle9*i*.

TYPE

Драйвер доступа к внешней таблице. Драйвер доступа – это API, интерпретирующий внешние данные для базы данных. Если не указать TYPE, то сервер Oracle использует драйвер доступа ORACLE_LOADER.

DEFAULT DIRECTORY

Объект каталога по умолчанию, соответствующий каталогу файловой системы, в котором могут находиться внешние источники данных. Каталог по умолчанию также может использоваться драйвером доступа для хранения служебных файлов, таких как журналы ошибок.

ACCESS PARAMETERS

Значения параметров конкретного драйвера доступа для данной внешней таблицы. *Непрозрачная_спецификация* содержит перечисление параметров и их значений. Инструкция USING CLOB *подзапрос* позволяет извлечь параметры и их значения посредством подзапроса, который не может включать в себя операции над множествами и инструкцию ORDER BY и должен возвращать одну строку, содержащую один элемент типа CLOB. Сервер Oracle не интерпретирует данную инструкцию; интерпретацией информации в контексте внешних данных занимается драйвер доступа.

LOCATION

Один или несколько внешних источников данных. Обычно *указатель_местоположения* представляет собой файл, но это не обязательно. Сервер Oracle не интерпретирует данную инструкцию; интерпретацией информации в контексте внешних данных занимается драйвер доступа.

REJECT LIMIT

Количество ошибок преобразования, которые могут произойти в процессе выполнения запроса внешних данных, прежде чем будет возвращена ошибка Oracle и запрос будет прерван. Значение по умолчанию равно 0.

LOB

Задает параметры хранения для данных LOB.

элемент_LOB

Имя столбца LOB.

STORE AS

Определяет имя сегмента данных LOB.

ENABLE STORAGE IN ROW

Указывает, что значение LOB будет храниться в строке. Если данные ключевые слова указаны для индексированной таблицы, необходимо также указать OVERFLOW. Это поведение по умолчанию.

DISABLE STORAGE IN ROW

Указывает, что значение LOB будет храниться вне строки.

CHUNK

Указывает, что для работы с LOB должно быть выделено указанное количество байт. Обратите внимание, что значение параметра *целое* будет округлено до числа, кратного размеру блока Oracle.

PCTVERSION

Указывает максимальный объем (в процентах) области хранения LOB, отводимый для создания новых версий LOB.

ENABLE ROW MOVEMENT

Означает, что строка может быть перемещена в другой раздел или подраздел, если это требуется в результате обновления ключа. Появилось в Oracle8*i*.

DISABLE ROW MOVEMENT

Означает, что строки не могут быть перемещены в другой раздел или подраздел, и возвращает ошибку, если это требуется в результате обновления ключа. Появилось в Oracle8*i*.

MONITORING

Означает, что для таблицы будет собираться статистика изменений. Появилось в Oracle9*i*.

NOMONITORING

Означает, что для таблицы не будет собираться статистика изменений. Это поведение по умолчанию. Появилось в Oracle9*i*.

ROWDEPENDENCIES

Указывает, что для данной таблицы разрешено отслеживание зависимостей на уровне строки. Такая возможность применяется в основном в реплицируемой среде и увеличивает длину каждой строки на 6 байт. Появилось в Oracle9*i*.

NOROWDEPENDENCIES

Указывает, что для данной таблицы не разрешено отслеживание зависимостей на уровне строки. Это поведение по умолчанию.

ENABLE

Указывает, что ограничение будет применяться ко всем новым данным таблицы.

DISABLE

Означает, что для данной таблицы ограничение не применяется.

VALIDATE

Если данное ключевое слово указано вместе с ENABLE, то сервер Oracle проверяет соответствие всех существующих данных таблицы ограничению целостности.

NOVALIDATE

Если данное ключевое слово указано вместе с ENABLE, то сервер Oracle не проверяет соответствие всех существующих данных таблицы ограничению целостности, но удостоверяется в том, что новые данные, добавляемые в таблицу, удовлетворяют ограничению целостности.

UNIQUE

Включает и отключает ограничение уникальности, определенное для указанного столбца или комбинации столбцов.

PRIMARY KEY

Включает и отключает использование ограничения первичного ключа таблицы.

CONSTRAINT

Включает и отключает использование ограничения целостности с именем *имя_ограничения*.

USING INDEX

Определяет параметры индекса, применяемого для наложения ограничения целостности.

EXCEPTIONS INTO

Задает имя таблицы, в которую сервер Oracle помещает информацию о строках, нарушающих ограничение. Прежде чем использовать данное ключевое слово, необходимо явно создать эту таблицу, запустив сценарий *UTLEXCPT1.SQL*.

KEEP

Указывает, что индекс, который сервер Oracle использовал для наложения ограничения уникальности или ограничения первичного ключа, будет сохранен.

DROP

Указывает, что индекс, который сервер Oracle использовал для наложения ограничения уникальности или ограничения первичного ключа, будет удален.

CASCADE

Указывает, что применение любых ограничений целостности, зависящих от указанного ограничения целостности, будет отключено. Для того чтобы отключить применение ограничения уникального или первичного ключа, определяющего ссылочную целостность, необходимо указать данную инструкцию.

Дополнительные ключевые слова для ALTER

ADD SUPPLEMENTAL LOG GROUP

Означает, что будет добавлена дополнительная журнальная группа и указывает имя файла. Появилось в Oracle9*i*.

DROP SUPPLEMENTAL LOG GROUP

Означает, что указанная дополнительная журнальная группа будет удалена. Появилось в Oracle9*i*.

ALLOCATE EXTENT

Явно выделяет новый экстент для таблицы, используя указанные параметры.

SIZE

Размер выделяемого экстента.

DATAFILE

Имя файла данных, в который будет добавлен экстент.

INSTANCE

Номер экземпляра, в который будет добавлен экстент.

DEALLOCATE UNUSED

Указывает, что неиспользуемое пространство в конце таблицы, раздела, подраздела, сегмента данных переполнения, сегмента данных LOB или индекса LOB будет освобождено и станет доступным для других сегментов табличного пространства.

KEEP

Указывает количество байт сверх маркера максимального заполнения, которые останутся в составе таблицы, сегмента данных переполнения, сегмента данных LOB или индекса LOB после освобождения неиспользуемого пространства.

UPGRADE

Означает, что метаданные таблицы должны быть преобразованы так, чтобы соответствовать последней версии каждого из упоминаемых типов. Данное ключевое слово применяется только к таблицам, содержащим объектные столбцы. Появилось в Oracle9*i*.

INCLUDING DATA

Указывает, что сервер Oracle преобразует данные таблицы к формату последней версии типа (если преобразование не было выполнено при изменении типа). В процессе обновления таблицы можно определить хранилище для любого нового столбца. Это поведение по умолчанию. Если указаны ключевые слова NOT INCLUDING DATA, то сервер Oracle оставит даннные столбца неизменными.

MINIMIZE RECORDS_PER_BLOCK

Указывает, что сервер Oracle должен вычислить наибольшее количество строк в каждом блоке таблицы (исследуя существующие табличные данные) и ограничить будущие вставки так, чтобы вычисленное количество не было превышено ни для одного блока. Для того чтобы отменить такие вычисления и ограничения, можно указать ключевое слово NOMINIMIZE (это поведение по умолчанию).

RENAME TO

Означает, что имя таблицы должно быть заменено на указанное.

ADD

Указывает имя и характеристики нового столбца, добавляемого в таблицу.

MODIFY

Указывает, что характеристики столбца будут изменены.

DROP PRIMARY KEY

Указывает, что ограничение первичного ключа для данной таблицы удаляется.

DROP UNIQUE

Указывает, что удаляется ограничение уникальности для столбца.

RENAME COLUMN

Означает, что имя столбца в данной таблице будет изменено. Появилось в Oracle9*i*.

MODIFY VARRAY

Указывает, что будут изменены параметры хранения VARRAY. Появилось в Oracle9*i*.

ADD Инструкция_ограничения

Указывает, что будет добавлено ограничение для таблицы.

MODIFY CONSTRAINT

Определяет, что указанное ограничение будет изменено.

RENAME CONSTRAINT

Изменяет имя существующего ограничения для таблицы. Появилось в Oracle9*i* Release 2.

DROP CONSTRAINT

Удаляет ограничение целостности.

RENAME COLUMN

Изменяет название существующего столбца таблицы. Появилось в Oracle9*i* Release 2.

MODIFY DEFAULT ATTRIBUTES

Указывает новые значения по умолчанию для атрибутов таблицы. Созданные впоследствии разделы и LOB-разделы унаследуют эти значения, если только они не будут явно подменены при создании раздела или LOB-раздела. На уже существующие разделы и LOB-разделы данная инструкция действия не оказывает.

FOR PARTITION

Означает, что атрибуты по умолчанию будут применены только к названной таблице. FOR PARTITION применяется только к таблицам с комбинированным секционированием.

MODIFY PARTITION

Означает, что характеристики указанного раздела будут изменены.

REBUILD UNUSABLE LOCAL INDEXES

Указывает, что локальные индексы обновляемого раздела будут перестроены. Если данная инструкция не указана, то индекс будет помечен как INVALID.

ADD PARTITION

Указывает, что для данной таблицы будет создан новый раздел.

COALESCE SUBPARTITION

Указывает, что сервер Oracle выбирает хеш-подраздел, распределяет его содержимое между оставшимися подразделами (которые определены хеш-функцией) и удаляет выбранный подраздел. Появилось в Oracle8*i*.

UPDATE GLOBAL INDEXES

Означает, что если для таблицы определен глобальный индекс, то сервер Oracle обновляет весь индекс целиком, а не только тот раздел, для которого выдана команда.

INVALIDATE GLOBAL INDEXES

Означает, что если для таблицы определен глобальный индекс, то сервер Oracle переводит в состояние INVALID весь индекс целиком, а не только тот раздел, для которого выдана команда.

ADD VALUES

Указывает, что сервер Oracle расширяет список значений раздела или подраздела, включая в него дополнительные значения. Появилась в Oracle9*i* Release 2.

DROP VALUES

Означает, что сервер Oracle сужает список значений раздела или подраздела, удаляя из него указанные значения. Появилась в Oracle9*i* Release 2.

MODIFY SUBPARTITION

Указывает, что характеристики названного подраздела будут изменены.

MOVE PARTITION

Означает, что сервер Oracle перемещает указанный раздел в другой сегмент.

MOVE SUBPARTITION

Означает, что сервер Oracle перемещает указанный подраздел в другой сегмент.

DROP PARTITION

Указывает, что названный раздел будет удален.

DROP SUBPARTITION

Указывает, что названный подраздел будет удален.

RENAME PARTITION

Указывает, что имя раздела таблицы, заданное параметром *имя_раздела*, будет заменено на *новое_имя_раздела*.

RENAME SUBPARTITION

Указывает, что имя раздела таблицы, заданное параметром *имя_подраздела* будет заменено на *новое_имя_подраздела*.

TRUNCATE PARTITION

Указывает, что все строки данных раздела будут удалены подобно тому, как это происходит при выполнении команды TRUNCATE TABLE. Для индексированной таблицы также очищаются все соответствующие разделы отображений.

TRUNCATE SUBPARTITION

Указывает, что все строки данных подраздела будут удалены подобно тому, как это происходит при выполнении команды TRUNCATE TABLE. Для индексированной таблицы также очищаются все соответствующие разделы отображений.

SPLIT

Указывает, что сервер Oracle создает два новых раздела или подраздела, каждый с новым сегментом, новыми физическими атрибутами и новыми начальными экстентами. Сегмент, сопоставленный исходному разделу или подразделу, удаляется.

MERGE

Указывает, что сервер Oracle сводит содержимое двух разделов или подразделов таблицы в один новый раздел, после чего удаляет исходные разделы.

EXCHANGE

Указывает, что сервер Oracle осуществляет обмен сегментов данных и индекса для одной из следующих комбинаций: несекционированной таблицы и хеш-раздела, спискового или диапазонного раздела (или одного хеш- или спискового подраздела); хеш-секционированной таблицы и хеш-подразделов таблицы с комбинированным (диапазонным и по хеш-ключу) секционированием; таблицы со списковым секционированием и списковых подразделов диапазонного раздела таблицы с комбинированным (диапазонным и списковым) секционированием. В любом случае структура таблицы и раздела или подраздела, для которых осуществляется обмен, включая ключи секционирования, должны совпадать. В случае списковых разделов и подразделов соответствующие списки значений также должны совпадать.

Дополнительные ключевые слова для DROP

CASCADE CONSTRAINTS

Указывает, что все ограничения ссылочной целостности, ссылающиеся на первичный и уникальный ключи удаляемой таблицы, также будут удалены.

Общие ключевые слова и инструкции: *столбец*, *COMPRESS*, *выражение*, *целое*, *LOGGING*, NOCOMPRESS, NOLOGGING, NOPARALLEL, PARALLEL, *имя_раздела*, *схема*, *имя_подраздела*, *имя_таблицы*, *табличное_пространство*, *Инструкция_ограничения_столбца*, *Инструкция_состояния_ограничения*, *Инструкция_параметров_LOB*, *Инструкция_хранения_LOB*, *Инструкция_раздела*, *Инструкция_описания_раздела*, *Инструкция_физических_атрибутов*, *Инструкция_хранения*, *Инструкция_ограничения_таблицы*, *Инструкция_хранения_Varray*.

CREATE/ALTER/DROP TABLE (Объектный синтаксис)

Синтаксис CREATE:

```
CREATE [GLOBAL TEMPORARY] TABLE [схема.]имя_таблицы
  OF [схема.]объектный_тип
( { { столбец | атрибут} [DEFAULT выражение]
    { SCOPE IS [схема.] имя_видимой_таблицы]
    | WITH ROWID
    | [CONSTRAINT имя_ограничения]  REFERENCES [схема.]объектная_таблица
       [ON DELETE CASCADE]
    }
|     { SCOPE FOR ({столбец | атрибут}) IS [схема.]имя_таблицы |
         REF ({столбец | атрибут}) WITH ROWID
     | [CONSTRAINT имя_ограничения] FOREIGN KEY (столбец)
         REFERENCES [схема.]объектная_таблица [ON DELETE CASCADE]
         [Инструкция_состояния_ограничения]
     }
)
[OBJECT ID {SYSTEM GENERATED | PRIMARY KEY}]
[ON COMMIT {DELETE | PRESERVE} ROWS]
[OIDINDEX [имя_индекса] ([Инструкция_физических_атрибутов]
[TABLESPACE табличное_пространство])]
[Инструкция_физических_атрибутов]
```

```
[TABLESPACE табличное_пространство]
[Инструкция_хранения]
[LOGGING | NOLOGGING]
[CLUSTER (столбец[,столбец ...]]
[ { ORGANIZATION HEAP
     [Инструкция_физических_атрибутов]
     [TABLESPACE табличное_пространство]
     [Инструкция_хранения]
     [LOGGING | NOLOGGING]
  | ORGANIZATION INDEX
     [PCTTHRESHOLD целое]
     [COMPRESS целое | NOCOMPRESS]
     [Инструкция_физических_атрибутов]
     [TABLESPACE табличное_пространство]
     [Инструкция_хранения]
     [LOGGING | NOLOGGING]
     [[INCLUDING столбец] OVERFLOW
       [Инструкция_физических_атрибутов]
       [TABLESPACE табличное_пространство]
       [Инструкция_хранения]
       [LOGGING | NOLOGGING]]
  }
[LOB
  {(элемент_LOB[,элемент_LOB...]) STORE AS
    (Инструкция_параметров_LOB)
  | (элемент_LOB) STORE AS [(имя_сегмента_LOB)]
     [(Инструкция_параметров_LOB)] ]
     [VARRAY элемент_varray STORE AS LOB [имя_сегмента_LOB]
       [(Инструкция_параметров_LOB)]]
     [NESTED TABLE вложенная_таблица STORE AS таблица_хранения
       (имя_таблицы [Инструкция_физических_атрибутов]
       [TABLESPACE табличное_пространство] [LOGGING | NOLOGGING]]
       [[LOB {(элемент_LOB[,элемент_LOB...]) STORE AS
        ([Инструкция_параметров_LOB)
|      (элемент_LOB) STORE AS [(имя_сегмента_LOB)]
     [(Инструкция_параметров_LOB)]}
} ]
[RETURN AS {LOCATOR | VALUE}]
[Инструкция_секционирования]
[ENABLE | DISABLE ROW MOVEMENT]
[CACHE | NOCACHE]
[MONITORING | NOMONITORING]
[PARALLEL целое | NOPARALLEL]
[{ENABLE | DISABLE} [VALIDATE | NOVALIDATE]
  { UNIQUE (столбец[, столбец...]
  | PRIMARY KEY
  | CONSTRAINT имя_ограничения
  }
  [USING INDEX
    [TABLESPACE табличное_пространство]
    [Инструкция_физических_атрибутов]
    [Инструкция_хранения]
    [NOSORT]
    [LOGGING | NOLOGGING]]
    [EXCEPTIONS INTO [схема.]имя_таблицы [CASCADE]]
[AS подзапрос]
```

Синтаксис ALTER:

```
ALTER TABLE [схема.]имя_таблицы
{ Инструкция_физических_атрибутов
| { ADD ([столбец  тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца]
    [,столбец тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца] ...)]
  | MODIFY
    { ([столбец  тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца]
       [,столбец тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца] ...)]
      [Инструкция_хранения_LOB]
    | COLUMN столбец [NOT] SUBSTITUTABLE AT ALL LEVELS [FORCE]
    }
  }
  { { столбец | атрибут} [DEFAULT выражение]
    { SCOPE IS [схема.]имя_видимой_таблицы]
    | WITH ROWID
    | [CONSTRAINT имя_ограничения]  REFERENCES [схема.]объектная_таблица
        [ON DELETE CASCADE]
    }
 |     { SCOPE FOR ({столбец | атрибут}) IS [схема.]имя_таблицы
 | REF ({столбец | атрибут}) WITH ROWID
 | [CONSTRAINT имя_ограничения] FOREIGN KEY (столбец)
     REFERENCES [схема.]объектная_таблица [ON DELETE CASCADE]
     [Инструкция_состояния_ограничения]
 }
 )
 [OBJECT ID {SYSTEM GENERATED | PRIMARY KEY}]
 [ON COMMIT {DELETE | PRESERVE} ROWS]
 [OIDINDEX [имя_индекса] ([Инструкция_физических_атрибутов]
 [TABLESPACE табличное_пространство])]
 [Инструкция_физических_атрибутов]
 [TABLESPACE табличное_пространство]
 [Инструкция_хранения]
 [LOGGING | NOLOGGING]
 [LOB
  {(элемент_LOB[,элемент_LOB...]) STORE AS
    (Инструкция_параметров_LOB)
  | (элемент_LOB) STORE AS [(имя_сегмента_LOB)]
     [(Инструкция_параметров_LOB)] ]
     [VARRAY элемент_varray STORE AS LOB [имя_сегмента_LOB]
       [(Инструкция_параметров_LOB)]]
     [NESTED TABLE вложенная_таблица STORE AS таблица_хранения
       (имя_таблицы [Инструкция_физических_атрибутов]
       [TABLESPACE табличное_пространство] [LOGGING | NOLOGGING]]
       [LOB (элемент_LOB[,элемент_LOB...]) STORE AS
        (Инструкция_параметров_LOB)]
  | (элемент_LOB) STORE AS [(имя_сегмента_LOB)]
     [(Инструкция_параметров_LOB)]
  ) ]
[RETURN AS {LOCATOR | VALUE}]
[Инструкция_секционирования]
[ENABLE | DISABLE ROW MOVEMENT]
[CACHE | NOCACHE]
[MONITORING | NOMONITORING]
[PARALLEL целое | NOPARALLEL]
[{ENABLE | DISABLE} [VALIDATE | NOVALIDATE]
  { UNIQUE (столбец[, столбец...])
```

```
  | PRIMARY KEY
  | CONSTRAINT имя_ограничения
  }
  [USING INDEX
    [TABLESPACE табличное_пространство]
    [Инструкция_физических_атрибутов]
    [Инструкция_хранения]
    [NOSORT]
    [LOGGING | NOLOGGING]]
    [EXCEPTIONS INTO [схема.]имя_таблицы
  [CASCADE]]]
```

Синтаксис DROP:

```
DROP TABLE [схема.]имя_таблицы
[CASCADE CONSTRAINTS]
```

Создает, изменяет или удаляет объектную таблицу.

Ключевые слова

GLOBAL TEMPORARY

Указывает, что создаваемая таблица будет временной таблицей, структура которой будет видна всем сеансам, а данные – только создающему. Появилось в Oracle9*i*.

OF

Указывает базовый объектный тип для создаваемой таблицы (см. также описание команды CREATE TYPE ранее в этой же главе).

атрибут

Уточненное имя столбца для элемента объекта.

DEFAULT

Задает значение по умолчанию для столбца, которое будет использоваться в команде INSERT, если значение не будет задано явно. Выражение должно соответствовать типу данных столбца.

SCOPE IS

Указывает, что каждое значение типа REF в столбце может ссылаться только на таблицу, определяемую параметром *имя_видимой_таблицы*.

WITH ROWID

Указывает, что ROWID должен храниться вместе со значением REF для данного столбца.

CONSTRAINT

Задает имя ограничения.

REFERENCES

Указывает имя таблицы, которая должна использоваться в ограничении ссылочной целостности для значения REF данного столбца.

ON DELETE CASCADE

Указывает, что при удалении строки из таблицы строки, содержащие зависимый внешний ключ, также будут автоматически удалены.

SCOPE FOR

Указывает, что значение REF для *столбца* или *атрибута* должно ссылаться на таблицу, определяемую параметром *имя_таблицы*.

REF WITH ROWID

Указывает, что ROWID должен храниться вместе со значением REF для данного столбца.

FOREIGN KEY

Определяет имя столбца, который должен использоваться в ограничении внешнего ключа для значения REF данного столбца.

OBJECT ID

Указывает, должен ли идентификатор объекта генерироваться системой (SYSTEM GENERATED – это значение по умолчанию) или же основываться на первичном ключе таблицы. Обратите внимание, что если указаны ключевые слова PRIMARY KEY, то ограничение первичного ключа должно быть определено.

ON COMMIT

Определяет, удаляются или сохраняются строки временной таблицы после выдачи команды COMMIT. Появилось в Oracle9*i*.

OIDINDEX

Задает имя индекса или физические атрибуты индекса для хранения значений идентификаторов объектов данной таблицы.

TABLESPACE

Указывает имя табличного пространства, в котором будет храниться таблица. Если ключевое слово TABLESPACE пропущено, сервер Oracle использует для владельца схемы табличное пространство по умолчанию.

ORGANIZATION HEAP

Указывает, что строки не хранятся в каком-то определенном порядке (это поведение по умолчанию).

ORGANIZATION INDEX

Указывает, что таблица создается как индексированная, т. е. строки данных фактически хранятся в индексе, который определен на первичном ключе таблицы.

PCTTHRESHOLD

Указывает объем памяти (в процентах) в каждом блоке индекса, зарезервированный для строк данных. Любая часть строки данных, которая не поместится в отведенное пространство, будет помещена в сегмент переполнения.

INCLUDING столбец

Определяет точку, которая делит строку таблицы на составляющие индекса и переполнения. Все последующие столбцы (за исключением столбцов первичного ключа) будут сохранены в сегменте переполнения.

LOB

Определяет атрибуты хранения для данных LOB, определяемых при помощи парамера *элемент_LOB*.

STORE AS

Указывает имя сегмента данных LOB.

VARRAY

Задает параметры хранения для LOB, в котором будет храниться тип VARRAY.

NESTED TABLE

Определяет параметры хранения для *вложенной_таблицы*, которая будет храниться в *таблице_хранения*.

RETURN AS LOCATOR

Указывает, что для копии данной вложенной таблицы будет возвращен указатель коллекции.

RETURN AS VALUE

Указывает, что будет возвращена копия вложенной таблицы.

ENABLE ROW MOVEMENT

Означает, что строка может быть перемещена в другой раздел или подраздел, если это требуется в результате обновления ключа.

DISABLE ROW MOVEMENT

Означает, что строки не могут быть перемещены в другой раздел или подраздел, и возвращает ошибку, если это требуется в результате обновления ключа.

USING INDEX

Определяет параметры индекса, который используется для наложения ограничения.

EXCEPTIONS INTO

Задает имя таблицы, в которую сервер Oracle помещает информацию о строках, нарушающих ограничение. Прежде чем использовать данное ключевое слово, необходимо явно создать эту таблицу, запустив сценарий *UTLEXCPT1.SQL*.

CASCADE

В случае применения DISABLE данное ключевое слово отключает использование всех ограничений целостности, которые зависят от указанного ограничения.

AS подзапрос

Указывает подзапрос, который будет применяться для вставки строк в таблицу при создании. Если определения столбцов в команде CREATE TABLE отсутствуют, то имена столбцов, типы данных и ограничения будут скопированы из таблицы, на которую ссылается подзапрос.

Общие ключевые слова и инструкции: *столбец*, *Инструкция_ограничения_столбца*, *COMPRESS*, *Инструкция_состояния_ограничения*, *выражение*, *имя_файла*, *имя_индекса*, *целое*, *Инструкция_параметров_LOB*, LOGGING, NOCOMPRESS, NOLOGGING, NOPARALLEL, PARALLEL, *Инструкция_раздела*, *имя_раздела*, *Инструкция_физических_атрибутов*, *схема*, *Инструкция_хранения*, *имя_подраздела*, *имя_таблицы*, *табличное_пространство*.

CREATE/ALTER/DROP TABLE (XML-синтаксис)

Синтаксис CREATE:

```
CREATE TABLE [схема.]таблица OF XMLTYPE
[XMLTYPE STORE AS
  {OBJECT RELATIONAL
  | CLOB
    [{имя_сегмента_LOB [(Инструкция_параметров_LOB)]
     | Инструкция_параметров_LOB
     }]
```

```
  }]
[[XMLSCHEMA URL_XML-схемы] ELEMENT
  {element
  | URL_XML-схемы # элемент
  }]
```

Синтаксис ALTER:

```
ALTER TABLE [схема.имя_таблицы
{ ADD (столбец  тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца]
  [,столбец тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца] ...)]
| MODIFY
  { (столбец  тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца]
     [,столбец тип_данных [DEFAULT выражение] [Инструкция_ограничения_столбца] ...)]
XMLTYPE COLUMN столбец STORE AS
  {OBJECT RELATIONAL
  | CLOB
    [{ имя_сегмента_LOB [(Инструкция_параметров_LOB)]
     | Инструкция_параметров_LOB
     }]
  }]
[[XMLSCHEMA URL_XML-схемы] ELEMENT
  {element
  | URL_XML-схемы # элемент
  }]
```

Синтаксис DROP:

```
DROP TABLE [схема.]имя_таблицы
[CASCADE CONSTRAINTS]
```

Создает, изменяет или удаляет таблицу XML. Появилась в Oracle9*i* Release 2.

Ключевые слова

STORE AS

Указывает, каким образом сервер Oracle будет управлять хранением базовых столбцов. Появилось в Oracle9*i*.

OBJECT RELATIONAL

Означает, что сервер Oracle будет хранить данные XMLType в объектно-реляционных столбцах. Если указаны ключевые слова OBJECT RELATIONAL, необходимо также определить XML-схему в инструкции XMLSCHEMA, при этом схема должна быть уже зарегистрирована (с помощью встроенного пакета DBMS_XMLSCHEMA). Сервер Oracle создаст таблицу в соответствии с зарегистрированной схемой.

CLOB

Указывает, что сервер Oracle будет хранить XML-данные в столбце CLOB. Если используется ключевое слово CLOB, то можно также указать имя сегмента LOB, *Инструкцию_параметров_LOB* или и то, и другое.

XMLSCHEMA

Задает URL зарегистрированной схемы XML (в инструкции XMLSCHEMA или в составе инструкции ELEMENT) и имя элемента XML. Элемент указать необходимо, а URL является необязательным. Если вы указываете URL схемы XML, то такая схема должна быть уже зарегистрирована (с помощью пакета DBMS_XMLSCHEMA).

XMLTYPE COLUMN

Определяет, какой XML-столбец будет изменен или добавлен. Появилось в Oracle9*i*.

Общие ключевые слова и инструкции: *столбец*, *Инструкция_ограничения_столбца*, *выражение*, *Инструкция_параметров_LOB*, *схема*, *имя_таблицы*, *табличное_пространство*.

CREATE/ALTER/DROP TABLESPACE

Синтаксис CREATE:

```
CREATE [UNDO] TABLESPACE табличное_пространство
DATAFILE 'имя_файла' [SIZE целое [K | M] [REUSE]] [Инструкция_авторасширения]
 [,'имя_файла' [SIZE целое [K | M] [REUSE]] [Инструкция_авторасширения]]
DEFAULT Инструкция_хранения
[ONLINE | OFFLINE]
[PERMANENT | TEMPORARY]
[LOGGING | NOLOGGING]
[MINIMUM EXTENT целое]
[EXTENT MANAGEMENT
 {DICTIONARY
 | LOCAL [AUTOALLOCATE
 | UNIFORM [SIZE целое [ K | M ]]]
 }]
[SEGMENT SPACE MANAGEMENT {MANUAL | AUTO}]
```

Синтаксис ALTER:

```
ALTER TABLESPACE табличное_пространство
{[ADD DATAFILE имя_файла [SIZE целое [K | M]] [REUSE]]
| [Инструкция_авторасширения]
| [RENAME DATAFILE 'имя_файла1' TO 'имя_файла2']
| [DEFAULT Инструкция_хранения
| [{ONLINE]|OFFLINE}]
| [{PERMANENT | TEMPORARY}]
| [{BEGIN BACKUP | END BACKUP}]
| [{LOGGING | NOLOGGING}]
| [MAXIMUM EXTENT целое]
}
```

Синтаксис DROP:

```
DROP TABLESPACE табличное_пространство
[INCLUDING CONTENTS [AND DATAFILES]] [CASCADE CONSTRAINTS]
```

Создает, изменяет или удаляет табличное пространство, при необходимости указывая используемые по умолчанию характеристики хранения объектов, создаваемых впоследствии в данном табличном пространстве.

Ключевые слова

UNDO

Означает, что будет создано табличное пространство UNDO. Появилось в Oracle9*i*.

DATAFILE

Определяет имя файла данных операционной системы для данного табличного пространства. Ключевое слово SIZE необходимо, если файл еще не существует. Если файл уже существует, необходимо указать ключевое слово REUSE.

DEFAULT Инструкция_хранения

Задает физические параметры хранения (подробная информация приведена в описании инструкции *Инструкция_хранения* в разделе «Общие инструкции SQL»).

ONLINE

Переводит табличное пространство в оперативный режим после создания (это поведение по умолчанию).

OFFLINE

Оставляет табличное пространство автономным.

PERMANENT

Указывает, что табличное пространство может содержать постоянные объекты.

TEMPORARY

Указывает, что табличное пространство будет содержать только временные объекты.

MINIMUM EXTENT

Означает, что размер каждого используемого или свободного экстента табличного пространства как минимум равен и является кратным величине, заданной параметром *целое*. Этот параметр управляет фрагментацией свободного пространства.

EXTENT MANAGEMENT

Определяет, каким образом будут управляться экстенты табличного пространства. Появилось в Oracle9*i*.

DICTIONARY

Указывает, что пространство будет управляться с помощью таблиц словаря данных.

LOCAL

Пространство управляется локально с помощью битового отображения в табличном пространстве.

AUTOALLOCATE

Табличное пространство управляется системой, и пользователи не могут указать размер экстента.

UNIFORM

Табличное пространство выделяется экстентами одинакового размера, заданного параметром SIZE (в байтах). Для того чтобы задать размер в килобайтах или мегабайтах, укажите соответственно K и M. По умолчанию все экстенты имеют размер 1 мегабайт.

SEGMENT SPACE MANAGEMENT

Определяет, будет ли сервер Oracle отслеживать используемое и свободное пространство в сегментах табличного пространства при помощи списков свободных блоков или битовых отображений. Инструкция применяется только к локально управляемым постоянным табличным пространствам. Появилось в Oracle9*i*.

MANUAL

Сервер Oracle будет управлять свободным пространством сегментов табличного пространства при помощи списков свободных блоков.

AUTO

Сервер Oracle будет управлять свободным пространством сегментов табличного пространства при помощи битового отображения. Если указать ключевое

слово AUTO, сервер Oracle будет игнорировать применение FREELIST и FREELIST GROUPS в последующих спецификациях хранения объектов в данном табличном пространстве.

BEGIN BACKUP

Оповещает сервер Oracle о том, что выполняется резервное копирование табличного пространства, в связи с чем журнальный файл занимается накапливанием всех изменений блоков для данного табличного пространства. Имейте в виду, что фактически данная команда не выполняет резервное копирование, она лишь сообщает о его начале.

END BACKUP

Оповещает сервер Oracle о том, что резервное копирование табличного пространства разрешено, в связи с этим журнал может вернуться в режим нормальной работы.

INCLUDING CONTENTS

Указывает, что все объекты, содержащиеся в данном табличном пространстве, будут автоматически удалены. Если данное ключевое слово не указано, а табличное пространство содержит какие-то объекты, то команда не будет выполнена.

AND DATAFILES

Означает, что сервер Oracle должен удалить соответствующие файлы операционной системы. Появилось в Oracle9*i*.

CASCADE CONSTRAINTS

Указывает, что все ограничения ссылочной целостности таблиц, расположенных вне данного табличного пространства, которые ссылаются на первичный и уникальный ключи таблиц данного табличного пространства, будут удалены.

Общие ключевые слова и инструкции: *Инструкция_авторасширения*, *имя_файла*, *целое*, LOGGING, NOLOGGING, *схема*, *Инструкция_хранения*, *табличное_пространство*.

CREATE TEMPORARY TABLESPACE

```
CREATE TEMPORARY TABLESPACE
TEMPFILE 'имя_файла' [SIZE целое [K | M] [REUSE]]
[Инструкция_авторасширения]
[EXTENT MANAGEMENT LOCAL]
[UNIFORM] [SIZE целое [K | M]]
```

Создает временное табличное пространство, используемое для хранения временных объектов (которые хранятся только в пределах одного сеанса). Как и любое другое табличное пространство, временное табличное пространство может быть удалено при помощи команды DROP TABLESPACE. Восстановление носителя не затрагивает временные табличные пространства.

Ключевые слова

TEMPFILE

Указывает имя файла данных операционной системы для данного табличного пространства.

SIZE

Указывает размер файла в байтах, килобайтах (K) или мегабайтах (M). Ключевое слово SIZE необходимо, если файл еще не существует. Если файл уже существует, необходимо указать ключевое слово REUSE.

REUSE

Означает, что файл операционной системы уже должен существовать и он будет повторно использован для данного табличного пространства.

INITIAL

Указывает размер первого экстента для нового объекта в байтах, килобайтах (K) или мегабайтах (M). Если этот размер не кратен размеру блока БД, то значение будет округлено до кратного размеру блока БД.

EXTENT MANAGEMENT LOCAL

Указывает, что временное табличное пространство будет локально управляемым, т. е. некоторая часть табличного пространства отводится под битовое отображение. Появилось в Oracle9*i*.

UNIFORM SIZE

Задает размер экстентов временного табличного пространства в байтах, килобайтах (K) или мегабайтах (M) и указывает, что все экстенты временного табличного пространства имеют одинаковый размер. По умолчанию все экстенты имеют размер 1 Мбайт.

Общие ключевые слова и инструкции: *имя_файла*, *целое*.

CREATE/ALTER/DROP TRIGGER

Создает, изменяет или удаляет триггер PL/SQL, который хранится в базе данных и исполняется, когда наступают определенные события в БД. Информация о синтаксисе и использовании приведена в главе 9.

CREATE/ALTER/DROP TYPE

Синтаксис создания неполного типа:

```
CREATE [OR REPLACE] TYPE [схема.]имя_типа
```

Синтаксис создания объекта:

```
CREATE [OR REPLACE] TYPE [схема.]имя_типа
[AUTHID {CURRENT_USER | DEFINER}]
{IS | AS} OBJECT (атрибут тип_данных[, атрибут тип_данных...]),
  {MEMBER | STATIC} {спецификация_процедуры | спецификация_функции}
  [,PRAGMA RESTRICT_REFERENCES ({имя_метода | DEFAULT},
  {RNDS|WNDS | RNPS | WNPS | TRUST}
   [,{RNDS | WNDS | RNPS | WNPS | TRUST}...] [,PRAGMA...]
  {MEMBER | STATIC} {спецификация_процедуры | спецификация_функции}
  [,PRAGMA RESTRICT_REFERENCES ({имя_метода | DEFAULT},
  {RNDS | WNDS | RNPS | WNPS | TRUST}
   [,{RNDS | WNDS | RNPS | WNPS | TRUST}...] [,PRAGMA...] ...]
  [, {MAP | ORDER} MEMBER спецификация_функции]
```

Синтаксис создания VARRAY:

```
CREATE [OR REPLACE] TYPE [схема.]имя_типа
{IS | AS} {VARRAY | VARYING ARRAY} (предел) OF тип_данных
```

Синтаксис создания вложенной таблицы:

```
CREATE [OR REPLACE] TYPE [схема.]имя_типа
{IS | AS} TABLE OF тип_данных
```

Синтаксис ALTER:

```
ALTER TYPE [схема.]имя_типа
{ COMPILE [DEBUG] [SPECIFICATION | BODY] [REUSE SETTINGS]
| REPLACE [AUTHID {CURRENT_USER | DEFINER}]
  AS OBJECT (атрибут тип_данных[, атрибут тип_данных...]),
 {MEMBER | STATIC} {спецификация_процедуры | спецификация_функции}
  [,PRAGMA RESTRICT_REFERENCES ({имя_метода | DEFAULT},
  {RNDS | WNDS | RNPS | WNPS | TRUST}]
   [,{RNDS | WNDS | RNPS | WNPS | TRUST}...]),PRAGMA...]
 {MEMBER | STATIC} {спецификация_процедуры | спецификация_функции}
  [,PRAGMA RESTRICT_REFERENCES ({имя_метода | DEFAULT},
  {RNDS | WNDS | RNPS | WNPS | TRUST}
   [,{RNDS | WNDS | RNPS | WNPS | TRUST}...)[,PRAGMA...] ...]
| [{ADD | DROP}
{ {MAP | ORDER} MEMBER спецификация_функции]
| {MEMBER | STATIC} {спецификация_процедуры | спецификация_функции}}
  [INVALIDATE | CASCADE [[NOT] INCLUDING TABLE DATA] [FORCE]]
| [{ADD | MODIFY} ATTRIBUTE
     {атрибут [тип_данных] | (атрибут тип_данных, атрибут тип_данных...)}]
| DROP ATTRIBUTE {атрибут | (атрибут, атрибут...}
| [NOT] {INSTANTIABLE | FINAL}
```

Синтаксис DROP:

```
DROP TYPE [схема.]имя_типа [FORCE]
```

Создает, изменяет или удаляет неполный тип, объектный тип, тип VARRAY или вложенную таблицу.

Ключевые слова

имя_типа

Имя типа.

AUTHID CURRENT_USER

Указывает, что методы-функции и методы-процедуры будут выполняться с привилегиями текущего пользователя (права вызывающего).

AUTHID DEFINER

Указывает, что методы-функции и методы-процедуры будут выполняться с привилегиями создателя объекта (права определяющего).

атрибут

Имя атрибута данного типа.

тип_данных

Тип данных Oracle для атрибута. Запрещаются следующие типы данных: ROWID, LONG и LONG ROW.

MEMBER

Указывает, что функция или процедура сопоставлена объектному типу, ссылка на который присутствует в атрибуте. Такой метод имеет неявный первый параметр, для обращения к которому применяется SELF. Функция-метод является специфичной для каждого экземпляра объекта, поэтому данная функция может ссылаться на атрибуты своего экземпляра класса.

STATIC

Указывает, что функция или процедура сопоставлена объектному типу, ссылка на который присутствует в атрибуте, но неявных параметров не имеет. Статическая функция относится ко всему классу в целом, поэтому может принимать входные данные или формировать выходные, но не может ссылаться на какие бы то ни было динамические функции-методы или переменные.

спецификация_процедуры

Спецификация процедуры (подробная информация приведена в главе 9).

спецификация_функции

Спецификация функции (подробная информация приведена в главе 9).

PRAGMA RESTRICT_REFERENCES

Указывает, что доступ на чтение/запись к таблицам БД и/или переменным пакетов будет запрещен для метода-функции.

имя_метода

Определяет имя функции-метода, к которой будет применена директива PRAGMA.

DEFAULT

Означает, что директива PRAGMA должна быть применена ко всем методам в типе, для которых она не была указана.

RNDS

Указывает, что таблицы БД не запрашиваются (состояние запрета чтения БД).

WNDS

Указывает, что таблицы БД не изменяются (состояние запрета записи БД).

RNPS

Указывает, что переменные пакета не используются в ссылках (состояние запрета чтения пакета).

WNPS

Указывает, что переменные пакета не изменяются (состояние запрета записи пакета).

TRUST

Означает, что ограничения, заданные данной директивой PRAGMA, не приводятся в исполнение, но считаются выполненными (истинными).

MAP MEMBER

Определяет метод-функцию, которая возвращает положение указанного экземпляра относительно остальных экземпляров объекта.

ORDER MEMBER

Указывает, что метод-функция, которая принимает экземпляр объекта в качестве явного аргумента, сравнивает его с неявным аргументом SELF и возвращает значение –1, 0 или положительное целое число, отражающее отношение между двумя объектами. Значение 1 означает, что объект, переданный явно, является «большим» из двух. Значение –1 означает, что объект, переданный явно, является «меньшим» из двух. Значение 0 указывает, что два объекта должны рассматриваться как равные при упорядочивании и сортировке.

предел

Максимальное количество элементов, возвращаемых типом VARRAY.

COMPILE

Указывает, что спецификация и/или тело объектного типа должны быть скомпилированы. Если не указано ни одно из ключевых слов (SPECIFICATION или BODY), то по умолчанию будут скомпилированы и тело, и спецификация.

DEBUG

Означает, что будет сформирован код для отладчика PL/SQL.

REPLACE AS OBJECT

Указывает, что создается объектный тип верхнего уровня.

ADD

Указывает, что в данный тип добавляется новый атрибут или определение метода. Появилось в Oracle9*i*.

DROP

Указывает, что из данного типа удаляется атрибут или определение метода. Появилось в Oracle9*i*.

INVALIDATE

Указывает, что все зависимые объекты станут недействительными.

CASCADE

Указывает, что изменение типа распространится на зависимые типы и таблицы.

INCLUDING TABLE DATA

Указывает, что данные, хранящиеся в пользовательских столбцах, будут преобразованы.

FORCE

Означает, что тип будет удален, даже если для него существуют зависимые объекты базы данных.

MODIFY

Указывает, что в данном типе должен быть изменен один или несколько атрибутов. Появилось в Oracle9*i*.

Если использовать ключевое слово FORCE для удаления типа вместе с зависимыми объектами, то данные зависимых объектов могут стать недоступными. Поэтому мы настоятельно не рекомендуем применять данную возможность.

Общие ключевые слова и инструкции: *схема*.

CREATE/DROP TYPE BODY

Синтаксис CREATE:

```
CREATE [OR REPLACE] TYPE BODY [схема.]имя_типа
{IS | AS} {MEMBER | STATIC} {спецификация_процедуры | спецификация_функции}
[{MEMBER | STATIC} {спецификация_процедуры | спецификация_функции} ...]
[, {MAP | ORDER} MEMBER спецификация_функции]
END
```

Синтаксис DROP:

```
DROP TYPE BODY [схема.]имя_типа
```

Создает или удаляет тело типа.

Обратите внимание, что команды ALTER TYPE BODY не существует. Но если выполнить для тела типа команду DROP, то объявление соответствующего объектного типа все еще будет существовать и можно будет выдать новую команду CREATE TYPE BODY для создания нового (и, возможно, отличного от предыдущего) тела типа.

Ключевые слова

имя_типа

Имя объектного типа, которому соответствует данное тело типа.

MEMBER

Указывает, что функция или процедура сопоставлена объектному типу, ссылка на который присутствует в атрибуте. Этот класс метода имеет неявный первый параметр, для обращения к которому применяется SELF.

STATIC

Указывает, что функция или процедура сопоставлена объектному типу, ссылка на который присутствует в атрибуте, но неявных параметров не имеет.

спецификация_процедуры

Спецификация процедуры (подробная информация приведена в главе 9).

спецификация_функции

Спецификация функции (подробная информация приведена в главе 9).

MAP MEMBER спецификация_функции

Определяет метод-функцию, которая возвращает положение указанного экземпляра относительно остальных экземпляров объекта.

ORDER MEMBER спецификация_функции

Указывает, что метод-функция, которая принимает экземпляр объекта в качестве явного аргумента, сравнивает его с неявным аргументом SELF и возвращает значение –1, 0 или положительное целое число.

Общие ключевые слова и инструкции: *схема*.

CREATE/ALTER/DROP USER

Синтаксис CREATE:

```
CREATE USER имя_пользователя
[IDENTIFIED {BY пароль | EXTERNALLY | GLOBALLY AS 'внешнее_имя'}]
[DEFAULT TABLESPACE табличное_пространство]
[TEMPORARY TABLESPACE табличное_пространство]
[QUOTA [целое [K | M] | UNLIMITED] ON табличное_пространство]
 [QUOTA [целое [K | M] | UNLIMITED] ON табличное_пространство]...]
[PROFILE имя_профиля]
[PASSWORD EXPIRE] [ACCOUNT {LOCK | UNLOCK}]
```

Синтаксис ALTER:

```
ALTER USER имя_пользователя
[REPLACE старый_пароль]
[IDENTIFIED {BY пароль | EXTERNALLY | GLOBALLY AS 'внешнее_имя'}]
[DEFAULT TABLESPACE табличное_пространство]
[TEMPORARY TABLESPACE табличное_пространство]
[QUOTA [целое [K | M] | UNLIMITED] ON табличное_пространство]
 [QUOTA [целое [K | M] | UNLIMITED] ON табличное_пространство]...]
[PROFILE имя_профиля]
```

```
[PASSWORD EXPIRE] [ACCOUNT {LOCK | UNLOCK}]
[DEFAULT ROLE
 {имя_роли[,имя_роли ...]
 | ALL [EXCEPT имя_роли[,имя_роли ...]]
 |      NONE
 }]
[имя_пользователя [,имя_пользователя ...] инструкция_прокси]
```

Синтаксис DROP:

```
DROP USER имя_пользователя [CASCADE]
```

Создает, изменяет или удаляет пользователя базы данных вместе с соответствующими параметрами безопасности и атрибутами хранения.

Ключевые слова

IDENTIFIED BY

Указывает, что будет производиться аутентификация пользователя. Существует три разновидности аутентификации:

пароль

Идентификация при помощи локально хранимого пароля. Пароль может содержать только однобайтовые символы из набора символов БД.

EXTERNALLY

Идентифицируется внешней службой, например операционной системой. Если вы хотите, чтобы пользователь получал возможность доступа только при помощи учетной записи ОС, добавьте перед именем пользователя значение параметра OS_AUTHENT_PREFIX.

GLOBALLY AS 'внешнее_имя'

Идентификация службой каталогов предприятия. Значением параметра *внешнее_имя* может быть отличительное имя из каталога или же NULL, т. е. каталог сопоставит пользователей соответствующей схеме БД.

DEFAULT TABLESPACE

Указывает имя табличного пространства, устанавливаемое по умолчанию, когда данный пользователь создает объект базы данных.

TEMPORARY TABLESPACE

Определяет имя табличного пространства, которое используется для создания временных сегментов при выполнении таких операций, как сортировки, требующих большего объема памяти, чем имеется в наличии.

PASSWORD EXPIRE

Означает, что пользователь или администратор должен будет изменить пароль пользователя, прежде чем пользователь сможет зарегистрироваться в БД.

ACCOUNT LOCK | UNLOCK

Отключает или включает доступ для данной учетной записи.

QUOTA

Определяет объем указанного табличного пространства, в котором пользователь может хранить объекты. UNLIMITED означает, что предела не существует (вплоть до объема всего табличного пространства).

PROFILE

Устанавливает профиль пользователя в соответствии с параметром *имя_профиля*, в результате чего к пользователю будут применяться ограничения, заданные в указанном профиле.

CASCADE

Означает, что прежде чем будет удален пользователь, будут удалены все объекты в схеме пользователя. Данное ключевое слово необходимо указать, если схема пользователя содержит объекты, в противном случае команда не выполнится. Если указано ключевое слово CASCADE, то ограничения ссылочной целостности для таблиц других схем, которые ссылаются на первичные или уникальные ключи таблиц данной схемы, также будут удалены. Если на таблицы или другие объекты БД данной схемы ссылаются представления, синонимы, хранимые процедуры, функции или пакеты другой схемы, такие ссылающиеся объекты будут помечены как INVALID, но не удалены.

Общие ключевые слова и инструкции: *целое*, *табличное_пространство*.

CREATE/ALTER/DROP VIEW

Синтаксис CREATE:

```
CREATE [OR REPLACE] [[NO] FORCE] VIEW [схема.]имя_представления
[ { ( { псевдоним [Инструкция_ограничения_столбца [Инструкция_ограничения_столбца]...]
      | Инструкция_ограничения_таблицы } [,[Инструкция_ограничения_таблицы]... ]
      } )
  | OF [схема.] имя_типа
    { WITH OBJECT IDENTIFIER { DEFAULT | (атрибут [,атрибут]...)}
    | UNDER [схема.] суперпредставление
    }
    ( { Инструкция_ограничения_таблицы
      | атрибут Инструкция_ограничения_таблицы [Инструкция_ограничения_таблицы...]
      })
  | OF XMLTYPE

    [[XMLSCHEMA URL_XML-схемы] ELEMENT {element
  | URL_XML-схемы # элемент
    WITH OBJECT IDENTIFIER {(выражение[,выражение...]) | DEFAULT}

  }]
AS запрос_представления [WITH
 {READ ONLY
 | CHECK OPTION [CONSTRAINT имя_ограничения]
 }]
```

Синтаксис ALTER:

```
ALTER VIEW [схема.] имя_представления
{ADD Инструкция_ограничения_таблицы
| MODIFY CONSTRAINT Инструкция_ограничения_таблицы {RELY | NORELY}
| DROP {CONSTRAINT имя_ограничения | PRIMARY KEY | UNIQUE (столбец[,столбец]... )}
| COMPILE
}
```

Синтаксис DROP:

```
DROP VIEW [схема.]имя_представления
```

Создает, изменяет или удаляет представление.

Команда ALTER VIEW явно перекомпилирует представление. Рекомендуем выполнять ее после выполнения каких-либо изменений в базовых таблицах представления.

Ключевые слова

Инструкция_ограничения_таблицы

Инструкция может включаться в команду CREATE/ALTER VIEW начиная с версии Oracle9*i* (подробная информация приведена в разделе «Общие ключевые слова и идентификаторы»).

FORCE

Означает, что представление должно быть создано вне зависимости от существования базовых таблиц представления и наличия у владельца схемы привилегий на работу с ними.

NOFORCE

Означает, что представление должно быть создано, только если его базовые таблицы существуют и владелец схемы имеет привилегии на работу с ними. Это поведение по умолчанию.

имя_представления

Имя создаваемого представления.

псевдоним

Один или несколько псевдонимов, соответствующих столбцам или выражениям, которые возвращаются запросом представления.

OF имя_типа

Означает, что данное представление соответствует объектному типу, который определяется параметром *имя_типа*.

WITH OBJECT IDENTIFIER

Означает, что данное представление соответствует объекту верхнего уровня (корневому). Эта инструкция позволяет указать атрибуты объектного типа, который будет выступать в качестве ключа для идентификации каждой строки объектного представления.

UNDER

Означает, что данное представление является представлением нижнего уровня, в основе которого лежит указанное суперпредставление.

OF XMLTYPE

Указывает, что данное представление является представлением XMLType, отображающим данные с типом данных XMLType из таблицы XMLSchema. Спецификация определяет схему XMLSchema, которая должна применяться для сопоставления данным XML их объектно-реляционных эквивалентов. Для того чтобы можно было создать представление XMLType, схема XMLSchema должна уже существовать.

WITH OBJECT IDENTIFIER

Для объектных таблиц (а также для таблиц XMLType, объектных представлений и представлений XMLType) имена столбцов не указываются. Поэтому сервер Oracle определяет системный столбец SYS_NC_ROWINFO$. Это имя столбца можно использовать в запросах и для создания объектных представлений при помощи ключевых слов WITH OBJECT IDENTIFIER.

запрос_представления

Любая SQL-команда SELECT.

WITH CHECK OPTION

Указывает, что результатом операций вставки и обновления, выполняемых с помощью представления, должны быть строки, которые запрос представления может выбрать.

READ ONLY

Означает, что данное представление является необновляемым.

CONSTRAINT

Задает имя для ограничения CHECK OPTION. Значение по умолчанию назначается системой в виде SYS_C*n*, где *n* – это целое число; такое имя уникально.

ADD

Означает, что в представление должно быть добавлено ограничение. Появилось в Oracle9*i*.

MODIFY

Определяет, необходимо ли при перезаписи запроса учитывать ограничение в режиме NOVALIDATE. Ключевое слово RELY позволяет активировать существующее ограничение в режиме NOVALIDATE для перезаписи запроса без принудительного наложения ограничения целостности. Ограничение находится в режиме NOVALIDATE, поэтому сервер Oracle не приводит его в действие. Значение по умолчанию равно NORELY.

COMPILE

Приводит к перекомпиляции представления.

Общие ключевые слова и инструкции: *псевдоним*, *Инструкция_ограничения_столбца*, OR REPLACE, *схема*, *Инструкция_ограничения_таблицы*.

DISASSOCIATE STATISTICS

```
DISASSOCIATE STATISTICS FROM
{ COLUMNS [схема.]имя_таблицы.столбец[, [схема.]имя_таблицы.столбец ...]
| FUNCTIONS [схема.]функция[,[схема.]функция ...]
| PACKAGES [схема.]пакет[,[схема.]пакет ...]
| INDEXES [схема.]имя_индекса[,[схема.]имя_индекса ...]
}
```

Разрывает связку вычисления статистики с объектами БД. Появилась в Oracle8*i*.

Ключевые слова

COLUMNS

Означает, что будет предложен список из одного или более столбцов.

столбец

Имя столбца, для которого определена связка с объектом БД.

FUNCTIONS

Указывает, что связка будет разорвана для одной или нескольких функций.

PACKAGES

Указывает, что связка будет разорвана для одного или нескольких пакетов.

INDEXES

Указывает, что связка будет разорвана для одного или нескольких индексов.

Общие ключевые слова и инструкции: *столбец*, *Инструкция_ограничения_столбца*, *схема*, *имя_таблицы*.

GRANT (объектные привилегии)

```
GRANT {объектная_привилегия[,объектная_привилегия ...] | ALL [PRIVILEGES]}
  [столбец [,столбец...]] ON
{ [схема.]имя_объекта
| DIRECTORY имя_каталога
| JAVA {SOURCE | RESOURCE} [схема.]объект_java
}
TO {имя_пользователя | роль | PUBLIC}
[WITH GRANT OPTION]
```

Выдает привилегии на доступ к объекту базы данных для одного или нескольких пользователей или ролей.

Ключевые слова

объектная_привилегия

Определяет имя выдаваемой объектной привилегии. Возможны следующие варианты: ALTER, DELETE, EXECUTE, INDEX, INSERT, REFERENCES, SELECT и UPDATE.

имя_объекта

Имя объекта, на доступ к которому выдаются привилегии.

DIRECTORY

Имя объекта каталога, на доступ к которому выдаются привилегии.

JAVA SOURCE

Имя исходного объекта Java, на доступ к которому выдаются привилегии.

JAVA RESOURCE

Имя Java-ресурса, на доступ к которому выдаются привилегии.

имя_пользователя

Имя пользователя, которому будет выдана привилегия на доступ к объекту.

роль

Имя роли, которой будет выдана привилегия на доступ к объекту.

PUBLIC

Означает, что привилегия на доступ к объекту выдается всем текущим и будущим пользователям.

WITH GRANT OPTION

Означает, что получивший привилегию может выдавать ее другим.

Общие ключевые слова и инструкции: *столбец*, *схема*.

GRANT (системные привилегии или роли)

```
GRANT {{привилегия | роль}[,{привилегия | роль} ...] | ALL PRIVILEGES}}
TO {имя_пользователя | имя_роли | PUBLIC}
```

```
 [,{имя_пользователя | имя_роли | PUBLIC} ...]
[WITH ADMIN OPTION]
```

Выдает системную привилегию или роль одному или нескольким пользователям или ролям.

Ключевые слова

привилегия

Имя выдаваемой системной привилегии (за подробной информацией обратитесь к главе 4).

роль

Имя выдаваемой роли (за подробной информацией обратитесь к главе 4).

имя_пользователя

Имя пользователя, которому будет выдана привилегия или роль.

имя_роли

Имя роли, которой будет выдана привилегия или роль.

PUBLIC

Указывает, что привилегия или роль выдается всем пользователям, включая еще не созданных.

WITH ADMIN OPTION

Означает, что получивший привилегию или роль может выдавать ее другим, а также может изменять и удалять роль.

NOAUDIT (объекты схемы)

```
NOAUDIT параметр_объекта[,параметр_объекта ...]
ON {[схема.]имя_объекта | DIRECTORY имя_каталога | DEFAULT}
[WHENEVER [NOT] SUCCESSFUL]
```

Останавливает аудит, который был включен командой AUDIT для объектов схемы.

Ключевые слова

параметр_объекта

Указывает, что будет остановлен аудит для определенной операции. Возможны следующие варианты: ALTER, AUDIT, COMMENT, DELETE, EXECUTE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT и UPDATE. Ключевое слово ALL эквивалентно указанию всех операций.

имя_объекта

Имя объекта схемы, аудит которого будет остановлен.

DIRECTORY

Имя каталога, аудит которого будет остановлен.

DEFAULT

Указывает, что по умолчанию аудит не будет выполняться для объектов, которые еще не были созданы для указанного *параметра_объекта*.

WHENEVER SUCCESSFUL

Выключает аудит только для успешно завершившихся команд SQL.

WHENEVER NOT SUCCESSFUL

Выключает аудит только для тех команд SQL, которые не удалось выполнить или в результате выполнения которых возникли ошибки.

Общие ключевые слова и инструкции: *схема*.

NOAUDIT (команды SQL)

```
NOAUDIT {параметр_команды | системная_привилегия}[,
        {параметр_команды | системная_привилегия} ...] [BY
{ имя_пользователя[,имя_пользователя ...]
| прокси_сервер [ON BEHALF OF
  {ANY
  | имя_пользователя [,имя_пользователя ...]
  }]
}]
[WHENEVER [NOT] SUCCESSFUL]
```

Останавливает аудит, который был включен командой AUDIT для команды SQL.

Ключевые слова

параметр_команды

Указывает, что будет остановлен аудит для определенной команды (за подробной информацией обратитесь к главе 4).

системная_привилегия

Системная привилегия, для которой останавливается аудит (за подробной информацией обратитесь к главе 4).

BY имя_пользователя

Останавливает аудит только для команд SQL, выданных пользователем, указанным в параметре *имя_пользователя*. По умолчанию аудит останавливается для всех пользователей.

BY прокси_сервер

Останавливает аудит только для команд SQL, выданных указанным прокси-сервером от имени пользователя или списка конкретных пользователей.

WHENEVER SUCCESSFUL

Выключает аудит только для успешно завершившихся команд SQL. Если указано ключевое слово NOT, аудит останавливается только для тех команд SQL, которые приводят к ошибке. Если данная инструкция отсутствует, аудит останавливается для всех команд SQL.

RENAME

```
RENAME старое_имя TO новое_имя
```

Изменяет имя существующей таблицы, представления, последовательности или частного синонима. Ограничения целостности, индексы и права на старый объект автоматически передаются новому.

Объекты, которые зависят от переименованного объекта (например, представления, синонимы, хранимые процедуры или функции) будут помечены как INVALID.

Ключевые слова

старое_имя Имя существующего объекта, которому вы хотите присвоить новое имя.

новое_имя Новое имя объекта БД.

REVOKE (привилегии доступа к объектам)

```
REVOKE
{объектная_привилегия [,объектная_привилегия ...]
| ALL [PRIVILEGES]
}
ON [схема.]имя_объекта
FROM {имя_пользователя | роль | PUBLIC}
```

Отзывает выданные привилегии на доступ к объекту БД у одного или нескольких пользователей или ролей.

Ключевые слова

объектн_прив

Имя отзываемой привилегии на доступ к объекту. Возможны следующие варианты: ALTER, DELETE, EXECUTE, INDEX, INSERT, REFERENCES, SELECT и UPDATE.

имя_объекта

Имя объекта, привилегии на доступ к которому отзываются.

имя_пользователя

Имя пользователя, у которого будет отозвана привилегия на доступ к объекту.

роль

Имя роли, у которой будет отозвана привилегия на доступ к объекту.

PUBLIC

Указывает, что выданная привилегия на доступ к объекту более не будет по умолчанию доступна всем пользователям.

Общие ключевые слова и инструкции: *схема*.

REVOKE (системные привилегии или роли)

```
REVOKE {{привилегия | роль}[,{привилегия | роль} ...] | ALL PRIVILEGES
FROM {имя_пользователя | имя_роли | PUBLIC}
 [,{имя_пользователя | имя_роли | PUBLIC} ...]
```

Отзывает системную привилегию или роль у одного или нескольких пользователей или ролей.

Ключевые слова

привилегия

Имя отзываемой системной привилегии (за подробной информацией обратитесь к главе 4).

роль

Имя отзываемой роли (за подробной информацией обратитесь к главе 4).

имя_пользователя

Имя пользователя, у которого будет отозвана системная привилегия или роль.

имя_роли

Имя роли, у которой будет отозвана системная привилегия или роль.

PUBLIC

Указывает, что выданная привилегия или роль более не будет по умолчанию доступна всем пользователям.

Команды языка манипулирования данными

Команды языка манипулирования данными (Data Manipulation Language – DML) обращаются к данным, хранящимся в БД Oracle, и манипулируют ими. Эти команды могут применяться для вставки, обновления, удаления и чтения данных, а также для изменения способа работы сервера Oracle при обращении к данным БД. В DML меньше команд, чем в DDL (Data Definition Language), но многие из них имеют разнообразнейшие параметры и по несколько вариантов синтаксиса, о чем мы и поговорим в данном разделе.

ALTER SESSION

```
ALTER SESSION
{ { SET
  | [CONSTRAINT[S] = IMMEDIATE | DEFERRED | DEFAULT]
  | [CREATE_STORED_OUTLINES = TRUE | FALSE | имя_категории]
  | [CURRENT_SCHEMA = схема]
  | [CURSOR_SHARING = FORCE | EXACT]
  | [DB_BLOCK_CHECKING = TRUE | FALSE]
  | [DB_FILE_MULTIBLOCK_READ_COUNT = целое]
  | [FAST_START_IO_TARGET = целое]
  | [FLAGGER = ENTRY | INTERMEDIATE | FULL | OFF]
  | [GLOBAL_NAMES = [TRUE | FALSE]
  | [HASH_AREA_SIZE = целое]
  | [HASH_JOIN_ENABLED = TRUE | FALSE]
  | [HASH_MULTIBLOCK_IO_COUNT = целое]
  | [INSTANCE = целое]
  | [ISOLATION_LEVEL = SERIALIZABLE | READ COMMITTED]
  | [LABEL = 'текст' | DBHIGH | DBLOW | OSLABEL]
  | [LOG_ARCHIVE_DEST_целое =
      {''
      | LOCATION = путь
      | SERVICE = служба_tnsnames
      }
      [MANDATORY | OPTIONAL] [REOPEN[ = целое]]]
  | [LOG_ARCHIVE_DEST_STATE_целое = ENABLE | DEFER]
  | [LOG_ARCHIVE_MINIMUM_SUCCEED_DEST = целое]
  | [MAX_DUMP_FILE_SIZE = целое | UNLIMITED]
  | [NLS_CALENDAR = 'текст']
  | [NLS_COMP = 'текст']
  | [NLS_CURRENCY = 'текст']
  | [NLS_DATE_FORMAT = 'формат_даты']
  | [NLS_DATE_LANGUAGE = язык]
```

```
  | [NLS_DUAL_CURRENCY = 'текст']
  | [NLS_ISO_CURRENCY = территория]
  | [NLS_LABEL_FORMAT = формат_метки]
  | [NLS_NUMERIC_CHARACTERS = 'текст']
  | [NLS_LANGUAGE = язык]
  | [NLS_SORT = сортировка | BINARY]
  | [NLS_TERRITORY = территория]
  | [OBJECT_CACHE_MAX_SIZE_PERCENT = целое]
  | [{OPTIMIZER_GOAL | OPTIMIZER_MODE} =
 ALL_ROWS | FIRST_ROWS | RULE | CHOOSE]
  | [OBJECT_CACHE_OPTIMAL_SIZE = целое]
  | [OPTIMIZER_INDEX_CACHING = целое]
  | [OPTIMIZER_INDEX_COST_ADJ = целое]
  | [OPTIMIZER_MAX_PERMUTATIONS = целое]
  | [OPTIMIZER_PERCENT_PARALLEL = целое]
  | [PARALLEL_BROADCAST_ENABLED = TRUE | FALSE]
  | [PARALLEL_INSTANCE_GROUP = 'текст']
  | [PARTITION_VIEW_ENABLED = TRUE | FALSE]
  | [PLSQL_V2_COMPATIBILITY = TRUE | FALSE]
  | [QUERY_REWRITE_ENABLED = TRUE | FALSE]
  | [QUERY_REWRITE_INTEGRITY = ENFORCED | TRUSTED | STALE_TOLERATED]
  | [REMOTE_DEPENDENCIES_MODE = TIMESTAMP | SIGNATURE]
  | [SESSION_CACHED_CURSORS = целое]
  | [SKIP_UNUSABLE_INDEXES = TRUE | FALSE]
  | [SORT_AREA_RETAINED_SIZE = целое]
  | [SORT_AREA_SIZE = целое]
  | [SORT_MULTIBLOCK_READ_COUNT = целое]
  | [SQL_TRACE = TRUE | FALSE]
  | [STAR_TRANSFORMATION_ENABLED = TRUE | FALSE]
  | [TIMED_STATISTICS = TRUE | FALSE]
  | [USE_STORED_OUTLINES = TRUE | FALSE | 'имя_категории']
  }
| [CLOSE DATABASE LINK связь_БД]
| [ADVISE COMMIT | ROLLBACK | NOTHING]
| [{ENABLE | DISABLE} COMMIT IN PROCEDURE]
| [{ENABLE | DISABLE | FORCE} PARALLEL {DML | DDL} [PARALLEL целое]]
| {[ENABLE RESUMABLE [TIMEOUT целое][NAME строка]
  | DISABLE RESUMABLE]
  }
}
```

Изменяет функциональные характеристики текущего сеанса связи с БД, в том числе характеристики поддержки национальных языков (National Language Support – NLS).

Многие параметры, которые могут быть заданы при помощи этой команды, определяются на уровне экземпляра БД параметрами файла инициализации (*INIT.ORA* или *SPFILE*). Несмотря на то что здесь могут быть указаны значения по умолчанию, значения параметров инициализации подменяют такие умолчания и становятся значениями по умолчанию де-факто. Прежде чем задавать какие-либо параметры для сеанса, убедитесь, что правильно понимаете, как они работают.

Ключевые слова

CONSTRAINT[S]

Определяет, когда будут приведены в исполнение условия, определенные в отложенном ограничении.

IMMEDIATE

Условия проверяются непосредственно после каждой команды DML.

DEFERRED

Означает, что условия проверяются после фиксации транзакции.

DEFAULT

Восстанавливает исходное состояние для всех ограничений (как они были определены при создании).

CREATE_STORED_OUTLINES

Определяет, будет ли сервер Oracle хранить шаблон плана выполнения для каждого запроса. Если указано *имя_категории*, то шаблон плана выполнения будет создаваться и храниться в категории *имя_категории*.

CURRENT_SCHEMA

Изменяет текущую схему на указанную. Несмотря на то что схема изменяется, пользователь не меняется, и дополнительные привилегии не появляются.

CURSOR_SHARING

Указывает, какие виды команд SQL могут совместно использовать курсор. Если указано ключевое слово FORCE, команды могут совместно использовать курсор, если они идентичны с точностью до нескольких литералов и если отличия не оказывают влияния на смысл команды. EXACT означает, что только полностью идентичные команды SQL могут совместно использовать курсор.

DB_BLOCK_CHECKING

Указывает, будет ли производиться проверка блоков данных.

DB_FILE_MULTIBLOCK_READ_COUNT

Определяет количество считываний блоков в рамках одной операции ввода-вывода при выполнении последовательного просмотра. Значение по умолчанию равно 8.

FAST_START_IO_TARGET

Указывает целевое количество операций чтения и записи из кэша и в него, которое должно быть выполнено при отказе системы или восстановлении экземпляра.

FLAGGER

Определяет уровень маркировки по федеральному стандарту обработки информации (Federal Information Processing Standard – FIPS), определяющий, какое сообщение об ошибке генерируется в случае, если команда SQL не соответствует стандартам ANSI SQL-92. Имейте в виду, что в настоящее время не существует отличий между вариантами ENTRY, INTERMEDIATE и FULL.

GLOBAL_NAMES

Определяет, будет ли для сеанса использоваться глобальное разрешение имен.

HASH_AREA_SIZE

Указывает объем памяти (в байтах), резервируемый для соединений хешированием.

HASH_JOIN_ENABLED

Определяет, будут ли в запросах выполняться соединения хешированием.

HASH_MULTIBLOCK_IO_COUNT

Задает количество блоков данных, считываемых или записываемых при соединении хешированием.

INSTANCE

Указывает, что в среде Parallel Server/Real Application Clusters доступ к файлам БД осуществляется так, как если бы сеансы были подключены к определенному экземпляру.

ISOLATION_LEVEL

Определяет, как будут обрабатываться изменения БД.

SERIALIZABLE

Если производится попытка изменения строки, измененной другим сеансом, но операция изменения еще не была зафиксирована, то команда не выполнится. Данная опция соответствует описанному в стандарте SQL-92 уровню изоляции транзакций SERIALIZABLE.

READ COMMITTED

Именно так сервер Oracle ведет себя по умолчанию: если строка заблокирована другой незафиксированной транзакцией, то команда будет ждать снятия блокировки строк.

LABEL

Меняет метку DBMS-сеанса на метку, заданную параметром *текст*, метку-эквивалент DBHIGH или DBLOW или метку операционной системы (OSLABEL).

LOG_ARCHIVE_DEST_целое

Указывает местоположение архивных групп журнальных файлов. Можно задать до пяти таких местоположений (с помощью параметра *целое*), и процесс архивирования попытается заархивировать журнальные файлы по каждому из них.

"

Означает, что для данного направления местоположение архивного журнала не определено. Однако хотя бы для одного из направлений архивации журнала (1–5) местоположение должно быть определено.

LOCATION

Определяет место расположения архивных журнальных файлов в ОС.

SERVICE

Задает имя службы Oracle Net Services, обслуживающей резервную базу данных. Запись *служба_tnsnames* должна содержаться в файле TNSNAMES.ORA.

MANDATORY

Указывает, что журнальный файл становится доступен для повторного использования только после успешной архивации по указанному направлению.

OPTIONAL

Указывает, что для повторного использования журнального файла не требуется, чтобы архивация в заданном направлении была успешной.

REOPEN

Указывает период времени (в секундах), который должен пройти после ошибки архивации в заданное местоположение, прежде чем туда же будет осуществлена следующая попытка архивации.

LOG_ARCHIVE_DEST_STATE_целое

Указывает состояние, сопоставляемое соответствующему LOG_ARCHIVE_DEST_*целое*. *ENABLE* означает, что можно задать любое разрешенное значение LOG_

ARCHIVE_DEST. DEFER означает, что LOG_ARCHIVE_DEST_*целое* с совпадающим значением *целое* не будет использоваться.

LOG_ARCHIVE_MINIMUM_SUCCEED_DEST

Задает минимальное количество направлений, которые должны быть успешно записаны, прежде чем журнальный файл можно будет использовать повторно.

MAX_DUMP_FILE_SIZE

Определяет максимальный размер файла дампа трассировки в блоках. Если указано ключевое слово UNLIMITED, то размер не ограничен.

NLS_CALENDAR

Определяет новый тип календаря, например, GREGORIAN или JAPANESE IMPERIAL.

NLS_COMP

Указывает, что сравнение с учетом языка должно выполняться на основе правил, определенных в параметре NLS_SORT, который указывается в текстовом виде, например, FRENCH или AMERICAN.

NLS_CURRENCY

Задает обозначение национальной валюты, возвращаемое элементом числового формата L. Этот параметр подменяет значение по умолчанию, указанное в параметре NLS_TERRITORY.

NLS_DATE_FORMAT

Определяет формат даты по умолчанию. Значение *формат_даты* должно быть корректной маской формата даты Oracle. Этот параметр подменяет значение по умолчанию, заданное в параметре NLS_TERRITORY.

NLS_DATE_LANGUAGE

Указывает язык (например, FRENCH, JAPANESE) для названий дней и месяцев, в также для других значений дат. Этот параметр подменяет значение по умолчанию, заданное в параметре NLS_TERRITORY.

NLS_DUAL_CURRENCY

Указывает значение, которое будет возвращаться элементом числового формата U (обычно применяется для евро).

NLS_ISO_CURRENCY

Определяет территорию (например, AMERICA, FRANCE), ISO-символ валюты. Этот параметр подменяет значение по умолчанию, указанное в параметре NLS_TERRITORY.

NLS_LABEL_FORMAT

Только при работе с Trusted Oracle. Изменяет формат метки по умолчанию для сеанса.

NLS_NUMERIC_CHARACTERS

Указывает разделитель десятичной дроби и разделитель групп (например, запятая и точка). Значение '*текст*' должно иметь форму '*dg*', где *d* – это разделитель десятичной дроби, а *g* – групповой разделитель. Этот параметр подменяет значение по умолчанию, указанное в параметре NLS_TERRITORY.

NLS_LANGUAGE

Указывает язык (например, FRENCH, JAPANESE) для сообщений Oracle, названий дней и месяцев, а также для выбора механизма сортировки по умолчанию.

NLS_SORT

Задает схему упорядочения для символьных сортировок. Ключевое слово BINARY задает двоичную сортировку, а параметр *сортировка* указывает имя конкретной схемы упорядочения (например, FRENCH или GERMAN).

NLS_TERRITORY

Указывает название территории (например, FRANCE, JAPAN), для которой будут установлены по умолчанию формат даты, разделитель десятичной дроби и разделитель групп, а также местное и по ISO обозначения валюты. Этот параметр может подменять значения по умолчанию, указанные в параметре NLS_LANGUAGE.

OBJECT_CACHE_MAX_SIZE_PERCENT

Указывает размер (в процентах), на который объектный кэш может быть увеличен сверх оптимального размера. Значение по умолчанию равно 10.

OBJECT_CACHE_OPTIMAL_SIZE

Определяет оптимальный размер (в килобайтах) объектного кэша. Значение по умолчанию равно 100.

OPTIMIZER_GOAL

В Oracle7 указывает цель оптимизации сеанса. Начиная с версии Oracle8 этот параметр был заменен на OPTIMIZER_MODE.

OPTIMIZER_INDEX_CACHING

Указывает количество (в процентах) блоков индекса, которые будут кэшированы.

OPTIMIZER_INDEX_COST_ADJ

Задает величину (в процентах), указывающую, насколько важным оптимизатор считает наличие индекса (по отношению к выполнению полного сканирования таблицы).

OPTIMIZER_MAX_PERMUTATIONS

Указывает количество перестановок таблиц, которое оптимизатор будет рассматривать для больших запросов с соединениями.

OPTIMIZER_MODE

Начиная с Oracle8 задает цель оптимизации для сеанса. Возможны следующие значения параметра:

ALL_ROWS

Оптимизация по наилучшей общей производительности.

FIRST_ROWS

Оптимизация по лучшему времени ответа.

RULE

Оптимизация по синтаксису.

CHOOSE

Если возможно, применять оптимизацию по стоимости, в противном случае – оптимизацию по синтаксису.

OPTIMIZER_PERCENT_PARALLEL

Указывает, в каком объеме оптимизатор задействует параллелизм для своих функций стоимости. Значение по умолчанию равно 0, что означает отсутствие параллелизма.

PARALLEL_BROADCAST_ENABLED

Означает, что для повышения производительности при выполнении соединений хешированием и слиянием может применяться параллельная обработка.

PARALLEL_INSTANCE_GROUP

Указывает группу параллельных экземпляров, которая будет использоваться для порождения подчиненных параллельных запросов. Этот параметр действует только при работе с компонентом Parallel Server или Real Application Clusters в параллельном режиме.

PARTITION_VIEW_ENABLED

Означает, что при операциях над секционированными представлениями доступ к ненужным таблицам может не осуществляться.

PLSQL_V2_COMPATIBILITY

Определяет, будет ли разрешено применение конструкций PL/SQL, которые были законны в Oracle7 (PL/ SQL версия 2.0), но не поддерживаются начиная с Oracle8. Значение TRUE разрешает старые конструкции, значение FALSE запрещает.

QUERY_REWRITE_ENABLED

Указывает, будет ли разрешена перезапись запроса для материализованных представлений. Перезапись запроса запрещена, если значение параметра OPTIMIZER_MODE равно RULE.

QUERY_REWRITE_INTEGRITY

Указывает степень согласованности данных при перезаписи запросов.

ENFORCED

Oracle принудительно вводит и гарантирует согласованность данных.

TRUSTED

Поддерживаются материализованные представления, созданные при помощи инструкции ON PREBUILD TABLE, допускаются не введенные принудительно сервером Oracle отношения соединения.

STALE_TOLERATED

Можно выбрать любое устаревшее, но пригодное для использования материализованное представление.

REMOTE_DEPENDENCIES_MODE

Указывает, как зависимости от удаленных хранимых процедур должны обрабатываться сервером БД.

SESSION_CACHED_CURSORS

Для данного сеанса определяет количество курсоров, которые могут удерживаться в кэше.

SKIP_UNUSABLE_INDEXES

Определяет, будут ли разрешены операции над таблицами с непригодными индексами или разделами индексов. Если значение равно TRUE, то такие операции будут разрешены, а если – FALSE, то будут завершаться с ошибкой.

SORT_AREA_RETAINED_SIZE

Указывает максимальный объем памяти (в байтах), которая будет удерживаться каждой операцией сортировки после первой выборки.

SORT_AREA_SIZE

Указывает максимальный объем памяти (в байтах), резервируемый для каждой операции сортировки.

SORT_MULTIBLOCK_READ_COUNT

Определяет количество блоков, которые считываются каждый раз при выполнении сортировкой чтения временных сегментов. Значение по умолчанию равно 2.

SQL_TRACE

Определяет, будет ли формироваться статистика производительности. Начальное значение устанавливается в файле инициализации.

STAR_TRANSFORMATION_ENABLED

Указывает, будет ли оптимизация по стоимости применяться к запросам типа «звезда».

TIMED_STATISTICS

Определяет, будет (TRUE) или нет (FALSE) сервер Oracle запрашивать информацию о времени у операционной системы при формировании статистик, основанных на времени.

USE_STORED_OUTLINES

Определяет, будет ли оптимизатор при формировании планов выполнения применять хранимые шаблоны планов выполнения. Если указать *имя_категории*, то будут применены только шаблоны планов выполнения из данной категории.

CLOSE DATABASE LINK

Закрывает соединение с удаленной базой данных, использующее связь БД, заданную параметром *связь_БД*. Эта команда выполнится успешно, только если связь БД не используется и в настоящий момент никакая транзакция в канале не фиксируется.

ADVISE

Отправляет уведомление о необходимости распределенной транзакции в удаленную базу данных, помещая значение 'C' (COMMIT), 'R' (ROLLBACK) или '' (NOTHING) в параметр DBA_2PC_PENDING_ADVICE удаленной БД.

ENABLE COMMIT IN PROCEDURE

Означает, что процедуры и хранимые функции могут выдавать команды COMMIT и ROLLBACK.

DISABLE COMMIT IN PROCEDURE

Означает, что процедуры и хранимые функции не могут выдавать команды COMMIT и ROLLBACK.

ENABLE PARALLEL

Означает, что при возможности команды DML или DDL будут выполняться параллельно. Это поведение по умолчанию для команд DDL.

DISABLE PARALLEL

Означает, что команды DML или DDL не будут выполняться параллельно. Это поведение по умолчанию для команд DML.

FORCE PARALLEL

Указывает, что последующие команды сеанса будут выполняться параллельно.

DML

Означает, что ключевое слово ENABLE, DISABLE или FORCE PARALLEL применяется к командам DML.

DDL

Означает, что ключевое слово ENABLE, DISABLE или FORCE PARALLEL применяется к командам DDL.

PARALLEL

Определяет степень параллелизма. Значение *целое* подменяет соответствующую инструкцию команды DDL, но не подсказку оптимизатору последующей команды DML.

ENABLE RESUMABLE

Указывает, что для сеанса разрешено возобновляемое выделение пространства. Появилось в Oracle9*i*.

TIMEOUT

Определяет период времени (в секундах), в течение которого операция может быть приостановлена в ожидании исправления условия ошибки. Если ошибка не исправлена в течение периода TIMEOUT, то сервер Oracle прерывает приостановленную операцию.

NAME

Указывает пользовательскую текстовую строку, которая поможет пользователям идентифицировать команды, выдаваемые в рамках сеанса, который работает в возобновляемом режиме. Сервер Oracle вставляет указанную *строку* в представления словаря данных USER_RESUMABLE и DBA_RESUMABLE. Если не задать NAME, сервер Oracle вставляет строку по умолчанию: 'User *имя_ пользователя(идентификатор_пользователя),* Session *идентификатор_сеанса,* Instance *идентификатор_экземпляра*'.

DISABLE RESUMABLE

Указывает, что для сеанса запрещено возобновляемое выделение пространства. Появилось в Oracle9*i*.

Общие ключевые слова и инструкции: *целое*, *схема*.

ANALYZE

```
ANALYZE {TABLE [схема.]имя_таблицы
 [PARTITION имя_раздела | SUBPARTITION имя_подраздела]
| INDEX [схема.]имя_индекса
 [PARTITION имя_раздела | SUBPARTITION имя_подраздела]
| CLUSTER [схема.]имя_кластера}
}
{ COMPUTE [SYSTEM] STATISTICS
| ESTIMATE [SYSTEM] STATISTICS [SAMPLE целое {ROWS | PERCENT}]
  {[FOR {TABLE
  | ALL [INDEXED] COLUMNS [SIZE целое]
  | COLUMNS [SIZE целое] {имя_столбца | атрибут} [SIZE целое]
  | ALL [LOCAL] INDEXES]
  }
}
| DELETE [SYSTEM] STATISTICS
```

```
| VALIDATE STRUCTURE [CASCADE]
| LIST CHAINED ROWS [INTO [схема.]имя_таблицы]
| VALIDATE REF UPDATE [SET DANGLING TO NULL]
}
```

Собирает или удаляет статистическую информацию об объекте БД, проверяет структуру объекта или идентифицирует перемещенные и сцепленные строки таблицы или кластера.

Получить доступ к статистике можно при помощи представлений ALL_TABLES, USER_TABLES и DBA_TABLES. Некоторые столбцы статистики доступны в представлениях ALL_TAB_COLUMNS, USER_TAB_COLUMNS и DBA_TAB_COLUMNS. Статистическая информация о кластерах имеется также в представлениях USER_CLUSTERS и DBA_CLUSTERS.

Ключевые слова

COMPUTE STATISTICS

Вычисляет точные статистические данные для всего указанного объекта и сохраняет их в словаре данных.

ESTIMATE STATISTICS

Приблизительно вычисляет статистики для указанного объекта и сохраняет их в словаре данных. Можно использовать необязательную инструкцию SAMPLE для определения объема выборки, для которой вычисляется статистика. Эта инструкция содержит следующие ключевые слова:

ROWS

В качестве выборки будет выступать заданное параметром *целое* количество строк таблицы, или кластера, или записей индекса.

PERCENT

В качестве выборки будет выступать заданное параметром *целое* количество процентов строк таблицы, или кластера, или записей индекса. Для PERCENT разрешен диапазон значений от 1 до 99. Если значение SAMPLE не указано, будет взято значение по умолчанию – 1050 строк.

COMPUTE STATISTICS вычисляет более точную статистику, но делает это медленнее. ESTIMATE STATISTICS обычно выполняется гораздо быстрее, при этом точность почти не страдает. Анализируемый объект блокируется на время сбора статистики, поэтому в средах с большим количеством транзакций более быстрый способ оказывается предпочтительным.

FOR

Определяет, будет ли анализироваться вся таблица (или индекс) или лишь отдельные столбцы. Появилось в Oracle9*i*.

TABLE

Означает, что будут собираться только статистики для таблицы (но не для столбцов).

ALL COLUMNS

Означает, что статистики будут собираться для всех столбцов таблицы. Если указано ключевое слово INDEXED, то анализироваться будут только столбцы, входящие в индекс для данной таблицы.

COLUMNS

Означает, что статистики будут собираться только для определенных столбцов (но не для таблицы). Столбцы могут задаваться по имени или по атрибутам.

ALL INDEXES

Означает, что будут исследоваться все индексы, сопоставленные таблице. Если указано ключевое слово LOCAL, то будут анализироваться локальные индексы (данное ключевое слово необходимо, если применяется инструкция PARTITION).

SIZE

Указывает количество столбцов в гистограмме. Значение параметра *целое* должно лежать в диапазоне от 1 до 254, значение по умолчанию равно 75.

DELETE STATISTICS

Приводит к удалению всех статистик, хранящихся в словаре данных для указанного объекта.

VALIDATE STRUCTURE

Инициирует проверку структуры указанного объекта. Ключевое слово CASCADE означает, что будут проверены и соответствующие объекту индексы.

LIST CHAINED ROWS

Формирует список сцепленных и перемещенных строк указанной таблицы или кластера (такая операция не разрешена для индекса). Записи делаются в таблице CHAINED_ROWS, при этом предполагается, что она существует в схеме пользователя. Для того чтобы указать другую целевую таблицу, применяется инструкция INTO.

VALIDATE REF UPDATE

Инициирует проверку ссылок указанного объекта.

SET DANGLING TO NULL

Означает, что всем ссылкам, указывающим на несуществующий или недействительный объект, присваивается значение NULL.

Общие ключевые слова и инструкции: *схема, имя_таблицы*.

DELETE

```
DELETE [FROM]
{ имя_таблицы[@связь_БД]
| имя_таблицы PARTITION (имя_раздела)
| имя_таблицы SUBPARTITION (имя_подраздела)
| представление[@связь_БД]
| моментальная_копия[@связь_БД]
| [(подзапрос)]
| [TABLE(подзапрос)]
}
[псевдоним_таблицы] |
ONLY (
{ имя_таблицы[@связь_БД]
| имя_таблицы PARTITION (имя_раздела)
| имя_таблицы SUBPARTITION (имя_подраздела)
| представление[@связь_БД]
| моментальная_копия[@связь_БД]
```

```
| [(подзапрос)]
| [TABLE(подзапрос)]
}
[псевдоним_таблицы])
[WHERE условие]
    [RETURNING выражение[,выражение ...] INTO элемент_данных[,элемент_данных ...]]
```

Удаляет строки из таблицы, представления или моментальной копии.

Ключевые слова

FROM

Необязательное ключевое слово, применяемое для удобочитаемости.

PARTITION

Указывает, что строки будут удалены из раздела (имя которого задано в параметре *раздел)* указанной таблицы.

SUBPARTITION

Указывает, что строки будут удалены из подраздела (имя которого указано в параметре *подраздел)* указанной таблицы.

подзапрос

Указывает подзапрос, который определяет строки-кандидаты на удаление (подробная информация о подзапросах приведена в разделе, посвященном команде SELECT).

TABLE подзапрос

Указывает команду SELECT, возвращающую значения единственного столбца, который должен быть вложенной таблицей. Ключевое слово TABLE информирует сервер Oracle о том, что значение представляет собой коллекцию, а не скаляр.

псевдоним_таблицы

Задает псевдоним для таблицы, представления или подзапроса. Если указан *псевдоним_таблицы*, то все столбцы, упоминаемые в команде DELETE со ссылкой на определенную таблицу, должны указываться с употреблением не имени таблицы, а ее псевдонима.

ONLY

Данная инструкция относится только к представлениям и применяется, если представление в инструкции FROM принадлежит к иерархии представлений, а вы не хотите удалять строки из каких бы то ни было его подпредставлений.

WHERE

Определяет *условие*, которое будет применяться для идентификации строк, подлежащих удалению. В качестве *условия* может выступать любое разрешенное условие WHERE.

RETURNING

Указывает, что для строк, удаляемых данной командой, возвращается значение (значения) указанного выражения (выражений). Действует только в рамках программы PL/SQL.

INTO

Определяет переменные PL/SQL, в которые сохраняются значения, возвращаемые строками, подлежащие удалению данной командой.

Общие ключевые слова и инструкции: *связь_БД*, *имя_таблицы*.

EXPLAIN PLAN

```
EXPLAIN PLAN
SET STATEMENT_ID = 'текст'
[INTO [схема.]имя_таблицы[@связь_БД]]
FOR SQL_команда
```

Создает пояснение для плана исполнения команды SQL.

Для выдачи команды пользователь должен обладать привилегий INSERT для таблицы назначения (имя которой указывается после INTO). Таблица назначения обычно называется PLAN_TABLE, ее можно создать при помощи сценария *utlxplan.sql*. Значение, указанное в инструкции SET, появляется в столбце STATEMENT_ID таблицы назначения.

Ключевые слова

SET STATEMENT_ID — Задает текстовую строку, используемую для идентификации результата данной команды EXPLAIN PLAN. Значение по умолчанию равно NULL.

INTO — Задает имя и местоположение таблицы плана. По умолчанию это таблица PLAN_ TABLE в текущей схеме.

FOR — Определяет, для какой SQL-команды формируется план исполнения.

Общие ключевые слова и инструкции: *связь_БД*, *схема*, *имя_таблицы*.

INSERT

```
INSERT INTO
{ имя_таблицы[@связь_БД]
| имя_таблицы PARTITION (имя_раздела)
| имя_таблицы SUBPARTITION (имя_подраздела)
| представление[@связь_БД]
}
[псевдоним_таблицы]
[(столбец[,столбец...])]
{ VALUES (выражение
  | DEFAULT[,выражение
  | DEFAULT...]
  )
| подзапрос
}
[RETURNING выражение[,выражение...] INTO элемент_данных[,элемент_данных...]]
```

Вставляет строку данных в таблицу или представление.

Ключевые слова

PARTITION

Указывает, что строки будут вставлены в раздел (имя которого задано в параметре *раздел*) указанной таблицы.

SUBPARTITION

Указывает, что строки будут вставлены в подраздел (имя которого указано в параметре *подраздел*) указанной таблицы.

представление

Имя представления, в которое будут вставлены строки.

псевдоним_таблицы

Псевдоним таблицы или представления. Если *псевдоним_таблицы* указан, то все столбцы, упоминающиеся в команде INSERT со ссылкой на определенную таблицу, должны задаваться с указанием не имени таблицы, а ее псевдонима.

столбец

Имя одного или нескольких столбцов таблицы или представления, в которых будут сохранены значения. Если указано ключевое слово VALUES, то для каждого из перечисленных столбцов в инструкции VALUES должно быть указано соответствующее выражение. Если список имен столбцов пропущен, то считается, что он содержит список всех столбцов таблицы или представления.

VALUES

Указывает значения, которые будут сохранены в каждом столбце вставляемой строки. Значением *выражения* может быть любое допустимое выражение SQL, при этом выражений должно быть ровно столько, сколько указано столбцов для таблицы или представления. Если список столбцов не приводится, то количество выражений должно совпадать с общим количеством столбцов таблицы или представления.

DEFAULT

Означает, что сервер Oracle должен использовать для столбца значение по умолчанию. Если оно не объявлено, то принимается равным NULL. Появилось в Oracle9*i*.

подзапрос

Указывает подзапрос, возвращающий значения, которые будут храниться во вставляемой строке. Если для таблицы или представления, в которые вставляются строки, указан список столбцов, то *подзапрос* должен возвращать ровно такое же количество столбцов и в той же последовательности. Если список столбцов не указан, то *подзапрос* должен возвращать столько столбцов, сколько содержится в таблице или представлении (подробная информация о подзапросах приведена в разделе, посвященном команде SELECT).

RETURNING

Указывает, что для строк, вставляемых данной командой, возвращаются значения указанных выражений. Действует только в рамках программы PL/SQL.

INTO

Указывает, что значения, возвращаемые для строк, вставленных данной командой, будет сохранены в переменных PL/SQL, указанных в параметре *элемент_данных*.

Общие ключевые слова и инструкции: *связь_БД*, *выражение*, *имя_раздела*, *имя_подраздела*, *имя_таблицы*.

MERGE

```
MERGE INTO имя_таблицы [псевдоним]
  USING {имя_таблицы | имя_представления | подзапрос}
ON (условие)
```

```
WHEN MATCHED THEN инструкция_обновления_merge
WHEN NOT MATCHED THEN инструкция_вставки_merge
```

Выбирает строки таблицы для обновления или вставки в другую таблицу. Данная команда позволяет избежать применения нескольких команд INSERT и UPDATE. Появилась в Oracle9*i* Release 2.

Ключевые слова

INTO

Задает имя целевой таблицы, т. е. таблицы, которая будет обновлена или в которую будут вставлены строки.

USING

Указывает имя таблицы, представления или подзапроса, из которого будут выбираться строки.

ON (условие)

Задает условие, которое будет вычислено как TRUE или FALSE. Это инструкция SQL, обычно содержащая проверку на равенство.

инструкция_обновления_merge

Указывает SQL DML-команду, которая будет выполнена в случае, если *условие* равно TRUE. Обычно это команда UPDATE.

инструкция_вставки_merge

Указывает SQL DML-команду, которая будет выполнена в случае, если *условие* не равно TRUE. Обычно это команда INSERT.

Общие ключевые слова и инструкции: *псевдоним, имя_таблицы*.

SAVEPOINT

```
SAVEPOINT точка_сохранения
```

Определяет точку выполнения транзакции, до которой можно будет выполнить откат при помощи команды ROLLBACK.

Ключевое слово

SAVEPOINT точка_сохранения

Указывает имя для создаваемой точки сохранения.

SELECT

```
SELECT [{DISTINCT | UNIQUE | ALL}]
{ [схема.]{имя_таблицы. | представление. | моментальная_копия.}
|  выражение [[AS] псевдоним][,выражение [[AS] псевдоним] ...]
| *
}
FROM
{имя_таблицы[@связь_БД] [AS OF {SCN | TIMESTAMP} выражение]
| имя_таблицы PARTITION (имя_раздела) [AS OF {SCN | TIMESTAMP} выражение]
| имя_таблицы SUBPARTITION (имя_подраздела) [AS OF {SCN | TIMESTAMP} выражение]
| имя_таблицы SAMPLE [BLOCK] процент_выборки [AS OF {SCN | TIMESTAMP} выражение]
| представление[@связь_БД] [AS OF {SCN | TIMESTAMP} выражение]
```

```
| моментальная_копия[@связь_БД] [AS OF {SCN | TIMESTAMP} выражение]
| (подзапрос)
| имя_таблицы
  { [тип_соединения] JOIN имя_таблицы
    { ON условие
    | USING (столбец [, столбец]... )
  | { CROSS JOIN | NATURAL [тип_соединения] JOIN имя_таблицы }
  }
[,[псевдоним_таблицы]
[,имя_таблицы[@связь_БД] [AS OF {SCN | TIMESTAMP} выражение]
| имя_таблицы PARTITION (имя_раздела) [AS OF {SCN | TIMESTAMP} выражение]
| имя_таблицы SUBPARTITION (имя_подраздела) [AS OF {SCN | TIMESTAMP} выражение]
| имя_таблицы SAMPLE [BLOCK] процент_выборки [AS OF {SCN | TIMESTAMP} выражение]
| представление[@связь_БД] [AS OF {SCN | TIMESTAMP} выражение]
| моментальная_копия[@связь_БД] [AS OF {SCN | TIMESTAMP} выражение]
| (подзапрос)
| имя_таблицы
  { [тип_соединения] JOIN имя_таблицы
    { ON условие
    | USING (столбец [, столбец]... )}
  | { CROSS JOIN | NATURAL [тип_соединения] JOIN имя_таблицы }
  }
}
[псевдоним_таблицы]]
[WHERE условие]
{[GROUP BY {выражение | {выражение[,выражение ...]}
| CUBE (выражение[,выражение ...])
| ROLLUP (выражение[,выражение ...])
}
GROUPING SETS (
  {выражение | {выражение[,выражение ...]}
  | CUBE (выражение[,выражение ...])
  | ROLLUP (выражение[,выражение ...])
  }
[HAVING условие]
[[START WITH условие] CONNECT BY условие]
[{UNION [ALL] | INTERSECT | MINUS} {подзапрос)
 [,{UNION [ALL] | INTERSECT | MINUS} {подзапрос) ...]]
[ORDER BY {выражение | позиция| псевдоним} [ASC | DESC]
 [, {выражение | позиция | псевдоним} [ASC | DESC] ...]
[FOR UPDATE OF {таблица | представление}.столбец
 [{ имя_таблицы | представление}.столбец ...]]
[NOWAIT]
```

Извлекает данные из таблиц, представлений или моментальных копий.

Ключевые слова

DISTINCT

Указывает, что должна быть возвращена лишь одна копия строки, даже в случае наличия дубликатов строк. Строка-дубликат – это строка, возвращающая все те же значения для всех столбцов из списка выборки.

представление

Имя представления.

Моментальная_копия

Имя моментальной копии.

*

Означает, что должны быть возвращены все столбцы. Является эквивалентом перечисления всех столбцов таблицы, представления или моментальной копии.

AS

Приводит псевдоним для столбца или выражения. Ключевое слово AS может отсутствовать.

FROM

Указывает имя (имена) одной или нескольких таблиц, представлений или моментальных копий, из которых будут извлекаться данные.

ALL

Указывает, что должны быть возвращены все строки, включая дубликаты. Это поведение по умолчанию.

AS OF

Указывает, что запрос должен работать с данными по состоянию на конкретный системный номер изменения (SCN) или временную метку, как указывает *выражение*. Данная инструкция не может применяться при работе со связями БД, она появилась в Oracle9*i* Release 2.

PARTITION

Указывает, что строки будут извлечены из раздела (имя которого задает параметр *раздел)* указанной таблицы.

SUBPARTITION

Указывает, что строки будут извлечены из подраздела (имя которого задает параметр *подраздел)* указанной таблицы.

SAMPLE [BLOCK]

Означает, что будет извлечена случайная выборка строк таблицы. Замените параметр *процент_выборки* на необходимое процентное отношение. Если указано ключевое слово BLOCK, то сервер Oracle выполняет блочную, а не строковую выборку.

подзапрос

Любая разрешенная команда SELECT. Имейте в виду, что подзапрос не может содержать инструкцию FOR UPDATE.

псевдоним_таблицы

Указывает псевдоним таблицы, представления или моментальной копии. Если *псевдоним_таблицы* указан, то все неоднозначно определенные столбцы, упоминаемые в команде SELECT, должны задаваться с указанием не имени таблицы, а ее псевдонима.

JOIN

Явно указывает, что выполняется соединение. Такая конструкция может заменить старую синтаксическую конструкцию для соединений Oracle: выражения для таблиц, разделенные запятыми. Появилось в Oracle9*i*.

тип_соединения

Определяет, какой тип соединения будет выполнен. Появилось в Oracle9*i*.

INNER

Выполняется внутреннее соединение. Этот тип соединения устанавливается по умолчанию.

RIGHT

Выполняется правое внешнее соединение.

LEFT

Выполняется левое внешнее соединение.

FULL

Выполняется полное или двустороннее внешнее соединение. В дополнение к внутреннему соединению строки обеих таблиц, которые не были возвращены в результате внутреннего соединения, будут сохранены и дополнены значениями NULL.

***ON** условие*

Задает условие соединения отдельно от каких бы то ни было условий поиска или фильтрации инструкции WHERE. Появилось в Oracle9*i*.

USING

Указывает, какие столбцы должны использоваться при определении соединения столбцов по равенству (*equijoin*), имеющих одинаковые имена в обеих таблицах. Данная инструкция может применяться, только если столбцы соединения имеют одинаковые имена в двух таблицах.

CROSS JOIN

Указывает, что будет выполнено перекрестное соединение. Перекрестное соединение формирует прямое произведение двух отношений.

NATURAL

Означает, что будет выполнено естественное соединение. В основе естественного соединения лежат все одноименные столбцы двух таблиц. Оно выбирает строки обеих таблиц, которые имеют одинаковые значения в соответствующих столбцах. Появилось в Oracle9*i*.

WHERE

Указывает, что будут извлечены только строки, удовлетворяющие *условию*. Вычисляется *условие*, и возвращаются только те строки, для которых оно равно TRUE. Если инструкция не указана, то будут возвращены все строки.

GROUP BY

Означает, что строки будут сгруппированы согласно предложенным выражениям и для каждой группы будет возвращена только одна строка сводной информации. Начиная с версии Oracle9*i* поддерживаются множественные группировки.

CUBE

Означает, что строки должны быть сгруппированы на основе всех возможных комбинаций значений предложенного списка выражений. Появилось в Oracle8*i*.

ROLLUP

Означает, что строки должны быть сгруппированы на основе значений предложенного списка выражений и сводных строк, возвращенных для каждого *выражения*, наряду с дополнительной строкой общего итога. Появилось в Oracle8*i*.

GROUPING SETS

Задает несколько групп данных для более удобного агрегирования. Если вы указываете только нужные группы, то серверу Oracle не приходится выполнять все множество агрегирований, которых требуют CUBE и ROLLUP. Появилось в Oracle9*i*.

START WITH

Указывает строки, выступающие в качестве корневых в иерархическом запросе. Если эти ключевые слова пропущены, то все строки таблицы считаются корневыми.

CONNECT BY

Определяет отношение между родительскими и дочерними строками иерархии.

UNION [ALL]

Означает, что результаты команды SELECT, предшествующей данному ключевому слову, должны быть объединены с результатами последующей команды SELECT.

INTERSECT

Означает, что результаты команды SELECT, которая предшествует данному ключевому слову, должны быть объединены с результатами последующей команды SELECT и только строки, присутствующие в обоих результатах, должны быть возвращены.

MINUS

Означает, что результаты команды SELECT, которая предшествует данному ключевому слову, должны быть объединены с результатами последующей команды SELECT. Все строки, появляющиеся в результатах второй команды SELECT, удаляются из множества строк, возвращенного первой командой.

ORDER BY

Указывает, что прежде чем строки возвращаются, они упорядочиваются. Сортировка может выполняться для выражений, псевдонимов или позиций. В данном случае *позиция* – это целое число, определяющее положение выражения в списке выборки. Считается, что первый элемент в списке выборки находится в позиции 1.

ASC

Значения упорядочиваются по возрастанию – от наименьшего к наибольшему. Это порядок по умолчанию.

DESC

Значения упорядочиваются по убыванию – от наибольшего к наименьшему.

FOR UPDATE

Означает, что выбранные строки будут заблокированы. Если указано ключевое слово OF, то будут заблокированы только строки названной таблицы.

NOWAIT

Означает, что если таблица уже заблокирована, то сервер Oracle не будет ждать снятия блокировки. Если не указать NOWAIT, сервер Oracle будет ожидать снятия блокировки.

Общие ключевые слова: *столбец*, *связь_БД*, *выражение*, *имя_раздела*, *схема*, *имя_подраздела*, *имя_таблицы*.

SET CONSTRAINT

```
SET {CONSTRAINT | CONSTRAINTS} {ALL | имя_ограничения[,имя_ограничения...]}
{IMMEDIATE | DEFERRED}
```

На уровне транзакции задает, будут ли определенные ограничения проверяться после каждой команды DML или же только в конце транзакции. Данная команда применяется только к откладываемым ограничениям. Для того чтобы проверить успешность откладываемых ограничений, можно выдать команду SET CONSTRAINTS ALL IMMEDIATE перед командой COMMIT.

Ключевые слова

ALL	Указывает, что данная команда действует для всех откладываемых ограничений в рамках транзакции.
имя_ограничения	Имя откладываемого ограничения.
IMMEDIATE	Указывает, что условие, налагаемое ограничением, будет проверяться после выполнения каждой команды DML.
DEFERRED	Указывает, что условие, налагаемое ограничением, будет проверяться после завершения и фиксации всей транзакции.

SET ROLE

```
SET ROLE
{ роль [IDENTIFIED BY пароль][,роль [IDENTIFIED BY пароль ...]]
| ALL [EXCEPT роль[,роль ...]]
| NONE
}
```

Включает и отключает роль для текущего сеанса.

Ключевые слова

роль	Имя предоставляемой роли.
IDENTIFIED BY	Задает пароль для роли. Необходимо, если роль защищена паролем.
ALL	Означает, что все выданные пользователю роли включены. Если применяется инструкция EXCEPT, то указанная роль может быть запрещена, но все остальные выданные пользователю роли будут действовать.
NONE	Означает, что все выданные пользователю роли не действуют для данного сеанса.

SET TRANSACTION

```
SET TRANSACTION [NAME 'имя']
{READ ONLY
| READ WRITE
| ISOLATION LEVEL {SERIALIZABLE | READ COMMITTED}
| USE ROLLBACK SEGMENT имя_сегмента
}
```

Учреждает текущую транзакцию как транзакцию только для чтения, для чтения и записи или указывает сегмент отката для данной транзакции. Если команда применяется, то она должна быть первой в соответствующей транзакции. Транзакция завершается командой COMMIT или COMMIT WORK.

Ключевые слова

NAME

Указывает имя данной транзакции. Появилось в Oracle9*i*.

READ ONLY

Означает, что транзакция занимается только чтением.

READ WRITE

Означает, что транзакция предназначена для чтения и записи.

ISOLATION_LEVEL

Определяет, как будут обрабатываться изменения БД.

SERIALIZABLE

Если производится попытка изменения строки, которая была изменена другим сеансом, но операция изменения еще не была зафиксирована, то команда не выполнится. Данный параметр соответствует уровню изоляции транзакций SERIALIZABLE, описанному в стандарте SQL-92.

READ COMMITTED

Если строка заблокирована другой незафискированной транзакцией, то команда будет ждать снятия блокировки строк. Это поведение сервера Oracle по умолчанию.

USE ROLLBACK SEGMENT

Сопоставляет транзакции сегмент отката с именем *имя_сегмента*. Данная инструкция подразумевает, что транзакция предназначена для чтения и записи и не действует для READ ONLY.

TRUNCATE

```
TRUNCATE
{ TABLE имя_таблицы [{PRESERVE | PURGE}
| {SNAPSHOT | MATERIALIZED VIEW} LOG]
| CLUSTER кластер
}
[{DROP | REUSE} STORAGE]
```

Удаляет все строки из таблицы или кластера.

Команда TRUNCATE не создает записей отката, поэтому для нее откат невозможен. Зато она выполняется очень быстро, и ее следует применять вместо DELETE FROM, если откат не нужен. Если таблица очищается и указана инструкция DROP STORAGE, то сохраняется только исходный экстент таблицы, все остальное пространство хранения освобождается.

Ключевые слова

кластер

Указывает имя кластера, из которого удаляются строки.

PRESERVE ... LOG

Означает, что существующие журналы материализованного представления или моментальной копии для данной таблицы должны быть сохранены при очистке таблицы. Такая возможность полезна, если таблица повторно загружается в про-

цессе операции Экспорт/TRUNCATE/Импорт, т. к. быстрое обновление не будет инициировано.

PURGE ... LOG

Указывает, что существующие журналы моментальной копии для данной таблицы должны быть очищены при очистке таблицы.

DROP STORAGE

Освобождает пространство, занимаемое строками, и возвращает его в пул свободного пространства. Это поведение по умолчанию.

REUSE STORAGE

Удерживает пространство, занимаемое удаленными строками. Такая возможность полезна, если в таблицу или кластер будут повторно загружены данные.

Общие ключевые слова и инструкции: *имя_таблицы*.

UPDATE

```
UPDATE
{имя_таблицы[@связь_БД]
|имя_таблицы PARTITION (имя_раздела)
|имя_таблицы SUBPARTITION (имя_подраздела)
|представление[@связь_БД]
|моментальная_копия[@связь_БД]
}
[псевдоним_таблицы]
SET
{ столбец={выражение | (подзапрос) | DEFAULT}
   [,столбец={выражение | (подзапрос) | DEFAULT} ... ]
| (столбец[,столбец ...])=подзапрос
}
[WHERE условие]
[RETURNING выражение[,выражение ...] INTO элемент_данных[,элемент_данных ...]]
```

Изменяет значение, хранящееся в одном или нескольких столбцах данных, в одной или нескольких таблицах, представлениях или моментальных копиях.

Ключевые слова

представление

Имя обновляемого представления.

моментальный_снимок

Имя обновляемой моментальной копии.

PARTITION

Указывает, что данные обновляются в разделе таблицы *имя_таблицы* с именем *имя_раздела*.

SUBPARTITION

Указывает, что данные обновляются в подразделе таблицы *имя_таблицы* с именем *имя_подраздела*.

подзапрос

Любая разрешенная команда SELECT. Имейте в виду, что подзапрос не может содержать инструкцию FOR UPDATE.

DEFAULT

Означает, что сервер Oracle должен использовать значение по умолчанию для столбца. Если оно не объявлено, то принимается равным NULL. Появилось в Oracle9*i*.

псевдоним_таблицы

Указывает псевдоним таблицы, представления или моментальной копии.

столбец

Имя столбца таблицы, представления или моментальной копии, которая будет обновлена.

WHERE

Указывает, что будут обновлены только строки, удовлетворяющие *условию*. Вычисляется *условие*, и обновляются только те строки, для которых оно равно TRUE. Если инструкция не указана, то будут обновлены все строки.

RETURNING

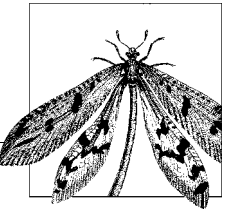
Указывает, что для строк, обновляемых данной командой, возвращаются значения указанных выражений. Действует только в рамках программы PL/SQL.

INTO

Указывает, что значения, возвращаемые для строк, обновляемых данной командой, будут сохранены в переменных PL/SQL *элемент_данных*.

Общие ключевые слова и инструкции: *столбец*, *связь_БД*, *выражение*, *имя_таблицы*.

8
Функции



Oracle содержит большое количество встроенных функций, которые можно использовать в SQL или в хранимых процедурах PL/SQL. Эти функции расширяют возможности языка SQL, выполняя часто встречающиеся операции. Например, в следующем SQL-предложении вызывается встроенная функция AVG, возвращающая среднее значение столбца salary для каждого из подразделений:

```
SELECT dept, AVG (salary) FROM emp GROUP BY dept
```

Пользователь может также создавать свои собственные функции и использовать их наравне со встроенными функциями Oracle. В главе 9 рассказано, как создавать и вызывать функции в языке PL/SQL.

В этой главе приведено краткое описание всех встроенных функций Oracle. Мы разделили их на следующие категории (внутри каждой категории функции упорядочены по алфавиту):

- Агрегатные и аналитические функции
- Функции для работы с числами
- Функции для работы с символами
- Функции для работы с датами
- Функции преобразования
- Объектные функции
- Функции для работы с XML
- Другие функции

Описание каждой функции включает в себя блок синтаксиса, содержащий параметры, которые могут быть переданы функции. В большинстве случаев каждый из параметров будет описан. Если же параметр был рассмотрен ранее в разделе «Общие ключевые слова и инструкции», будет приведена соответствующая ссылка. Если значение параметра очевидно (например, *выражение*), подробное описание отсутствует.

Общие ключевые слова и инструкции

При вызове большинства функций, описанных в данной главе, можно использовать следующие ключевые слова:

ASC

Означает, что последовательность упорядочивается по возрастанию.

DESC

Означает, что последовательность упорядочивается по убыванию.

dfmt

Задает спецификацию формата даты, состоящую из элементов, описанных в приложении D.

fmt

Задает спецификацию числового формата, состоящую из элементов, описанных в приложении C.

nlsparams

Задает символьную строку, состоящую из элементов, позволяющих указать различные характеристики поддержки национальных языков (National Language Support – NLS) для формирования результата:

`NLS_NUMERIC_CHARACTERS = ''dg''`

Определяет символы для обозначения разделителя десятичной дроби *d* и разделителя групп *g*. Кроме того, необходимо будет указать символы D и G в спецификации формата, чтобы задать местоположение десятичного и группового разделителей.

`NLS_CURRENCY = ''текст''`

Определяет обозначение для валюты (длиной до 10 символов), которое будет использоваться вместо любых символов L, содержащихся в спецификации формата.

`NLS_ISO_CURRENCY = ''территория''`

Определяет территорию NLS, обозначение валюты которой вы хотите использовать вместо любых символов C, содержащихся в спецификации формата.

Параметры *nlsparams* – это строки, встроенные в строку. Обратите внимание на то, что в данной инструкции используются две одинарные кавычки, а не одна двойная.

nlsdateparam

Задает строку в виде NLS_DATE_LANGUAGE = *имя_языка*, где *имя_языка* представляет собой разрешенное в NLS название языка. Имя языка в NLS определяет написание названий дней и месяцев.

NULLS FIRST

Указывает, что значения NULL должны находиться в начале упорядоченной последовательности.

NULLS LAST

Указывает, что значения NULL должны находиться в конце упорядоченной последовательности.

Агрегатные и аналитические функции

Агрегатные функции (*aggregate functions*) – это функции SQL, предназначенные для суммирования данных из нескольких строк таблицы или представления. Функции данной категории можно вызывать только из выражений PL/SQL.

Аналитические функции также суммируют данные нескольких строк, но при этом могут возвращать несколько строк из каждой группы, в то время как агрегатные функции возвращают всего одну строку.

Многие функции раздела могут использоваться и как агрегатные, и как аналитические. Аналитическую часть можно без труда распознать по ключевому слову OVER, предваряющему аналитическую инструкцию (см. далее раздел «Аналитическая инструкция»).

GROUP BY

Если инструкция GROUP BY применяется для агрегатной функции, то сервер Oracle выдает агрегатное значение для каждого неповторяющегося значения группируемых столбцов.

Инструкция GROUP BY приводит к тому, что строки результата группируются для каждого неповторяющегося значения в столбце, указанном в инструкции GROUP BY. Результаты агрегатных функций возвращаются для каждого из таких значений.

DISTINCT и ALL

Большая часть агрегатных функций допускает использование в списке параметров ключевых слов DISTINCT и ALL. Эти ключевые слова изменяют способ обработки повторяющихся значений столбцов и имеют следующий смысл:

DISTINCT

Приводит к тому, что агрегатные функции учитывают только различные значения и игнорируют дубликаты.

ALL

Означает, что агрегатные функции обрабатывают все значения, включая дубликаты. Это поведение по умолчанию.

Аналитическая инструкция

Если функция применяется как аналитическая, то перед аналитической инструкцией всегда стоит ключевое слово OVER. Аналитические функции всегда используют аналитическую инструкцию, которая имеет такой синтаксис:

```
[PARTITION BY выражение [,выражение...] ]
[ORDER [SIBLINGS] BY
    {выражение | позиция | псевдоним_столбца} [ASC | DESC] [NULLS FIRST | NULLS LAST]
  [,{выражение | позиция | псевдоним_столбца} [ASC | DESC] [NULLS FIRST | NULLS LAST]...]
  {ROWS | RANGE} BETWEEN
    {UNBOUNDED PRECEDING | CURRENT ROW | выражение_для_значения {PRECEDING | FOLLOWING}} AND
    {UNBOUNDED PRECEDING | CURRENT ROW | выражение_для_значения {PRECEDING | FOLLOWING}}}]
```

Параметры

PARTITION

Означает, что результирующее множество запроса должно быть разбито на один или несколько разделов на основе одного или нескольких выражений (*выражение*).

ORDER BY

Указывает, каким образом данные раздела должны быть упорядочены.

позиция

Означает, что раздел должен быть упорядочен в соответствии с выражением, указанным в данной относительной позиции в списке выражений.

псевдоним_столбца

Указывает, что раздел должен быть упорядочен по столбцу, определяемому данным псевдонимом.

ROWS

Означает, что окно определено в терминах физических строк.

RANGE

Означает, что окно определено в терминах логического смещения.

BETWEEN

Означает, что будут определены начальная и конечная точки окна.

UNBOUNDED PRECEDING

Означает, что окно заканчивается последней строкой запроса.

UNBOUNDED FOLLOWING

Означает, что окно начинается с первой строки запроса.

CURRENT ROW

Означает, что окно начинается с текущей строки или значения в зависимости от того, какой из параметров (ROWS или RANGE) был задан.

выражение_для_значения

Число, представляющее собой физическое смещение при указании параметра ROWS или логическое смещение при задании параметра RANGE.

Общие ключевые слова и инструкции: ASC, DESC, NULLS FIRST, NULLS LAST.

AVG

```
AVG ([DISTINCT | ALL] выражение) [OVER (аналитическая_инструкция)]
```

Вычисляет среднее значение столбца или выражения для множества строк, возвращаемых запросом, или множества строк, указанных в инструкции GROUP BY.

CORR

```
CORR (выражение1, выражение2) [OVER (аналитическая_инструкция)]
```

Вычисляет коэффициент корреляции для множества числовых пар, предоставляемых *выражением1* и *выражением2* (*выражение1* и *выражение2* должны возвращать числа).

COUNT

```
COUNT ([DISTINCT | ALL] выражение [,DISTINCT | ALL] выражение...] | *)
  [OVER (аналитическая_инструкция)]
```

Вычисляет количество строк, возвращаемых выражением. Если использовать в качестве аргумента функции специальный символ *, то при подсчете будут учтены все строки, включая и содержащие значения NULL.

COVAR_POP

```
COVAR_POP (выражение1, выражение2) [OVER (аналитическая_инструкция)]
```

Вычисляет *ковариацию совокупности* (*population covariance*) для множества пар, предоставляемых *выражением1* и *выражением2* (*выражение1* и *выражение2* должны возвращать числа). После удаления тех пар, в которых *выражение1* или *выражение2* возвращает NULL, сервер Oracle выполняет следующее вычисление.

```
(SUM (выражение1 * выражение2) - SUM (выражение2) * SUM (выражение1) / n) / n
```

где n – количество возвращаемых строк.

COVAR_SAMP

```
COVAR_SAMP (выражение1, выражение2) [OVER (аналитическая_инструкция)]
```

Вычисляет *выборочную ковариацию* (*sample covariance*) для множества пар, предоставляемых *выражением1* и *выражением2* (*выражение1* и *выражение2* должны возвращать числа). После удаления тех пар, в которых *выражение1* или *выражение2* возвращает NULL, сервер Oracle выполняет следующее вычисление.

```
(SUM(выражение1 * выражение2) - SUM(выражение1) * SUM(выражение2) / n) / (n-1)
```

где n – количество возвращаемых строк.

CUME_DIST

Агрегатный синтаксис:

```
CUME_DIST (выражение[,выражение...]) WITHIN GROUP
    (ORDER BY выражение [ASC | DESC][NULLS {FIRST | LAST}]
        [, выражение [ASC | DESC] [NULLS {FIRST | LAST}]...])
```

Аналитический синтаксис:

```
CUME_DIST () OVER (аналитическая_инструкция)
```

Вычисляет относительную позицию строки по отношению ко всем строкам группы агрегации, для гипотетической строки, которая идентифицируется аргументами функции и спецификацией ORDER BY. Возвращаемое значение больше 0 и меньше либо 1. Все аргументы данной функции должны быть постоянными выражениями. Выражения в базовой функции должны занимать те же позиции, что и в инструкции ORDER BY.

Параметры

Общие ключевые слова и инструкции: ASC, DESC, NULLS FIRST, NULLS LAST.

DENSE_RANK

Агрегатный синтаксис:

```
DENSE_RANK (выражение[,выражение...]) WITHIN GROUP
    (ORDER BY выражение [ASC | DESC][NULLS {FIRST | LAST}]
        [, выражение [ASC | DESC] [NULLS {FIRST | LAST}]...])
```

Аналитический синтаксис:

```
DENSE_RANK () OVER (аналитическая_инструкция)
```

Вычисляет позицию строки (при ранжировании без пропусков) по отношению ко всем строкам группы агрегации, для гипотетической строки, которая идентифицируется аргументами функции и спецификацией ORDER BY. Возвращается целое значение больше либо равное 1, наибольшее возможное значение равно количеству уникальных значений, возвращаемых запросом. Все аргументы данной функции должны быть постоянными выражениями. Выражения в базовой функции должны занимать те же позиции, что и в инструкции ORDER BY. Параметры совпадают с параметрами функции CUME_DIST.

FIRST_VALUE

```
FIRST_VALUE (выражение) OVER (аналитическая_инструкция)
```

Возвращает первое значение из упорядоченного множества значений. Это только аналитическая функция.

GROUP_ID

```
GROUP_ID ()
```

Присваивает уникальное целое число, начиная с 0, повторяющимся группам, получившимся в результате применения инструкции GROUP BY. В вызов функции необходимо включать скобки. Появилась в Oracle9*i*.

GROUPING

```
GROUPING (выражение)
```

Предназначена для применения в команде SELECT, содержащей оператор CUBE или ROLLUP. Операторы CUBE и ROLLUP приводят к вставке в запросы дополнительных строк со значениями NULL для суммирования группы записей. Возвращает 1, если значение NULL является результатом дополнительной строки, которая возвращается вследствие применения CUBE или ROLLUP; в противном случае возвращается 0. Появилась в Oracle8*i*.

GROUPING_ID

```
GROUPING_ID (выражение[,выражение...])
```

Присваивает целое число, соответствующее битовому вектору GROUPING, сопоставленному строке, избавляя от необходимости применения нескольких функций GROUPING. Включается в команду SELECT, содержащую такие расширения GROUP BY, как ROLLUP или CUBE. Появилась в Oracle9*i*.

KEEP

```
агрегатная_функция KEEP (DENSE_RANK {FIRST | LAST}
    ORDER BY выражение [ASC | DESC][NULLS {FIRST | LAST}]
      [, выражение [ASC | DESC] [NULLS {FIRST | LAST}]...])
    [OVER (аналитическая_инструкция)]
```

Применяется совместно с другими агрегатными функциями (MIN, MAX, SUM, AVG, COUNT, VARIANCE или STDDEV) для работы с набором значений из множества

строк, ранжированных как FIRST или LAST, что указано в инструкции ORDER BY. Выражения должны занимать те же позиции, что и в инструкции ORDER BY.

Параметры

агрегатная_функция

MIN, MAX, SUM, AVG, COUNT, VARIANCE или STDDEV.

DENSE RANK

Указывает, что агрегирование производится только для строк с максимальным или минимальным рангом.

аналитическая_инструкция

Может использовать только часть PARTITION BY аналитической инструкции.

Общие ключевые слова и инструкции: ASC, DESC, NULLS FIRST, NULLS LAST.

LAG

```
LAG (выражение [,смещение][,умолчание]) OVER (аналитическая_инструкция)
```

Предоставляет одновременный доступ к нескольким строкам таблицы без выполнения самосоединения. Только аналитическая функция.

Параметры

смещение

Физическое смещение назад по отношению к текущей строке для дополнительной извлекаемой строки.

умолчание

Значение, которое возвращается, если смещение выходит за пределы окна.

LAST_VALUE

```
LAST_VALUE (выражение) OVER (аналитическая_инструкция)
```

Возвращает последнее значение в упорядоченном множестве значений. Только аналитическая функция.

LEAD

```
LEAD (выражение [,смещение][,умолчание]) OVER (аналитическая_инструкция)
```

Предоставляет одновременный доступ к нескольким строкам таблицы без выполнения самосоединения. Только аналитическая функция.

Параметры

смещение

Физическое смещение вперед по отношению к текущей строке для дополнительной извлекаемой строки.

умолчание

Значение, которое возвращается, если смещение выходит за пределы окна.

MAX

```
MAX ([DISTINCT | ALL] выражение)
```

Вычисляет максимальное значение столбца или выражения над множеством строк, возвращенных запросом, или над множеством строк, определяемом инструкцией GROUP BY. Наличие ключевого слова ALL или DISTINCT не влияет на значение, возвращаемое функцией MAX.

MIN

```
MIN ([DISTINCT | ALL] выражение)
```

Вычисляет минимальное значение столбца или выражения над множеством строк, возвращенных запросом, или над множеством строк, определяемом инструкцией GROUP BY. Наличие ключевого слова ALL или DISTINCT не влияет на значение, возвращаемое функцией MIN.

NTILE

```
NTILE (выражение_ntile) OVER (аналитическая_инструкция)
```

Разбивает упорядоченное множество данных на порции и присваивает каждой строке номер соответствующей порции. Только аналитическая функция.

Параметры

выражение_ntile

Задает число, представляющее количество частей, на которое разбиваются строки.

PERCENT_RANK

Агрегатный синтаксис:

```
PERCENT_RANK (выражение [,выражение...]) WITHIN GROUP
    (ORDER BY выражение [ASC | DESC][NULLS {FIRST | LAST}]
        [, выражение [ASC | DESC] [NULLS {FIRST | LAST}]...])
```

Аналитический синтаксис:

```
PERCENT_RANK () OVER (аналитическая_инструкция)
```

Вычисляет относительную позицию строки, деленную на количество строк в наборе агрегирования, для гипотетической строки, идентифицируемой аргументами функции и спецификацией ORDER BY. Возвращаемое значение > 0 и <= 1. Все аргументы данной функции должны быть постоянными выражениями. Выражения в базовой функции должны занимать те же позиции, что и в инструкции ORDER BY.

Параметры

Общие ключевые слова и инструкции: ASC, DESC, NULLS FIRST, NULLS LAST.

PERCENTILE_CONT

```
PERCENTILE_CONT (выражение_процентиля) WITHIN GROUP
    (ORDER BY выражение [ASC | DESC]) [OVER (аналитическая_инструкция)]
```

Вычисляет интерполяцией значение, которое должно относиться к указанному значению процентиля *выражение_процентиля* с учетом инструкции ORDER BY. Функция предполагает *непрерывное распределение* (*continuous distribution*) значений. Появилась в Oracle9*i*.

Параметры

выражение_процентиля

Должно оцениваться как число от 0 до 1, задающее значение процентиля.

Общие ключевые слова и инструкции: ASC, DESC.

PERCENTILE_DISC

```
PERCENTILE_DISC (выражение_процентиля) WITHIN GROUP
    (ORDER BY выражение [ASC | DESC]) [OVER (аналитическая_инструкция)]
```

Вычисляет интерполяцией значение, которое должно относиться к указанному значению процентиля *выражение_процентиля* с учетом инструкции ORDER BY. Функция предполагает *дискретное распределение* (*discrete distribution*) значений. Появилась в Oracle9*i*.

Параметры

выражение_процентиля

Должно оцениваться как число от 0 до 1, задающее значение процентиля.

Общие ключевые слова и инструкции: ASC, DESC.

RANK

Агрегатный синтаксис:

```
RANK (выражение[,выражение...]) WITHIN GROUP
    (ORDER BY выражение [ASC | DESC][NULLS {FIRST | LAST}]
        [, выражение [ASC | DESC] [NULLS {FIRST | LAST}]...])
```

Аналитический синтаксис:

```
RANK () OVER (аналитическая_инструкция)
```

В качестве агрегатной функции вычисляет для строки, которая идентифицируется аргументами функции и спецификацией ORDER BY, ранг данной строки по отношению ко всем строкам группы агрегирования. Как аналитическая функция, вычисляет ранг каждой возвращаемой строки по отношению к другим возвращенным строкам. Возвращается целое значение >= 1, наибольшее возможное значение равно количеству уникальных значений, возвращаемых запросом. Все аргументы данной функции должны быть постоянными выражениями. Выражения должны занимать те же позиции, что и в инструкции ORDER BY.

Общие ключевые слова и инструкции: ASC, DESC, NULLS FIRST, NULLS LAST.

RATIO_TO_REPORT

```
RATIO_TO_REPORT (выражение_отношения) OVER (аналитическая_инструкция)
```

Вычисляет отношение значения к сумме множества значений. Только аналитическая функция.

Параметры

выражение_отношения

Возвращает число, которое должно быть сравнено с суммой множества значений, возвращаемых *аналитической_инструкцией*.

REGR...

```
REGR_тип_регрессии (выражение1, выражение2) [OVER (аналитическая_инструкция)]
```

Вычисляет *линию регрессии методом наименьших квадратов* (least-squares regression) для пар чисел, возвращаемых *выражением1* и *выражением2*.

Параметры

тип_регрессии

Может принимать одно из следующих значений:

SLOPE	Возвращает наклон линии регрессии.
INTERCEPT	Возвращает угловой коэффициент Y линии регрессии.
COUNT	Возвращает количество не-NULL-пар, использованных для вычисления линии регрессии.
R2	Возвращает коэффициент смешанной корреляции (также называемый коэффициентом хи-квадрат (Chi-Squared), или согласия (Goodness of Fit)) для линии регрессии.
AVGX	Возвращает среднее значение независимой переменной (*выражение2*) для линии после удаления значений NULL. Возвращаемое значение может быть равно NULL.
AVGY	Возвращает среднее значение независимой переменной (*выражение1*) для линии после удаления значений NULL. Возвращаемое значение может быть равно NULL.
SXX	Вычисляет значение REGR_COUNT (*выражение1*,*выражение2*) * VAR_POP(*выражение2*) после удаления NULL-пар *выражение1*,*выражение2*.
SYY	Вычисляет значение REGR_COUNT (*выражение1*,*выражение2*) * VAR_POP(*выражение1*) после удаления NULL-пар *выражение1*,*выражение2*.
SXY	Вычисляет значение REGR_COUNT (*выражение1*,*выражение2*) * COVAR_POP(*выражение1*,*выражение2*) после удаления NULL-пар *выражение1*,*выражение2*.

ROW_NUMBER

```
ROW_NUMBER () OVER (аналитическая_инструкция)
```

Присваивает каждой строке запроса уникальный последовательный порядковый номер, начиная с 1. Только аналитическая функция.

STDDEV

```
STDDEV ([DISTINCT | ALL] выражение) [OVER (аналитическая_инструкция)]
```

Вычисляет *выборочное стандартное отклонение* (*sample standard deviation*) значений столбца или выражения над множеством строк, возвращаемых запросом или множеством строк, которое определяется в инструкции GROUP BY. Выборочное стандартное отклонение использует количество строк, возвращенных запросом, в качестве знаменателя. STDDEV аналогична функции STDDEV_SAMP, только данная функция возвращает 0, а не NULL, если *выражение* возвращает всего одну строку.

STDDEV_POP

```
STDDEV_POP ([DISTINCT | ALL] выражение) [OVER (аналитическая_инструкция)]
```

Вычисляет *отклонение генеральной совокупности* (*population standard deviation*) значений столбца или выражения над множеством строк, возвращаемых запросом или множеством строк, которое определяется в инструкции GROUP BY. Отклонение генеральной совокупности использует количество строк, возвращенных запросом, в качестве знаменателя.

STDDEV_SAMP

```
STDDEV_SAMP ([DISTINCT | ALL] выражение) [OVER (аналитическая_инструкция)]
```

Вычисляет выборочное стандартное отклонение значений столбца или выражения над множеством строк, возвращаемых запросом или множеством строк, которое определяется в инструкции GROUP BY. Выборочное стандартное отклонение использует количество строк, возвращенных запросом, в качестве знаменателя.

SUM

```
SUM ([DISTINCT | ALL] выражение) [OVER (аналитическая_инструкция)]
```

Вычисляет сумму значений столбца или выражения над множеством строк, возвращаемых запросом или множеством строк, определяемом в инструкции GROUP BY.

VAR_POP

```
VARIANCE ([DISTINCT | ALL] выражение) [OVER (аналитическая_инструкция)]
```

Вычисляет *дисперсию генеральной совокупности* (*population variance*) для значений столбца или выражения над множеством строк, возвращаемых запросом или множеством строк, которое определяется в инструкции GROUP BY. Дисперсия генеральной совокупности использует количество строк, возвращенных запросом, в качестве знаменателя.

VAR_SAMP

```
VARIANCE ([DISTINCT | ALL] выражение) [OVER (аналитическая_инструкция)]
```

Вычисляет *выборочную дисперсию* (*sample variance*) значений столбца или выражения над множеством строк, возвращаемых запросом или множеством строк, которое определяется в инструкции GROUP BY. Выборочное стандартное отклонение использует количество строк, возвращенных запросом, в качестве знаменателя. VAR_SAMP почти идентична функции VARIANCE, разница лишь в том, что данная функция

возвращает NULL, если *выражение* возвращает всего одну строку, в то время как VARIANCE в такой ситуации возвращает 0.

VARIANCE

```
VARIANCE ([DISTINCT | ALL] выражение) [OVER (аналитическая_инструкция)]
```

Вычисляет выборочную дисперсию значений столбца или выражения над множеством строк, возвращаемых запросом или множеством строк, которое определяется в инструкции GROUP BY. Выборочное стандартное отклонение использует количество строк, возвращенных запросом, в качестве знаменателя.

Функции для работы с числами

Рассматриваемые в этом разделе функции принимают в качестве входных параметров числа и возвращают числовые значения.

ABS

```
ABS (n)
```

Возвращает абсолютное значение числа n.

ACOS

```
ACOS (n)
```

Возвращает арккосинус значения из диапазона от -1 до 1, величину угла в радианах. Значение функции находится в диапазоне от 0 до π включительно. Обращает вывод функции COS.

ASIN

```
ASIN (n)
```

Возвращает арксинус значения из диапазона от -1 до 1, величину угла в радианах. Значение функции находится в диапазоне от $-\pi/2$ до $\pi/2$ включительно. Обращает вывод функции SIN.

ATAN

```
ATAN (n)
```

Возвращает арктангенс значения, величину угла в радианах. Значение функции находится в диапазоне от $-\pi/2$ до $\pi/2$ включительно. Обращает вывод функции TAN.

ATAN2

```
ATAN2 (n,m)
```

Возвращает арктангенс значения n/m, величину угла в радианах. Значение функции находится в диапазоне от $-\pi/2$ до $\pi/2$ включительно. ATAN2(n,m) – это то же самое, что ATAN(n/m).

BITAN

```
BITAN (n,m)
```

Выполняет операцию AND над битами аргументов n и m, которые должны быть неотрицательными целыми числами. Тип данных возвращаемого значения не определен, поэтому данная функция всегда должна быть упакована в другую функцию, такую как TO_NUMBER.

CEIL

```
CEIL (n)
```

Возвращает наименьшее целое, превосходящее значение аргумента или равное ему. Если n – целое, возвращается само n.

COS

```
COS (n)
```

Возвращает косинус угла, где n – величина угла в радианах.

COSH

```
COSH (n)
```

Возвращает гиперболический косинус угла, где n – величина угла в радианах.

EXP

```
EXP (n)
```

Возвращает значение e, возведенное в степень n.

FLOOR

```
FLOOR (n)
```

Возвращает наибольшее целое, меньшее значения аргумента или равное ему.

GREATEST

```
GREATEST (выражение [,выражение...])
```

Возвращает наибольшее значение из предложенного списка аргументов. Может применяться как для чисел, так и для символьных строк и дат. При использовании данных разных типов сервер Oracle воспринимает в качестве основного тип данных первого аргумента и преобразует (если это возможно) все остальные аргументы к данному типу, а затем выбирает наибольшее значение.

LEAST

```
LEAST (выражение [,выражение...])
```

Возвращает наименьшее значение из предложенного списка аргументов. Может применяться как для чисел, так и для символьных строк и дат. При использовании данных разных типов сервер Oracle воспринимает как основной тип данных первого аргумента и преобразует (если это возможно) все остальные аргументы к данному типу, а затем выбирает наименьшее значение.

LN

```
LN (n)
```

Возвращает натуральный логарифм числа *n*.

LOG

```
LOG (m,n)
```

Возвращает логарифм числа *n* по основанию *m*.

MOD

```
MOD (m,n)
```

Возвращает остаток от деления *m* на *n*.

POWER

```
POWER (m,n)
```

Возвращает m^n (m^{-n} эквивалентно $1/m^n$).

Параметры

m Ненулевое число.

n Показатель степени. Если *m* – это положительное число, то *n* может быть любым положительным или отрицательным числом. Если же *m* отрицательное, то *n* должно быть целым.

ROUND

```
ROUND (n,m)
```

Округляет значение *n* до количества десятичных разрядов, указанных значением *m*.

SIGN

```
SIGN (n)
```

Возвращает значение, определяющее знак *n*. Возможны следующие значения функции SIGN:

-1 Отрицательное число.

0 Это ноль.

1 Число больше нуля.

SIN

```
SIN (n)
```

Возвращает синус угла, где *n* – величина угла в радианах.

SINH

```
SINH (n)
```

Возвращает гиперболический синус угла, где *n* – величина угла в радианах.

SQRT

```
SQRT(n)
```

Возвращает квадратный корень из *n*, где *n* – положительное число.

TAN

```
TAN (n)
```

Возвращает тангенс угла, где *n* – величина угла в радианах.

TANH

```
TANH (n)
```

Возвращает гиперболический тангенс угла, где *n* – величина угла в радианах.

TRUNC

```
TRUNC (n [,m])
```

Усекает число *n* до количества десятичных разрядов, заданного аргументом *m*.

WIDTH_BUCKET

```
WIDTH_BUCKET (выражение, min, max, n)
```

Функция предназначена для построения гистограмм. Возвращает номер столбца гистограммы, в который попадает значение *выражения*. Если сервер Oracle встречает значения, которые меньше *min* и/или больше *max*, то они помещаются в столбцы «переполнения» соответственно с номерами 0 и *n*+1. Появилась в Oracle9*i*.

Параметры

выражение	Задает выражение, для которого строится гистограмма. Значением выражения должно быть число или значение типа DATE.
min	Определяет наименьшее допустимое значение *выражения*.
max	Определяет наибольшее допустимое значение *выражения*.
n	Определяет количество столбцов в гистограмме.

Функции для работы с символами

Эти функции работают с символьными значениями и возвращают как символьные, так и числовые значения. Имейте в виду, что описанные в разделе «Функции для работы с числами» функции GREATEST и LEAST также могут работать с символьными значениями.

ASCII

```
ASCII (символ)
```

Возвращает десятичное представление символа, основываясь на действующем для базы данных наборе символов. Значение ASCII возвращается лишь в том случае, если символы БД принадлежат к 7-битному набору символов ASCII. Если *символ* – это строка, то возвращается ASCII-значение первого символа строки.

CHR

```
CHR (n [USING NCHAR_CS])
```

Возвращает символ из набора символов, используемых БД, который соответствует числу *n*.

Параметры

n

Числовое значение, представляющее символ, который должен быть возвращен.

USING NCHAR_CS

Указывает, что будет использоваться национальный набор символов базы данных.

CONCAT

```
CONCAT (строка1, строка2)
```

Соединяет две строки и возвращает результат.

INITCAP

```
INITCAP (строка)
```

Возвращает полученную строку, при этом первые буквы всех слов строки выводятся в верхнем регистре, а остальные – в нижнем.

INSTR...

```
INSTR (строка1, строка2[, n[, m]])
```

Выполняет поиск *строки2* в *строке1* и возвращает номер символа *строки1*, с которого начинается *строка2*. Функция INSTR может принимать следующие формы:

INSTR Использует набор символов входного значения.

INSTRB Оперирует байтами вместо символов.

INSTRC Работает с символами Unicode (появилось в Oracle9*i*).

INSTR2 Работает с кодами UCS2 (появилось в Oracle9*i*).

INSTR4 Работает с кодами UCS4 (появилось в Oracle9*i*).

Параметры

строка1 Строка, в которой производится поиск.

строка2 Искомая строка.

n Позиция символа, с которого начинается поиск. По умолчанию поиск начинается с первого символа, т. е. с позиции 1. Отрицательные значения указывают начальную позицию, отсчитываемую от правого края *строки1*.

m Какое из вхождений *строки2* вы хотите найти (если она встречается несколько раз). По умолчанию будет искаться первое вхождение.

LENGTH...

```
LENGTH (строка)
```

Возвращает количество символов в строке. Функция LENGTH может принимать следующие формы:

LENGTH Использует набор символов входного значения.

LENGTHB Оперирует байтами вместо символов.

LENGTHC Работает с символами Unicode (появилось в Oracle9*i*).

LENGTH2 Работает с кодами UCS2 (появилось в Oracle9*i*).

LENGTH4 Работает с кодами UCS4 (появилось в Oracle9*i*).

LOWER

```
LOWER (строка)
```

Преобразует все символы строки к нижнему регистру.

LPAD

```
LPAD (строка1, n[, строка2])
```

Дополняет *строку1* слева пробелами или копиями символьной строки *строка2*, до тех пор пока результирующая строка не достигнет *n* символов.

Параметры

строка1 Строка, которая будет дополнена.

n Результирующая длина *строки1* после заполнения.

строка2 Строка, которая будет добавлена слева к *строке1*. По умолчанию добавляются пробелы.

LTRIM

```
LTRIM (строка1 [,строка2])
```

Удаляет символы, указанные в *строке2*, слева из *строки1*.

Параметры

строка1 Строка, в которой производится поиск.

строка2 Строка, которая будет удалена слева из *строки1*. По умолчанию удаляются пробелы.

NLS_INITCAP

```
NLS_INITCAP (строка[, 'NLS_SORT=сортировка'])
```

Работает как INITCAP, но использует национальный набор символов.

Параметры

строка Входная символьная строка.

сортировка Имя последовательности лингвистической сортировки, которая задает правила преобразования регистров в используемом языке.

NLS_LOWER

```
NLS_LOWER (строка[, 'NLS_SORT=сортировка'])
```

Работает как LOWER, но использует национальный набор символов.

Параметры

строка Входная символьная строка.

сортировка Имя последовательности лингвистической сортировки, которая задает правила преобразования регистров в используемом языке.

NLS_UPPER

```
NLS_UPPER (строка[, 'NLS_SORT=сортировка'])
```

Работает как UPPER, но использует национальный набор символов.

Параметры

строка Входная символьная строка.

сортировка Имя последовательности лингвистической сортировки, которая задает правила преобразования регистров в используемом языке.

NLSSORT

```
NLSSORT (строка[, 'NLS_SORT=сортировка'])
```

Возвращает строку байтов, служащую для представления значения, которое будет сортироваться с применением последовательности лингвистической сортировки.

Параметры

строка Входная символьная строка.

сортировка Имя последовательности сортировки с учетом языка, или ключевое слово BINARY (при указании которого функция возвращает *строку*).

REPLACE

```
REPLACE (строка, строка_поиска [,строка_замены])
```

Просматривает строку и заменяет одну подстроку другой.

Параметры

строка

Строка, в которой производится поиск.

строка_поиска

Заменяемая подстрока.

строка_замены

Строка, выступающая в качестве замены *строки_поиска*. Если этот необязательный аргумент не указан, то все вхождения *строки_поиска* удаляются.

RPAD

```
RPAD (строка1, n[, строка2])
```

Дополняет *строку1* справа пробелами или копиями символьной строки *строка2*, до тех пор пока результирующая строка не достигнет длины *n* символов.

Параметры

строка1 Строка, которая будет дополнена.

n Результирующая длина строки после заполнения.

строка2 Строка, которая будет добавлена справа к *строке1*. По умолчанию добавляются пробелы.

RTRIM

```
RTRIM (строка1 [,строка2])
```

Удаляет символы, указанные в *строке2*, справа из *строки1*.

Параметры

строка1 Строка, в которой производится поиск.

строка2 Строка, которая будет удалена справа из строки *строка1*. По умолчанию удаляются пробелы.

SOUNDEX

```
SOUNDEX (строка)
```

Возвращает строку символов, представляющую собой фонетическую транскрипцию *строки*. Применение функции SOUNDEX может упростить поиск строки, т. к. орфографически правильный ввод строки будет не обязателен.

Для получения строки аргумента выполните следующие шаги:

1. Оставьте первую букву строки.
2. Удалите все вхождения букв: a, e, h, i, o, u, w, y.

3. Все оставшиеся буквы замените цифрами (правила замены приведены в табл. 8.1).
4. Усеките получившуюся строку до 4 символов.

Таблица 8.1. Сопоставление букв цифрам, используемое функцией SOUNDEX

Буквы	Цифры SOUNDEX
b, f, p, v	1
c, g, j, k, q, s, x, z	2
d, t	3
l	4
m, n	5
r	6

SUBSTR...

```
SUBSTR (строка, m [,n])
```

Просматривает *строку* и возвращает подстроку длиной *n* символов, которая начинается в позиции *m* (отсчет ведется слева). SUBSTR может принимать следующие формы:

SUBSTR Использует набор символов входного значения.

SUBSTRB Оперирует байтами вместо символов.

SUBSTRC Работает с символами Unicode (появилось в Oracle9*i*).

SUBSTR2 Работает с кодами UCS2 (появилось в Oracle9*i*).

SUBSTR4 Работает с кодами UCS4 (появилось в Oracle9*i*).

Параметры

строка Строка, в которой производится поиск.

m Начальная позиция в строке по отношению к левому краю. Отрицательные значения указывают смещение от правого края *строки1*.

n Количество возвращаемых символов. Если параметр не указан, то будут возвращены все символы, начиная с позиции *m*.

TRANSLATE

```
TRANSLATE (строка, исходная_строка, строка_замены)
```

Изменяет строку ввода, заменяя все вхождения *исходной_строки* на *строку_замены*.

Параметры

строка

Изменяемая строка.

исходная_строка

Строка, которая должна быть заменена.

строка_замены

Строка, которая должна заменить *исходную_строку*.

TREAT

```
TREAT (выражение AS [REF] [схема.]тип)
```

Изменяет объявленный тип выражения *выражение* на тип, указанный как *тип*. Необходимо, чтобы *тип* был супертипом или подтипом типа *выражения*. В противном случае функция возвращает NULL. Появилась в Oracle9*i*.

Параметры

REF Указывает, что значение для *выражения* – это REF.

схема Владелец *типа*.

тип Тип, который должен быть использован для *выражения*.

TRIM

```
TRIM ([LEADING | TRAILING | BOTH] [удаляемый_символ FROM] строка)
```

Удаляет символы из строки.

Параметры

LEADING

Указывает, что должны быть отброшены только начальные пробелы.

TRAILING

Указывает, что должны быть отброшены только конечные пробелы.

BOTH

Указывает, что должны быть отброшены и начальные, и конечные пробелы. Это значение по умолчанию.

удаляемый_символ

Означает, что удаляется не пробел, а какой-то другой символ.

строка

Входная строка.

UPPER

```
UPPER (строка)
```

Преобразует все символы строки к верхнему регистру.

Функции для работы с датой и временем

Функции работают со значениями даты или времени и возвращают именно такие значения. В разделе будут описаны как традиционные функции дат, так и новые функции интервалов и временных меток, появившиеся в Oracle9*i*.

В нижеследующих описаниях *дата* всегда будет означать выражение стандартного типа данных Oracle DATE.

ADD_MONTHS

```
ADD_MONTHS (дата, n)
```

Добавляет *n* месяцев к *дате*. Отрицательные значения *n* соответствуют вычитанию месяцев из *даты*. При определении дня месяца действуют следующие правила:

- Если исходная дата представляет собой последний день месяца, то получившаяся дата будет скорректирована так, чтобы тоже являться последним днем месяца.
- Если сохранить день месяца не удается из-за того, что в новом месяце меньше дней, чем в исходном, то полученная дата уменьшается с тем, чтобы быть корректной датой нового месяца.

Параметр

n Количество месяцев, прибавляемых к *дате*.

CURRENT_DATE

```
CURRENT_DATE
```

Возвращает текущие дату и время для часового пояса сессии в виде значения типа DATE. До тех пор пока с помощью команды ALTER SESSION не будет изменен часовой пояс сессии, команда CURRENT_DATE будет возвращать дату и время того сервера, на котором вы зарегистрировались при входе в систему. Появилась в Oracle9*i*.

CURRENT_TIMESTAMP

```
CURRENT_TIMESTAMP [(точность)]
```

Возвращает текущие дату и время для часового пояса сессии в виде значения типа TIMESTAMP WITH TIMEZONE. По умолчанию *точность* равна 6. Команда CURRENT TIMESTAMP похожа на LOCAL_TIMESTAMP, но возвращает значение типа TIMESTAMP WITH TIME ZONE, а не TIMESTAMP. Появилась в Oracle9*i*.

DBTIMEZONE

```
DBTIMEZONE
```

Возвращает часовой пояс БД. Результатом может быть как смещение часового пояса, так и его имя, в зависимости от того, как в самой последней команде CREATE DATABASE или ALTER DATABASE был определен часовой пояс. Появилась в Oracle9*i*.

EXTRACT

```
EXTRACT ({
    [YEAR | MONTH | DAY | HOUR | MINUTE | SECOND] |
    [TIMEZONE_HOUR | TIMEZONE_MINUTE] |
    [TIMEZONE_REGION | TIMEZONE_ABBR] }
    FROM {выражение_даты_времени | выражение_значения_интервала} )
```

Извлекает значение выбранного поля даты и времени из указанного выражения даты и времени или выражения интервала. Извлекаемое поле должно быть корректной составляющей указанного выражения. Появилась в Oracle9*i*.

Параметры

выражение_даты_времени

Выражение, вычисляемое как тип Oracle DATE.

выражение_значения_интервала

Выражение, вычисляемое как значение временного интервала Oracle.

FROM_TZ

```
FROM_TZ (значение_временной_метки, значение_часового_пояса)
```

Преобразует *значение_временной_метки* для часового пояса в значение типа TIMESTAMP WITH TIMEZONE. Появилась в Oracle9*i*.

Параметры

значение_временной_метки

Значение преобразуемой временной метки.

значение_часового_пояса

Символьная строка в виде TZH:TZM или символьное выражение, возвращающее строку вида TZR с необязательным форматом TZD.

INTERVAL DAY TO SECOND

```
INTERVAL '{целое1 | целое1 выражение_времени | выражение_времени}'
    {[DAY | HOUR | MINUTE] (точность_первой_части)|
    SECOND (точность_первой_части [, точность_дробной_части])}
    [TO {DAY | HOUR | MINUTE | SECOND [(точность_дробной_части)]}]
```

Преобразует целые литералы, представляющие дни, часы, минуты и секунды в интервал времени. В этой функции может применяться одна или две единицы измерения времени. Если указаны две единицы измерения, то первая должна быть больше, чем секунда, как например, INTERVAL ... DAY TO SECOND или INTERVAL ... HOUR TO MINUTE. Появилась в Oracle9*i*.

Функции INTERVAL можно применять для выполнения арифметических операций с датами, как показано в разделе «Примеры» (подробная информация о типах данных INTERVAL и о других типах данных Oracle приведена в приложении А).

Параметры

целое1

Целое число, представляющее первую единицу времени.

выражение_времени

Выражение времени, которое может принимать следующие формы:

HH:MI:SS[.*n*]

MI:SS[.*n*]

SS[.*n*]

Если применяется совместно с *целое1*, то начальное поле должно иметь тип DAY.

точность_первой_части

Целое, представляющее собой точность первого поля, по умолчанию равно 2.

точность_дробной_части

Целое, представляющее собой точность дробной части поля SECOND, по умолчанию равно 6.

Примеры

Приведенные примеры иллюстрируют применение различных форм синтаксиса функции INTERVAL и показывают получаемые в результате значения (синтаксис функции INTERVAL YEAR TO MONTH описан ниже):

```
INTERVAL '10-4' YEAR(3) TO MONTH
```

10 лет, 4 месяца

```
INTERVAL '10-4' YEAR TO MONTH
```

Ошибка, поскольку точность по умолчанию (2) недостаточна для первого значения.

```
INTERVAL '240' MONTH
```

20 лет

```
INTERVAL '5 4' DAY TO MINUTE
```

5 дней и 4 минуты

```
INTERVAL '5 4' DAY TO HOUR
```

5 дней и 4 часа

```
INTERVAL '3 11:15' DAY TO MINUTE
```

3 дня, 11 часов и 15 минут

```
INTERVAL '20' DAY - INTERVAL '120' HOUR
```

15 дней

INTERVAL YEAR TO MONTH

```
INTERVAL 'целое1 [- целое2]' YEAR | MONTH [точность_первой_части] [TO MONTH]
```

Преобразует целые литералы, представляющие годы, месяцы или годы и месяцы в интервал времени. В этой функции может применяться одна или две единицы измерения времени. Если указаны две единицы измерения, то первая должна быть больше, чем секунда, как в INTERVAL ... DAY TO SECOND. Появилась в Oracle9*i*.

Функции INTERVAL можно применять для выполнения арифметических операций с датами, как показано в разделе «Примеры» для предыдущей функции INTERVAL DAY TO SECOND.

Параметры

целое1 Целое число, представляющее первую единицу времени: YEAR или MONTH.

целое2 Необязательное целое, представляющее собой точность второй необязательной единицы времени MONTH. Разрешен диапазон значений 0–11.

точность_первой_части

Необязательное целое, представляющее собой точность первого поля, по умолчанию равно 2. Если количество цифр превышает это значение, то функция возвращает ошибку.

LAST_DAY

```
LAST_DAY(дата)
```

Возвращает дату, соответствующую последнему дню месяца, к которому относится *дата*.

LOCAL_TIMESTAMP

```
CURRENT_TIMESTAMP [(точность)]
```

Возвращает текущие дату и время для часового пояса сессии в виде значения типа TIMESTAMP. По умолчанию *точность* равна 6. Появилась в Oracle9*i*.

MONTHS_BETWEEN

```
MONTHS_BETWEEN (дата1, дата2)
```

Возвращает количество месяцев, разделяющих даты *дата1* и *дата2*. Если обе даты представляют собой один и тот же день месяца или же обе являются последними днями соответствующих месяцев, то функция возвращает целое значение. В противном случае функция возвращает дробное значение (считается, что месяц состоит из 31 дня).

NEW_TIME

```
NEW_TIME (дата, чп1, чп2)
```

Преобразует значение даты/времени из часового пояса *чп1* в значение даты/времени для часового пояса *чп2*. Часовые пояса представляются идентификаторами, перечень которых приведен в табл. 8.2. Помните, что эти идентификаторы никак не связаны с элементами формата TZD и TZR, которые применяются для часовых поясов в значениях временных меток в Oracle9*i*.

Таблица 8.2. Идентификаторы часовых поясов

Идентификатор	Часовой пояс
AST	Стандартное атлантическое (нью-йоркское) время
ADT	Дневное атлантическое время
BST	Стандартное время по Берингу
BDT	Дневное время по Берингу
CST	Стандартное центральное время
CDT	Дневное центральное время
EST	Восточное стандартное время
EDT	Восточное дневное время
GMT	Гринвичское время
HST	Гавайское стандартное время

Идентификатор	Часовой пояс
HDT	Гавайское дневное время
MST	Стандартное горное время
MDT	Дневное горное время
NST	Стандартное время Ньюфаундленда
PST	Стандартное тихоокеанское время
PDT	Дневное тихоокеанское время
YST	Стандартное время Юкона
YDT	Дневное время Юкона

NEXT_DAY

```
NEXT_DAY (дата, строка)
```

Возвращает дату первого дня недели, заданного параметром *строка*, который следует за датой, заданной параметром *дата*. Составляющая времени в дате сохраняется и возвращается в составе результата. NEXT_DAY всегда выполняет просмотр вперед. Если дата, переданная как параметр, совпадает с искомым днем недели, то NEXT_DAY возвратит дату того же дня следующей недели.

Параметры

строка День недели, при этом *строка* может быть как полным именем (Wednesday), так и сокращением (Wed). Имена дней должны соответствовать текущему значению NLS_DATE_LANGUAGE.

NUMTODS_INTERVAL

```
NUMTODSINTERVAL (n, 'выражение')
```

Преобразует *n* в литерал INTERVAL DAY TO SECOND. Появилась в Oracle9*i*.

Параметры

n Число, подлежащее преобразованию в литерал INTERVAL DAY TO SECOND.

выражение Возвращаемая единица. Допустимы следующие значения: 'DAY', 'HOUR', 'MINUTE' или 'SECOND'.

NUMTOYMINTERVAL

```
NUMTOYMINTERVAL (n, 'выражение')
```

Преобразует *n* в литерал INTERVAL YEAR TO MONTH. Появилась в Oracle9*i*.

Параметры

n Число, подлежащее преобразованию в литерал INTERVAL YEAR TO MONTH.

выражение Возвращаемая единица. Допустимы следующие значения: 'YEAR' или 'MONTH.'

ROUND

```
ROUND (дата [,dfmt])
```

Округляет *дату* до ближайшей единицы даты/времени, определяемой *dfmt*. При округлении может оказаться, что новая дата больше начальной. Если вас не устраивает такой вариант, вместо округления применяйте функцию TRUNC.

Общие ключевые слова и инструкции: *dfmt*.

SESSION_TIMEZONE

```
SESSIONTIMEZONE
```

Возвращает значение часового пояса для текущей сессии. Это может быть как смещение часового пояса, так и его название, в зависимости от того, каким образом значение часового пояса было указано в последней команде ALTER SESSION. Появилась в Oracle9*i*.

SYS_EXTRACT_UTC

```
SYS_EXTRACT_UTC (выражение_даты_времени)
```

Возвращает время UTC (или GMT) из *выражения_даты_времени*. UTC – всеобщее скоординированное время (Coordinated Universal Time), GMT – Гринвичское время (Greenwich Mean Time). Появилась в Oracle9*i*.

Параметр

выражение_даты_времени

Выражение, которое оценивается как дата/время со смещением часового пояса.

SYSDATE

```
SYSDATE
```

Возвращает текущую дату и время, включая часы, минуты и секунды.

SYSTIMESTAMP

```
SYSTIMESTAMP
```

Возвращает системную дату и время, включая часовой пояс базы данных, в виде значения типа TIMESTAMP WITH TIME ZONE. Появилась в Oracle9*i*.

TO_DSINTERVAL

```
TO_DSINTERVAL (строка['NLS_NUMERIC_CHARACTERS="dg"'])
```

Преобразует *строку* в значение типа INTERVAL DAY_TO SECOND (см. описание функций INTERVAL). Появилась в Oracle9*i*.

Параметры

строка Строка, содержащая символы, которые должны быть преобразованы.

d Символ десятичного разделителя.

g Символ группового разделителя.

TO_TIMESTAMP

```
TO_TIMESTAMP (строка[,dfmt['NLS_NUMERIC_CHARACTERS="dg"']])
```

Преобразует *строку* в значение типа TIMESTAMP. Если *dfmt* не указано, то *строка* должна соответствовать формату по умолчанию для типа TIMESTAMP. Появилась в Oracle9*i*.

Параметры

строка Строка, содержащая символы, которые должны быть преобразованы.

d Символ десятичного разделителя.

g Символ группового разделителя.

Общие ключевые слова и инструкции: *dfmt*.

TO_TIMESTAMP_TZ

```
TO_TIMESTAMP_TZ (строка[,dfmt['NLS_NUMERIC_ CHARACTERS="dg"']])
```

Преобразует *строку* в значение типа TIMESTAMP WITH TIME ZONE. Если *dfmt* не указано, то *строка* должна соответствовать формату по умолчанию для типа TIMESTAMP WITH TIME ZONE. Появилась в Oracle9*i*.

Параметры

строка Строка, содержащая символы, которые должны быть преобразованы.

d Символ десятичного разделителя.

g Символ группового разделителя.

Общие ключевые слова и инструкции: *dfmt*.

TO_YMINTERVAL

```
TO_YMINTERVAL (строка)
```

Преобразует *строку* в значение типа INTERVAL YEAR TO MONTH (см. описание функций INTERVAL). Появилась в Oracle9*i*.

TRUNC

```
TRUNC (дата [,dfmt])
```

Возвращает значение даты/времени, усеченное до указанных единиц. TRUNC применяется в тех случаях, когда необходимо работать только с датами без составляющей времени.

Общие ключевые слова и инструкции: *dfmt*.

TZ_OFFSET

```
TZ_OFFSET ({'имя_часового_пояса' | '{+ | -}hh:mi' |
  SESSIONTIMEZONE | DBTIMEZONE})
```

Возвращает смещение часового пояса. Появилась в Oracle9*i*.

Параметры

имя_часового_пояса

Имя часового пояса.

+

Означает, что смещение будет положительным, то есть в будущее.

-

Означает, что смещение будет отрицательным, то есть в прошлое.

hh

Часы смещения.

mi

Минуты смещения.

SESSIONTIMEZONE

Указывает, что смещение будет значением часового пояса для текущей сессии.

DBTIMEZONE

Указывает, что смещение будет значением часового пояса для базы данных.

Функции преобразования

Функции данного раздела позволяют преобразовать значение из одного типа данных в другой.

ASCIISTR

```
ASCIISTR ('символьная_строка')
```

Преобразует *символьную_строку* в ASCII-строку для набора символов БД. Появилась в Oracle9*i*.

BIN_TO_NUM

```
BIN_TO_NUM (выражение[,выражение...])
```

Преобразует одно или несколько выражений, которые вычисляются как двоичные, в значение типа NUMBER. Появилась в Oracle9*i*.

CAST

```
CAST {выражение | (подзапрос) | MULTISET (подзапрос)} AS новый_тип)
```

Преобразует значение встроенного типа данных или типа коллекции в значение типа *новый_тип*.

Параметры

подзапрос

Подзапрос, который должен возвращать единственное значение типа коллекции или встроенного типа.

MULTISET

Указывает, что сервер Oracle должен принять результирующее множество подзапроса и возвратить значение коллекции.

новый_тип

Имя типа коллекции или встроенного типа.

CHARTOROWID

```
CHARTOROWID (строка)
```

Преобразует строковое значение *строка* в значение ROWID.

COMPOSE

```
COMPOSE ('строка')
```

Преобразует *строку* любого типа данных в строку Unicode в ее полной нормализованной форме для того же набора символов, что и исходная строка. Появилась в Oracle9*i*.

CONVERT

```
CONVERT (строка, конечный_набор_символов[, исходный_набор_символов])
```

Преобразует символьную строку из одного набора символов в другой.

Параметры

строка

Преобразуемая символьная строка.

конечный_набор_символов

Имя конечного набора символов.

исходный_набор_символов

Имя исходного набора символов. По умолчанию – набор символов БД.

DECOMPOSE

```
DECOMPOSE ('строка')
```

Преобразует строку любого типа данных в строку Unicode после канонического разложения в том же наборе символов, что и строка ввода. Появилась в Oracle9*i*.

HEXTORAW

```
HEXTORAW (строка)
```

Преобразует шестнадцатеричные цифры *строки* в RAW значение байтов.

NUMTODS_INTERVAL

```
NUMTODSINTERVAL (n,{'DAY' | 'HOUR' | 'MINUTE' | SECOND'})
```

Преобразует *n* в литерал INTERVAL DAY TO SECOND (см. описание функций INTERVAL). Появилась в Oracle9*i*.

Параметры

n	Преобразуемое числовое значение.
DAY	Означает, что *n* должно восприниматься как *n* дней.
HOUR	Означает, что *n* должно восприниматься как *n* часов.
MINUTE	Означает, что *n* должно восприниматься как *n* минут.
SECOND	Означает, что *n* должно восприниматься как *n* секунд.

NUMTOYM_INTERVAL

```
NUMTOYMINTERVAL (n,{'YEAR' | 'MONTH'})
```

Преобразует *n* в литерал INTERVAL YEAR TO MONTH (см. описание функций INTERVAL). Появилась в Oracle9*i*.

Параметры

n	Преобразуемое числовое значение.
YEAR	Означает, что *n* должно восприниматься как *n* лет.
MONTH	Означает, что *n* должно восприниматься как *n* месяцев.

RAWTOHEX

```
RAWTOHEX (raw)
```

Преобразует необработанные данные *raw* в строку VARCHAR2, содержащую текстовое представление шестнадцатеричного числа, где каждое двузначное шестнадцатеричное число соответствует одному байту *raw*.

RAWTONHEX

```
RAWTONHEX (raw)
```

Преобразует необработанные данные *raw* в строку NVARCHAR2, содержащую текстовое представление шестнадцатеричного числа, где каждое двузначное шестнадцатеричное число соответствует одному байту *raw*. Появилась в Oracle9*i*.

ROWIDTOCHAR

```
ROWIDTOCHAR (rowid)
```

Преобразует значение типа ROWID в символьную строку. Значения ROWID также неявно преобразуются в символьные строки, когда их выбирают из таблиц средствами SQL*Plus. Появилась в Oracle9*i*.

ROWIDTONCHAR

```
ROWIDTONCHAR (rowid)
```

Преобразует значение типа ROWID в строку NCHAR. Появилась в Oracle9*i*.

TO_CHAR (символ)

```
TO_CHAR (значение)
```

Преобразует *значение* типа NCHAR, NVARCHAR2, CLOB или NCLOB в набор символов БД. Появилась в Oracle9*i*.

TO_CHAR (дата/время)

```
TO_CHAR (дата [, dfmt [,'nlsparams']])
```

Преобразует значение *дата* в символьное представление на основе *dfmt*.

Параметр

дата Указывает значение даты/времени (тип DATE).

Общие ключевые слова и инструкции: *dfmt*, *nlsparams*.

TO_CHAR (число)

```
TO_CHAR (n [, fmt [,'nlsparams']])
```

Преобразует числовое значение в его символьное представление.

Параметр

n Преобразуемое числовое значение.

Общие ключевые слова и инструкции: *fmt*, *nlsparams*.

TO_CLOB

```
TO_CLOB ({столбец_LOB | символьная_строка})
```

Преобразует NCLOB в *столбце_LOB* или символьной строке *символьная_строка* в значение CLOB. Появилась в Oracle9*i*.

Параметры

столбец_LOB

Имя столбца LOB.

символьная_строка

Имя символьной строки.

TO_DATE

```
TO_DATE (строка [, dfmt [,'nlsdateparam']])
```

Преобразует *строку* в значение типа DATE.

Параметры

строка Символьное представление значения даты/времени, которое должно быть преобразовано.

Общие ключевые слова и инструкции: *dfmt*, *nlsdateparam*.

TO_DSINTERVAL

```
TO_DSINTERVAL (строка['NLS_NUMERIC_CHARACTERS="dg"'])
```

Преобразует *строку* в тип INTERVAL DAY TO SECOND (см. описание функций INTERVAL). Появилась в Oracle9*i*.

Параметры

строка Строка, содержащая символы, подлежащие преобразованию.

d Символ десятичного разделителя.

g Символ группового разделителя.

TO_LOB

```
TO_LOB (столбец_LONG)
```

Преобразует значение *столбец_LONG* типа LONG или LONG RAW в значение типа CLOB, BLOB или NCLOB. Функция может применяться только в подзапросе команды INSERT … SELECT FROM, используемой для наполнения столбца LOB. Значения LONG преобразуются в значения типа CLOB или NCLOB в зависимости от типа данных столбца назначения. Значения LONG RAW преобразуются в BLOB.

TO_MULTI_BYTE

```
TO_MULTI_BYTE (строка)
```

Преобразует однобайтные символы в *строке* в их многобайтные эквиваленты.

TO_NCHAR (символ)

```
TO_NCHAR (значение [, dfmt [,'nlsparams']])
```

Преобразует *значение*, которое может быть символьной строкой, значением типа CLOB или NCLOB, в символьное представление этого значения в национальном наборе символов.

Общие ключевые слова и инструкции: *dfmt*, *nlsparams*.

TO_NCHAR (дата/время)

```
TO_NCHAR ({дата_время | интервал}[, dfmt [,'nlsparams']])
```

Преобразует значение *дата_время* или *интервал* из набора символов БД в национальный набор символов.

Параметры

дата_время

Значение даты/времени.

интервал

Значение интервала.

Общие ключевые слова и инструкции: *dfmt*, *nlsparams*.

TO_NCHAR (число)

```
TO_NCHAR (значение [, fmt [, 'nlsparams']])
```

Преобразует числовое *значение* в национальный набор символов. Появилась в Oracle9*i*.

Общие ключевые слова и инструкции: *fmt*, *nlsparams*.

TO_NCLOB

```
TO_NCLOB (значение)
```

Преобразует *значение* в тип NCLOB, при этом *значение* может быть столбцом LOB, CHAR, VARCHAR2, NCHAR, NVARCHAR2, CLOB или NCLOB. Появилась в Oracle9*i*.

TO_NUMBER

```
TO_NUMBER (строка [, fmt[, 'nlsparams']])
```

Преобразует *строку* в значение типа NUMBER.

Параметры

строка Строка, содержащая символьное представление преобразуемого значения.

Общие ключевые слова и инструкции: *fmt*, *nlsparams*.

TO_SINGLE_BYTE

```
TO_SINGLE_BYTE (строка)
```

Преобразует многобайтные символы в *строке* в их однобайтные эквиваленты.

TO_YMINTERVAL

```
TO_YMINTERVAL (символьная_строка)
```

Преобразует *символьную_строку* типа CHAR, VARCHAR2, NCHAR или NVARCHAR2 в значение типа INTERVAL YEAR TO MONTH (см. описание функций INTERVAL). Появилась в Oracle9*i*.

TRANSLATE ... USING

```
TRANSLATE (текст USING {CHAR_CS | NCHAR_CS})
```

Преобразует *текст* в набор символов БД или в национальный набор символов.

Параметры

текст Строка, которую вы хотите преобразовать.

CHAR_CS Приводит к преобразованию строки из национального набора символов в набор символов БД. Результат возвращается в формате VARCHAR2.

NCHAR_CS Приводит к преобразованию строки из набора символов БД в национальный набор символов. Результат возвращается в формате VARCHAR2.

UNISTR

```
UNISTR ('строка')
```

Преобразует *строку* любого набора символов в Unicode для набора символов Unicode БД. Кодовые символы UCS2 представлены как числа, которым предшествует обратная косая черта (\). Появилась в Oracle9*i*.

Объектные функции

Следующие функции используются для работы с *указателями* (reference *pointers* – REF) на объектные типы.

DEREF

```
DEREF (выражение)
```

Возвращает указатель объекта для *выражения*, которое должно возвращать указатель объекта. По умолчанию сервер Oracle возвращает идентификатор объекта для объекта в запросе. Появилась в Oracle8*i*.

MAKE_REF

```
MAKE REF (таблица | представление , ключ [, ключ . . .])
```

Создает указатель на строку объектного *представления* или объектной *таблицы*. В основе идентификатора объекта для строки должен лежать первичный ключ (*ключ*). Появилась в Oracle8*i*.

REF

```
REF (переменная_корреляции)
```

Возвращает указатель для экземпляра объекта в объектной таблице, псевдонимом которой является *переменная_корреляции*. Появилась в Oracle8*i*.

REFTOHEX

```
REFTOHEX (выражение)
```

Преобразует указатель *выражения* в его шестнадцатеричный эквивалент. Появилась в Oracle8*i*.

VALUE

```
VALUE (переменная_корреляции)
```

Возвращает экземпляры объекта, хранящегося в объектной таблице, псевдонимом которой является *переменная_корреляции*. Появилась в Oracle8*i*.

Функции для работы с XML

Обработкой XML-документов занимаются следующие функции:

EXISTSNODE

```
EXISTSNODE (экземпляр_XMLtype, строка_XMLпути)
```

Определяет наличие узлов при обходе документа *экземпляра_XMLtype*, основываясь на указанном пути. Функция возвращает 0, если узлов не осталось; в противном случае возвращается положительное число. Появилась в Oracle9*i*.

Параметры

экземпляр_XMLtype

Имя экземпляра XMLtype, содержащего документ XML.

строка_XMLпути

Строка VARCHAR2, содержащая XML-путь.

EXTRACT (XML)

```
EXTRACT (экземпляр_XMLtype, строка_XMLпути)
```

Аналогична функции EXISTSNODE. Извлекает экземпляр XMLType *строки_XMLпути* из *экземпляра_XMLtype*. Появилась в Oracle9*i*.

Параметры

экземпляр_XMLtype

Имя экземпляра XMLtype

строка_XMLпути

Строка VARCHAR2, содержащая XML-путь.

EXTRACTVALUE

```
EXTRACTVALUE (экземпляр_XMLtype, строка_XMLпути)
```

Извлекает скалярное значение *строки_XMLпути* из *экземпляра_XMLtype*. Появилась в Oracle9*i*.

Параметры

экземпляр_XMLtype

Имя экземпляра XMLtype.

строка_XMLпути

Строка VARCHAR2, содержащая XML-путь.

SYS_XMLAGG

```
SYS_XMLAGG (выражение [xmlfmt])
```

Возвращает единый XML-документ, который представляет собой совокупность всех XML-документов, представленных *выражением*. Функция добавляет новый объемлющий элемент с именем по умолчанию ROWSET. Если указан аргумент *xmlfmt* (экземпляр объекта SYS.XMLGenFormatType), то XML-документ форматируется соответствующим образом. Появилась в Oracle9*i*.

SYS_XMLGEN

```
SYS_XMLGEN (выражение [xmlfmt])
```

Возвращает экземпляр типа SYS.XMLType, содержащий XML-документ из *выражения*, которое оценивается как определенная строка и столбец БД. Если указан аргумент *xmlfmt* (экземпляр объекта SYS.XMLGenFormatType), то XML-документ форматируется соответствующим образом. Появилась в Oracle9*i*.

UPDATEXML

```
UPDATEXML (экземпляр_XMLtype, строка_XMLпути, выражение_для_значения)
```

Возвращает экземпляр XMLType типа *экземпляр_XMLtype*, найденный в строке *строка_XMLпути*, имеющей *выражение_для_значения*.

Параметры

экземпляр_XMLtype

Имя экземпляра XMLtype.

строка_XMLпути

Строка VARCHAR2, содержащая XML-путь.

XMLAGG

```
XMLAGG (экземпляр_XMLtype [ORDER BY список_сортировки] )
```

Возвращает сводный документ XML для набора фрагментов XML. XMLAGG аналогична SYS_XMLAGG, только XMLAGG возвращает набор узлов. Однако она не допускает форматирования при помощи объекта XMLFormat и не заключает вывод в тег элемента, как это делает SYS_XMLAGG. Появилась в Oracle9*i*.

Параметры

экземпляр_XMLtype

Имя экземпляра XMLtype

список_сортировки

Список сортируемых значений.

XMLCOLATTVAL

```
XMLCOLATTVAL (выражение_для_значения [AS псевдоним_столбца] )
```

Создает фрагмент XML для *выражения_для_значения* с именем столбца и распространяет получившийся XML так, что все фрагменты имеют имя «столбец» с атрибутом «имя». Если указан *псевдоним_столбца*, то он заменяет имя столбца. Появилась в Oracle9*i*.

XMLCONCAT

```
XMLCONCAT(экземпляр_XMLtype[,экземпляр_XMLtype. . .] )
```

Возвращает сцепленные последовательности из *экземпляр_XMLtype*s. Появилась в Oracle9*i*.

Параметры

экземпляр_XMLtype

Имя экземпляра XMLtype

XMLELEMENT

```
XMLELEMENT ([NAME] идентификатор
    [, XMLATTRIBUTES(выражение_для_значения [AS псевдоним_столбца]
    [, выражение_для_значения [AS псевдоним_столбца]. . .]]
    [,выражение_для_значения[, выражение_для_значения . . .])])
```

Возвращает экземпляр XMLType с именем *идентификатор*, с атрибутами, перечисленными в инструкции XMLATTRIBUTES, и значение *выражения_для_значения* (*псевдоним_столбца* – это необязательное альтернативное имя для значения). Обычно применяется для форматирования значений, возвращаемых из столбцов во фрагменты XML. Появилась в Oracle9*i*.

XMLFOREST

```
XMLFOREST (выражение_для_значения [AS псевдоним_столбца] )
```

Преобразует каждое *выражение_для_значения* в формат XML и возвращает фрагмент XML, соединяя выражения (*псевдоним_столбца* – это необязательное альтернативное имя для значения). Появилась в Oracle9*i*.

XMLSEQUENCE

```
XMLSEQUENCE (экземпляр_XMLtype | REF_курсор [, xmlfmt] )
```

Возвращает массив VARRAY значений XMLTypes. Если указан аргумент *xmlfmt* (экземпляр объекта SYS.XMLGenFormatType), то XML-документ форматируется соответствующим образом. Появилась в Oracle9*i*.

Параметры

экземпляр_XMLtype

С этим аргументом функция возвращает массив VARRAY узлов самого верхнего уровня в XMLType.

REF_курсор

Экземпляр указателя REF CURSOR с необязательным экземпляром XMLFormat. С этим аргументом функция возвращает документ XML типа XMLSequence для каждой строки курсора.

XMLTRANSFORM

```
XMLTRANSFORM (экземпляр_XMLtype, экземпляр_XSLType)
```

Преобразует *экземпляр_XMLtype* при помощи *экземпляра_XSLType*. Появилась в Oracle9*i*.

Параметры

экземпляр_XMLtype

С этим аргументом функция возвращает массив VARRAY узлов самого верхнего уровня в XMLType.

Другие функции

Оставшиеся функции не попадают ни в одну из категорий, приведенных выше в этой главе.

BFILENAME

```
BFILENAME (каталог, имя_ файла)
```

Возвращает *указатель* (*locator*) BFILE на указанный файл.

Параметры

каталог Каталог (созданный ранее при помощи команды CREATE DIRECTORY), содержащий файл.

имя_файла Имя файла, на который должен указывать BFILE.

COALESCE

```
COALESCE (выражение[,выражение...])
```

Возвращает первое выражение в списке аргументов, значение которого отличается от NULL. Если значения всех выражений равны NULL, то функция возвращает NULL. Появилась в Oracle9*i*.

DECODE

```
DECODE (выражение, значение, результат [,значение, результат...] [,умолчание])
```

Предоставляет возможности встроенной команды IF. DECODE получает входное значение и список, в который может входить до 255 пар значение/результат. DECODE ищет пару, в которой *значение* совпадает со входным значением. Когда соответствующая пара найдена, DECODE возвращает *результат* из этой пары в качестве ре-

зультата функции. Если соответствующая пара не найдена, то DECODE возвращает *умолчание*.

Типы данных определяются первой парой *значение, результат*. Входное выражение и все *значения* преобразуются к типу данных первого *значения*. Возвращаемое значение преобразуется в тип данных первого значения *результата*.

Параметры

выражение Входное значение. DECODE сравнивает его с последующими *значениями* в поиске нужной пары значение/результат.

значение Сравниваемая со входным выражением часть пары значение/результат.

результат Результирующая часть пары значение/результат.

умолчание Необязательный результат по умолчанию, который функция DECODE возвращает, если ни одно из *значений* не совпадает со входным выражением.

DEPTH

```
DEPTH (значение_корреляции)
```

Возвращает количество уровней пути, задаваемого условиями UNDER_PATH или EQUALS_PATH. По аргументу *значение_корреляции* выполняется сопоставление функции с основным условием, если команда содержит несколько таких условий. Появилась в Oracle9*i*.

DUMP

```
DUMP (выражение [,формат_возврата [,начальная_позиция [,длина]]])
```

Возвращает значение типа VARCHAR2, показывающее тип и внутреннее представление данных, хранящихся в столбце, или данных, возвращенных выражением.

Параметры

выражение

Данные, сохраняемые в дамп. Это может быть название столбца или корректное выражение SQL.

формат_возврата

Формат, в котором будут возвращены данные, для которых выполнен дамп. Выберите одно из следующих значений:

8 Восьмеричная система счисления.

10 Десятичная система счисления (по умолчанию).

16 Шестнадцатеричная система счисления.

17 Выводить результат в символьном формате.

Если добавить к спецификатору формата 1000, то функция DUMP возвратит и имя набора символов.

начальная_позиция

Указывает байт данных, с которого начинается выполнение дампа. По умолчанию – первый байт данных.

длина Указывает количество байт для дампа. По умолчанию выполняется дамп всех данных.

EMPTY_BLOB

```
EMPTY_BLOB()
```

Возвращает пустой указатель BLOB, который затем можно использовать для инициализации столбца BLOB. В вызов функции необходимо включать скобки.

EMPTY_CLOB

```
EMPTY_CLOB()
```

Возвращает пустой указатель CLOB, который затем можно использовать для инициализации столбца CLOB. В вызов функции необходимо включать скобки.

NLS_CHARSET_DECL_LEN

```
NLS_CHARSET_DECL_LEN (байтовый_размер, номер_набора_символов)
```

Возвращает объявленную ширину столбца NCHAR (в терминах количества символов) на основе *байтового_размера*.

Параметры

байтовый_размер

Размер столбца NCHAR в байтах.

номер_набора_символов

Номер, идентифицирующий набор символов NLS для столбца. Функция NLS_CHARSET_ID применяется для получения номера набора символов по его названию.

NLS_CHARSET_ID

```
NLS_CHARSET_ID (текст)
```

Возвращает идентификатор, соответствующий имени набора символов NLS, заданному параметром *текст*.

NLS_CHARSET_NAME

```
NLS_CHARSET_NAME (n)
```

Возвращает имя, соответствующее набору символов NLS с идентификатором *n*.

NULLIF

```
NULLIF (выражение1, выражение2)
```

Сравнивает *выражение1* с *выражением2*. Если выражения равны, то функция возвращает NULL; в противном случае она возвращает *выражение1* (*выражение1* не может быть установлено в литеральный NULL). Появилась в Oracle9*i*.

NVL

```
NVL (выражение1, выражение2)
```

Возвращает альтернативное значение, которое будет использоваться, если входное значение – NULL. NVL возвращает *выражение2*, если *выражение1* – это NULL; в противном случае функция просто возвращает *выражение1*.

NVL2

```
NVL2 (выражение1, выражение2, выражение3)
```

Возвращает или *выражение2*, или *выражение3*. Если *выражение1* равно NULL, то функция возвращает *выражение2*; иначе – *выражение3*; *выражение2* и *выражение3* не могут относиться к типу LONG.

PATH

```
PATH (значение_корреляции)
```

Возвращает относительный путь к ресурсу, определяемому условием UNDER_PATH или EQUALS_PATH. Аргумент *значение_корреляции* предназначен для сопоставления функции с основным условием, если команда содержит несколько таких условий. Появилась в Oracle9*i*.

SYS_CONNECT_BY_PATH

```
SYS_CONNECT_BY_PATH (столбец, символ)
```

Возвращает путь значения столбца от корня к узлу, при этом значения столбца разделяются *символом* для каждой строки, возвращаемой CONNECT BY. Появилась в Oracle9*i*.

SYS_CONTEXT

```
SYS_CONTEXT (пространство_имен, имя_атрибута [,длина])
```

Возвращает значение атрибута в пространстве имен контекста приложения.

Параметры

пространство_имен

Имя пространства имен, ранее созданного при помощи команды CREATE CONTEXT. Также можно указать пространство по умолчанию USERENV.

имя_атрибута

Имя атрибута пространства имен, возвращаемого данной функцией. Различные предопределенные атрибуты, доступные для пространства по умолчанию USERENV, перечислены в табл. 8.3.

длина

Максимально возможная длина возвращаемого значения атрибута. Этот необязательный параметр появился только в Oracle 8.1.6 и действует только для атрибута AUTHENTICATION_DATA; кроме того, он не может превышать 4000.

Таблица 8.3. Предопределенные атрибуты в пространстве имен USERENV

Имя атрибута	Описание	Максимальная длина	Доступно начиная с версии
AUTHENTICATION_DATA	Данные, необходимые для аутентификации пользователя. Если аутентификация выполняется в соответствии с сертификатом X.503, то содержимое сертификата возвращается в шестнадцатеричном формате.	256–4000	8.1.6
AUTHENTICATION_TYPE	Указывает способ аутентификации пользователя: DATABASE: По имени и паролю пользователя. OS: Аутентификация средствами операционной системы. NETWORK: Аутентификация по сетевому протоколу или при помощи Advanced Networking Option (ANO) PROXY: Аутентификация проводится через Oracle Call Interface (OCI)	30	8.1.6
BG_JOB_ID	Если текущая сессия была создана фоновым процессом Oracle, то это идентификатор задания, в противном случае NULL.	30	8.1.6
CLIENT_INFO	Возвращает до 64 байт информации о сессии пользователя, которая может быть сохранена средствами пакета DBMS_APPLICATION_INFO.	64	8.1.6
CURRENT_SCHEMA	Имя текущей схемы.	30	8.1.5
CURRENT_SCHEMAID	Идентификатор, сопоставленный текущей схеме.	30	8.1.5
CURRENT_USER	Текущее имя пользователя. Если вызывалась хранимая процедура, то данное имя может не совпадать с регистрационным именем, которое возвращается SESSION_USER.	30	8.1.5
CURRENT_USERID	Идентификатор, сопоставленный текущему пользователю.	30	8.1.5
DB_DOMAIN	Домен БД (указанный в параметре инициализации DB_DOMAIN).	256	8.1.6
DB_NAME	Имя БД (указанное в параметре инициализации DB_NAME).	30	8.1.6

Имя атрибута	Описание	Максимальная длина	Доступно начиная с версии
ENTRYID	Идентификатор записи аудита. Этот атрибут не действует для распределенных команд SQL, ему также необходимо, чтобы параметр инициализации AUDIT_TRAIL был установлен ? TRUE.	30	8.1.6
EXTERNAL_NAME	Внешнее имя пользователя БД. Для пользователей, аутентификация которых производится с применением SSL, это отличительное имя (Distinguished Name – DN) из сертификата v.503.	256	8.1.6
FG_JOB_ID	Идентификатор задания сессии для сессий, созданных приоритетным клиентским процессом.	30	8.1.6
HOST	Имя машины, с которой клиент устанавливает соединение.	54	8.1.6
INSTANCE	Номер, идентифицирующий экземпляр, к которому вы подключены в настоящий момент.	30	8.1.6
IP_ADDRESS	IP-адрес пользователя. Относится только к TCP/IP-соединениям.	30	8.1.5
ISDBA	TRUE или FALSE в зависимости от того, разрешена ли роль ISDBA.	30	8.1.6
LANG	Аббревиатура ISO для названия текущего языка.	62	8.1.6
LANGUAGE	Текущие значения языка, территории и имя набора символов БД.	52	8.1.6
NETWORK_PROTOCOL	Имя сетевого протокола, применяемого для соединения.	256	8.1.6
NLS_CALENDAR	Текущее имя календаря NLS.	62	8.1.5
NLS_CURRENCY	Текущее обозначение валюты NLS.	62	8.1.5
NLS_DATE_FORMAT	Текущий формат даты NLS.	62	8.1.5
NLS_DATE_LANGUAGE	Текущий язык даты NLS.	62	8.1.5
NLS_SORT	Текущая последовательность сортировки NLS.	62	8.1.5
NLS_TERRITORY	Текущее имя территории NLS.	62	8.1.5
OS_USER	Имя пользователя ОС для клиентского процесса, инициировавшего сеанс связи с БД,	30	8.1.6
PROXY_USER	Имя пользователя, который открыл текущую сессию от имени текущего пользователя сессии.	30	8.1.6

Таблица 8.3 (продолжение)

Имя атрибута	Описание	Максимальная длина	Доступно начиная с версии
PROXY_USERID	Идентификатор пользователя, открывшего текущую сессию от имени текущего пользователя сессии.	30	8.1.6
SESSION_USER	Имя, с которым зарегистрировался текущий пользователь. Не меняется, даже если вызывается хранимая процедура, принадлежащая другому пользователю.	30	8.1.5
SESSION_USERID	Идентификатор, сопоставленный пользователю сессии.	30	8.1.5
SESSIONID	Идентификатор сессии аудита. Не действует для распределенных команд SQL	30	8.1.6
TERMINAL	Идентификатор ОС для клиента текущей сессии. В распределенной среде может применяться только для удаленных команд SELECT, при этом будет возвращен идентификатор локальной сессии.	10	8.1.6

SYS_DBURIGEN

```
SYS_DBURIGEN ({столбец | атрибут}[rowid]
  [,{столбец | атрибут}[rowid]...][,'текст()'])
```

Возвращает URL типа данных DBUriType для объекта строки или столбца, который можно использовать для извлечения документа XML из БД. Появилась в Oracle9*i*.

SYS_EXTRACT_UTC

```
SYS_EXTRACT_UTC(дата_время_с_часовым_поясом)
```

Возвращает UTC/GMT для значения даты/времени со смещением часового пояса.

SYS_GUID

```
SYS_GUID()
```

Возвращает 16-байтное значение типа RAW, которое может выступать в качестве глобального уникального идентификатора. Для большей части платформ значение представляет собой комбинацию идентификатора хоста, идентификатора процесса или потока и порядкового номера. В вызов функции необходимо включать скобки.

SYS_TYPEID

```
SYS_TYPEID(значение_объектного_типа)
```

Возвращает идентификатор типа для наиболее специфичного типа значения *_объектного_типа*.

UID

```
UID
```

Возвращает целое значение, уникально идентифицирующее текущего пользователя БД. Значение берется из столбца USER# представления V$SESSION.

USER

```
USER
```

Возвращает имя текущего пользователя. Обычно USER возвращает имя пользователя, указанное при регистрации в БД. Если функция вызывается из хранимой процедуры или функции, то USER возвращает имя владельца процедуры или функции. Если вызывается из триггера, то возвращает регистрационное имя пользователя.

USERENV

```
USERENV (параметр)
```

Возвращает информацию о текущем пользователе в виде значения VARCHAR2. Начиная с версии Oracle 8.1.6 SYS_CONTEXT также может применяться для извлечения этих значений окружения пользователя. Для ENTRYID и SESSIONID необходимо установить в TRUE параметр инициализации AUDIT_TRAIL; кроме того, эти переменные не могут использоваться в распределенной среде.

Параметры

параметр Возвращает одно из значений табл. 8.4.

Таблица 8.4. Значения параметра USERENV

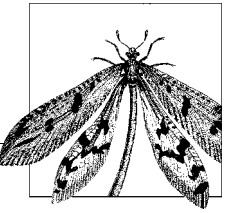
Параметр	Описание
ENTRYID	Возвращает идентификатор записи аудита.
INSTANCE	Возвращает идентификатор экземпляра.
ISDBA	Возвращает логическое значение, указывающее, разрешена ли роль ISDBA.
LANG	Возвращает аббревиатуру текущего языка ISO.
LANGUAGE	Возвращает текущий язык пользователя и параметры настройки территории.
SESSIONID	Возвращает идентификатор сессии аудита.
TERMINAL	Возвращает терминальный идентификатор операционной системы для текущей сессии.

VSIZE

```
VSIZE (выражение)
```

Возвращает размер в байтах внутреннего представления значения, определяемого *выражением*.

9
PL/SQL



Язык PL/SQL (Procedural Language extensions to the Structured Query Language – процедурное расширение структурированного языка запросов) – это хорошо структурированный, удобный и доступный язык, тесно интегрированный с базой данных Oracle. PL/SQL стал первым языком, получившим поддержку в БД Oracle, и до сих пор остается наиболее популярным средством создания программных модулей БД, таких как процедуры, функции, пакеты и триггеры.

Эта глава содержит краткое описание основных понятий и синтаксиса PL/SQL. Описания операторов PL/SQL, насколько это возможно, приведены в соответствии с блочной структурой кода PL/SQL. (Программы на PL/SQL обычно состоят из четырех секций: секции заголовка, секции объявлений, секции выполнения и секции обработки исключений.) Так, например, в разделе «Секция выполнения» вы найдете описание синтаксиса и понятий, с которыми вам придется иметь дело в этой секции. Если некоторый тип данных, такой как курсор, должен быть объявлен до использования, то правила его объявления находятся в разделе «Секция объявлений», а правила использования – в разделе «Секция выполнения». Некоторые понятия, такие как пакеты, триггеры и внешние процедуры, рассмотрены в конце главы в соответствующих разделах.

Этот мощный и выразительный язык достоин более глубокого изучения. Его описанию и применению посвящена превосходная книга Стивена Фейерштейна (Steven Feuerstein) и Билла Прибыла (Bill Pribyl) «Oracle PL/SQL Programming», вышедшая в издательстве «O'Reilly & Associates».[1]

Основы PL/SQL

В этом разделе рассмотрены основные элементы PL/SQL: набор символов, идентификаторы, литералы, разделители, комментарии и структура блоков.

Набор символов

В языке PL/SQL используются следующие буквы, цифры, специальные символы и символы-разделители:

---

[1] Фейерштейн С., Прибыл Б. «Oracle PL/SQL для профессионалов», 3-е издание. – Пер. с англ. – СПб: Питер, 2003.

- Буквы: A–Z, a–z
- Цифры: 0–9
- Специальные символы: ~ ! @ # $ % & * ( ) _ - + = | [ ] { } : ; " ' < > , . ? / ^
- Пробельные символы: пробел, табуляция, возврат каретки, новая строка

Перечисленные символы могут присутствовать в идентификаторах, литералах и разделителях.

Идентификаторы

Идентификаторы – это имена объектов PL/SQL, включая константы, скалярные переменные, составные переменные (записи или коллекции), исключения, процедуры, функции, пакеты, типы, курсоры, зарезервированные слова и метки. Идентификаторы обладают следующими характеристиками:

- Могут состоять не более чем из 30 символов
- Не могут содержать пробельные символы (пробел, табуляцию, возврат каретки, новую строку)
- Должны начинаться с буквы
- Могут содержать знаки знак доллара ($), подчеркивание (_) или знак диеза (#)
- Не чувствительны к регистру

Если идентификатор заключен в двойные кавычки, то в силе остается только последнее из этих правил.

Литералы

Литералы – это конкретные значения, не представленные идентификаторами. Существует несколько различных типов литералов.

Числовые, строковые и логические литералы

Большинство литералов в PL/SQL представляют собой числа (например, 29 или 6.001), строки («Rick» или «Это интересная книга») или логические значения (TRUE или FALSE). Они также могут иметь неопределенное значение NULL. Литералы не представляются идентификаторами. Не существует литералов для составных типов, т. к. они существуют только во внутреннем представлении. В отличие от идентификаторов, литералы чувствительны к регистру. Для того чтобы поместить одинарную кавычку в строковый литерал, расположите две одинарные кавычки друг за другом.

Литералы интервалов дат и времени

В Oracle9*i* были введены два новых типа данных: интервалы дат и времени INTERVAL YEAR TO MONTH и INTERVAL DAY TO SECOND. Они представляют временные интервалы, выраженные в годах и месяцах либо в днях, часах, минутах, секундах и долях секунд. Литералы этого типа имеют форму INTERVAL YEAR (*точность_года*) TO MONTH или INTERVAL DAY (*точность_дня*) TO SECOND (*точность_долей_секунды*) (более подробные пояснения и примеры приведены в приложении А).

Разделители

Разделители (*delimiters*) – это символы или последовательности символов, имеющие специальное значение в PL/SQL. Имеющиеся в PL/SQL разделители перечислены табл. 9.1.

Таблица 9.1. Разделители PL/SQL

Разделитель	Описание
;	Ограничитель (для команд и объявлений)
+ - * / ** =	Арифметические операторы
<> != ^= ~=	Операторы неравенства
< > <= >= <>	Операторы сравнения
‖	Оператор конкатенации
:=	Оператор присваивания
( )	Ограничители выражения или списка
,	Запятая. Разделитель элементов
'	Одинарная кавычка. Ограничитель литерала
"	Двойная кавычка. Ограничитель «закавыченного» идентификатора
<< >>	Ограничители метки
:	Ограничитель переменной базового языка
%	Ограничитель атрибута
.	Точка. Признак компонента, в частности поля записи (*запись.поле*) или элемента пакета (*пакет.элемент*)
@	Признак связи БД
=>	Признак связывания параметров при вызове
..	Две точки. Оператор диапазона в цикле FOR
--	Признак однострочного комментария
/* */	Ограничители многострочного комментария

Структура блока

Каждая программа на PL/SQL представляет собой блок, состоящий из стандартного набора элементов, обозначенных ключевыми словами. Блок определяет область видимости объявленных переменных и порядок обработки и передачи исключений.

Структура блока определяется следующей синтаксической конструкцией:

```
[CREATE OR REPLACE имя [(параметр тип_данных [,параметр тип_данных . . .])
    {IS | AS}]                          -- секция заголовка
[[DECLARE]
    переменная тип_переменной;
    [переменная тип_переменной; . . .]] -- секция объявлений
BEGIN
    выполняемый_код;
    [выполняемый_код; . . ]             -- секция выполнения
[EXCEPTION
    код_исключения;
    [код_исключения; . . ] ]            -- секция исключений
END;
```

Назначение секций:

Секция заголовка

Определяет имя блока. Необходима для именованных блоков и недопустима в анонимных блоках.

Секция объявлений

Содержит объявления переменных, констант, курсоров, типов и локальных программ, используемых в данном блоке. Может отсутствовать.

Секция выполнения

Содержит выполняемый код. Обычно присутствует, но необязательна в спецификациях пакетов и типов.

Секция исключений

Определяет обработку исключений. Может отсутствовать.

Основные свойства этих четырех секций описаны ниже.

Секция заголовка

В секции заголовка указывается имя блока PL/SQL или программного модуля. Наличие или отсутствие этой секции определяется типом блока PL/SQL.

Типы блоков

Есть два основных типа блоков:

Анонимный блок

Не может быть вызван из-за пределов содержащего его блока. Необязательная секция объявлений начинается ключевым словом DECLARE. Блок этого типа имеет такой вид:

```
DECLARE
   today DATE DEFAULT SYSDATE;
BEGIN
   -- Вывести дату
   DBMS_OUTPUT.PUT_LINE ('Сегодня ' || today);
END;
```

Именованный блок

Требует наличия заголовка. Может быть вызван из-за пределов содержащего его блока. В нем для обозначения секции объявлений не применяется ключевое слово DECLARE. Пример блока этого типа:

```
CREATE OR REPLACE PROCEDURE show_the_date
IS
   today DATE DEFAULT SYSDATE;
BEGIN
   -- Вывести дату
   DBMS_OUTPUT.PUT_LINE ('Сегодня ' || today);
END show_the_date;
```

В секции заголовка могут быть объявлены параметры блока. Параметры, если присутствуют, заключены в скобки и разделены запятыми. Имя каждого параметра должно сопровождаться указанием его типа.

Параметры

В процедурах, функциях и курсорах параметры служат для передачи информации между подпрограммой и вызывающей программой. Параметры перечисляются в списке, содержащем один или более параметров. В определение каждого параметра входят его имя, тип данных, назначение и (необязательно) значение по умолчанию.

Синтаксис

```
имя_параметра [назначение] [NOCOPY] тип_данных
   [(:= | DEFAULT) значение]
```

Ключевые слова

назначение

Указывает, будет ли параметр использован для получения или возврата значений. Может принимать следующие значения:

IN

Параметр только для чтения. Значение фактического параметра может использоваться внутри программы, но не может быть изменено. Этот режим установлен по умолчанию.

OUT

Параметр для чтения и записи.

IN OUT

Параметр для чтения и записи.

Если в ходе выполнения процедуры или функции возникает исключение, то значения, присвоенные параметрам типа OUT и IN OUT теряются, если только не задана подсказка NOCOPY.

NOCOPY

Подсказка компилятору использовать передачу параметра по ссылке, а не по значению. По умолчанию PL/SQL передает параметр типа IN OUT по значению, создавая его копию в подпрограмме. Когда параметр содержит много данных (будучи, например, коллекцией или объектом), такое копирование замедляет работу и требует много памяти. NOCOPY указывает PL/SQL, что параметр передается по ссылке при помощи указателя на единственный экземпляр параметра. Недостаток применения NOCOPY в том, что при возникновении исключения в процессе выполнения программы, которая изменяет параметр типа OUT или IN OUT, изменения реальных параметров не откатываются, т. к. они передавались по ссылке, а не по значению.

тип_данных

Может быть любым типом данных PL/SQL или пользовательским типом, но не может быть ограничен в размере. Фактический размер параметра определяется вызывающей программой или ограничением %TYPE, о котором будет рассказано далее в главе.

```
CREATE OR REPLACE PROCEDURE empid_to_name
(in_id           emp.emp_id%TYPE - Компилируется успешно.
,out_last_name   VARCHAR2        -- Компилируется успешно.
,out_first_name  VARCHAR2(10)    -- Не компилируется.
) IS
...
```

Длина out_last_name и out_first_name определяется вызывающей программой:

```
DECLARE
   surname     VARCHAR2(10);
   first_name  VARCHAR2(10);
BEGIN
   empid_to_name(10, surname, first_name);
END;
```

DEFAULT

Указывает значение по умолчанию для входного (IN) параметра. Если входной параметр имеет значение по умолчанию, то можно не указывать его значение явно при вызове программы. Если при вызове значение не указано, автоматически будет подставлено значение по умолчанию.

Нотация передачи параметров

Формальные параметры – это имена, объявляемые в заголовке процедуры или функции.Фактические *параметры* – это значения или выражения, помещаемые в список параметров при вызове функции или процедуры.

PL/SQL поддерживает два типа нотации передачи аргументов для списков параметров:

По позиции (positional notation)

Сопоставляет каждое значение в списке аргументов, заданном при вызове программы, с формальным параметром, стоящим в соответствующей позиции. Именно такой тип связывания параметров используется по умолчанию.

По имени (named notation)

Явно сопоставляет значение аргумента параметру по имени, а не по позиции. В случае применения связывания по имени можно указывать аргументы в любом порядке и пропускать входные параметры, имеющие значения по умолчанию.

Приведем вызов процедуры empid_to_name с использованием двух типов нотации:

```
BEGIN
   -- Неявное связывание по позиции.
   empid_to_name(10, surname, first_name);

   -- Явное связывание по имени.
   empid_to_name(in_id=>10
      ,out_last_name=>surname
      ,out_first_name=>first_name);
END;
```

Связывание по имени и по позиции можно комбинировать. При этом необходимо следить за тем, чтобы позиционные аргументы находились слева от любых аргументов, связываемых по имени. При вызове хранимых функций PL/SQL из SQL связывание по имени не поддерживается.

Секция объявлений

В эту секцию помещаются объявления всех переменных, констант, курсоров и локальных подпрограмм, используемых в данном блоке. Наличие секции объявлений необязательно: если в программе нет перечисленных выше сущностей, то она может отсутствовать. (Сведения об использовании объявленных в этой секции сущностей приведены в разделе «Секция выполнения» ниже в этой главе.)

Типы данных

Объявление переменной содержит указание на тип данных. В PL/SQL в дополнение ко всем разрешенным в SQL типам применяются также несколько собственных типов. Полный список типов данных приведен в приложении A.

Объявления с ограничениями и без ограничений

Указание на тип данных может содержать ограничение, а может не содержать. Ограничение заключается в указании допустимого размера, масштаба или точности, не превышающих соответствующие значения для неограниченного типа. Например:

```
total_sales    NUMBER(15,2); -- Есть ограничение.
emp_id         VARCHAR2(9);  -- Есть ограничение.
company_number NUMBER;       -- Нет ограничения.
book_title     VARCHAR2;     -- Ошибка.
```

Данные, объявленные с ограничениями, занимают меньше места, чем не ограниченные. Не все типы данных могут быть указаны без ограничений. Например, нельзя объявить переменную с типом VARCHAR2 – необходимо указать ее максимальный размер.

Переменные

Прежде чем использовать переменную, ее необходимо объявить в секции объявлений программного блока, или – как глобальную – в спецификации пакета. Встретив объявление переменной, PL/SQL выделяет область памяти для хранения значения и сопоставляет ей имя, обращаясь к которому, можно извлекать значение и изменять его. Для объявления переменной применяется следующий синтаксис:

```
имя_переменной тип_данных [CONSTANT] [NOT NULL]
   [{:= | DEFAULT} начальное_значение]
```

Значения по умолчанию

Сразу после объявления переменной ей по умолчанию присваивается значение NULL. Инициализация всех переменных – характерная черта PL/SQL, этим он отличается от таких языков, как C и Ada. Инициализировать переменную значением, отличным от NULL, позволяет оператор присваивания или указание ключевого слова DEFAULT:

```
counter   BINARY_INTEGER := 0;
priority  VARCHAR2(8)    DEFAULT 'LOW';
```

NOT NULL

В объявление переменной можно добавить ограничение NOT NULL, означающее, что значения NULL для этой переменной недопустимы. Если такое ограничение указано, то необходимо явно присвоить начальное значение этой переменной.

Константы

При объявлении констант (которое осуществляется добавлением ключевого слова CONSTANT) необходимо указать начальное значение, которое в дальнейшем не может быть изменено, например:

```
min_order_qty   NUMBER(1) CONSTANT := 5;
```

Курсорные переменные

По сути курсор представляет собой указатель на результирующее множество в БД. Курсоры применяются для построчного доступа к данным результирующего множества. Существуют два вида курсоров:

Явные курсоры

Команды SELECT, явно объявленные в секции объявлений текущего блока или в спецификации пакета.

Неявные курсоры

Автоматически создаются в ответ на команду SQL в секции исполнения блока PL/SQL.

Об использовании явных и неявных курсоров будет рассказано далее в разделе «Секция выполнения».

Объявление явных курсоров

Для объявления явного курсора в качестве курсорной переменной в секции объявлений блока или пакета (о пакетах поговорим позже) могут применяться следующие формы:

- Курсор без параметров:

```
CURSOR company_cur
   IS
   SELECT company_id FROM company;
```

- Курсор, принимающий аргументы из списка параметров, описанного в разделе «Секция заголовка»:

```
CURSOR company_cur (id_in IN NUMBER) IS
   SELECT name FROM company
   WHERE  company_id = id_in;
```

- Заголовок курсора, содержащий инструкцию RETURN вместо команды SELECT:

```
CURSOR company_cur (id_in IN NUMBER)
   RETURN company%ROWTYPE
```

Последний пример показывает, что объявление курсора может быть отделено от его реализации, например, заголовок может находиться в спецификации пакета, а реализация – в теле пакета.

Ограничения для курсорных переменных

На курсорные переменные налагается ряд ограничений:

- Нельзя объявлять курсорные переменные на уровне пакета. Однако такие переменные можно объявлять в функциях и процедурах пакетов.
- Нельзя присваивать курсорной переменной значения NULL, а также применять операторы сравнения для проверки на равенство, неравенство или неопределенность.
- Ни столбцы базы данных, ни коллекции не могут хранить курсорные переменные.
- Нельзя применять вызовы удаленных процедур для передачи курсорных переменных с одного сервера на другой.

Переменные REF CURSOR

Переменная типа CURSOR указывает на конкретное результирующее множество, в то время как переменная типа REF CURSOR может применяться для указания на различные результирующие множества. Кроме того, SQL-код, используемый в переменной REF CURSOR, может назначаться в выполняемой секции блока PL/SQL, что позволяет реализовывать динамические команды SQL.

Курсор и результирующее множество представляют собой объекты базы данных. Результирующее множество хранится до тех пор, пока на него указывает курсор. Поэтому через переменную REF CURSOR можно организовать передачу результирующих множеств данных от одного программного модуля PL/SQL к другому. Также переменные типа REF CURSOR позволяют скрыть незначительные изменения в запросах.

Тип REF_CURSOR имеет такой синтаксис:

```
TYPE имя_ref-курсора IS REF CURSOR
   [RETURN тип_запись];
```

где *тип_запись* – это существующая в базе данных строка.

Если не применяется инструкция RETURN, то объявляется слабый тип REF CURSOR. Переменные, объявляемые со слабым типом REF CURSOR, могут быть сопоставлены любому запросу во время исполнения. Если же инструкция RETURN присутствует, то определяется сильный тип REF CURSOR. Переменная сильного типа REF CURSOR может быть сопоставлена таким запросам, результирующее множество которых при выполнении совпадает по количеству элементов и их типу со структурой записи, указанной в инструкции RETURN.

Для того чтобы использовать переменные типа REF CURSOR, необходимо сначала создать тип REF_CURSOR, затем объявить экземпляр на основе этого типа. Рассмотрим пример объявления слабого и сильного типов REF CURSOR:

```
DECLARE
   -- Создать тип курсора для таблицы компаний.
   TYPE company_curtype IS REF CURSOR
      RETURN companies%ROWTYPE;

   -- Создать переменную типа REF CURSOR.
   company_cur company_curtype;

   -- Слабый тип, общий подход.
   TYPE any_curtype IS REF CURSOR;
   generic_curvar any_curtype;
```

Закрепленные объявления типов

Для того чтобы *закрепить* (*anchor*) тип данных скалярной переменной (которая содержит одно значение) за другой переменной или столбцом таблицы или представления БД, применяется атрибут %TYPE. Атрибут %ROWTYPE позволяет закрепить объявление за курсором или таблицей.

Рассмотрим различные виды закрепленных объявлений:

```
DECLARE
   tot_sales NUMBER(20,2);
   -- Закрепить тип данных за переменной PL/SQL.
   monthly_sales tot_sales%TYPE;
```

```
-- Закрепить тип данных за столбцом БД.
v_ename employee.last_name%TYPE;

CURSOR mycur IS
   SELECT * FROM employee;

-- Закрепить тип данных за курсором.
myrec mycur%ROWTYPE;
```

В объявлении переменной (но не в определении столбца) инструкция NOT NULL может следовать за закреплением типа, в этом случае необходимо, чтобы объявления типов содержали значения по умолчанию. Значение по умолчанию для закрепленного объявления может отличаться от значения по умолчанию основного объявления.

Подтипы, определяемые программистом

PL/SQL позволяет определять неограниченные скалярные подтипы. Неограниченный подтип предоставляет псевдоним для исходного базового типа данных, например:

```
CREATE OR REPLACE PACKAGE std_types
IS
   -- Объявить глобальные типы на основе стандартных.
   SUBTYPE dollar_amt_t IS NUMBER;
END std_types;

CREATE OR REPLACE PROCEDURE process_money
IS
   -- Использовать объявленный выше глобальный тип.
   credit std_types.dollar_amt_t;
   ...
```

Ограниченный подтип ограничивает новый тип данных до подмножества исходного типа.

Можно определять собственные ограниченные подтипы в программах (начиная с версии Oracle8*i*), что обычно и делается в секции пакета:

```
PACKAGE std_types
IS
   SUBTYPE currency_t IS NUMBER (15,2);

END;
```

Записи

Запись (*record*) – это составная структура данных, которая и по сути, и по структуре похожа на строку таблицы базы данных. Она состоит из нескольких элементов, называемых *полями* (*fields*). Запись в целом не имеет значения – значение имеет каждое отдельное поле, а запись предоставляет возможность хранения этих значений и доступ к ним как к группе. Для того чтобы использовать запись, необходимо определить ее и объявить переменную такого типа.

Типы записей

Существуют три типа записей: на основе таблицы, на основе курсора и определяемые программистом. Записи на основе таблицы и курсора не надо определять явно, т. к. они неявно определяются с той же структурой, что таблица или курсор. Переменные этих типов объявляются при помощи атрибута %ROWTYPE. Поля записей соответствуют столбцам таблицы или столбцам списка выборки для курсора.

А вот записи, определяемые программистом, должны быть явно объявлены командой TYPE в секции объявлений PL/SQL или в спецификации пакета. Затем можно объявить переменные этого типа или использовать его как часть другого типа:

```
DECLARE
   TYPE name_rectype IS RECORD(
      prefix       VARCHAR2(15)
      ,first_name  VARCHAR2(30)
      ,middle_name VARCHAR2(30)
      ,sur_name    VARCHAR2(30)
      ,suffix      VARCHAR2(10) );

   -- Объявить переменную этого типа.
   new_emp_rec name_rectype;
BEGIN
```

Вложенные записи

Вложенные записи – это записи, состоящие из полей, которые представляют собой записи. Вложение записей – это мощное средство нормализации структур данных и сокрытия сложности в программах PL/SQL. Например:

```
DECLARE
   -- Определить запись.
   TYPE phone_rectype IS RECORD (
      area_code  VARCHAR2(3),
      exchange   VARCHAR2(3),
      phn_number VARCHAR2(4),
      extension  VARCHAR2(4));

   -- Определить запись, состоящую из записей.
   TYPE contact_rectype IS RECORD (
      day_phone#  phone_rectype,
      eve_phone#  phone_rectype,
      cell_phone# phone_rectype);

-- Объявить переменную для вложенной записи.
 auth_rep_info_rec contact_rectype;
BEGIN
```

Коллекции

Коллекция (*collection*) – это составная структура данных, которая ведет себя как список или одномерный массив (PL/SQL не поддерживает традиционные массивы).

В PL/SQL существует три типа коллекций: *ассоциативные массивы* (*associative arrays*) (которые назывались *ассоциативными таблицами* (*index-by tables*) в Oracle8 и Oracle8*i* и таблицами PL/SQL в ранних версиях), *вложенные таблицы* (*nested tables*) и *массивы переменной размерности* (*variable arrays*) – VARRAY. Ниже будут кратко описаны объявления всех этих типов. Сравнение типов коллекций и сведения об их применении приведены в разделах «Секция выполнения» и «Использование коллекций» далее в этой главе (сравнение типов коллекций также представлено в табл. 9.2).

Коллекции реализуются как типы (TYPE). Как и для других типов, определяемых пользователем, необходимо сначала определить тип, а затем объявить экземпляр такого типа. Определение типа может быть сохранено в БД или помещено в программу PL/SQL. Каждый экземпляр такого типа является коллекцией.

Таблица 9.2. Сравнение типов коллекций

Характеристика	Ассоциативный массив	Вложенная таблица	VARRAY
Размерность	Одна	Одна	Одна
Может применяться в SQL?	Нет	Да	Да
Может использоваться как тип данных столбца таблицы?	Нет	Да; данные хранятся в отдельной таблице	Да; данные обычно хранятся в той же таблице
Неинициализированное состояние	Пустой (не может содержать NULL); элементы не определены	Атомарно пустая; обращение к элементам недопустимо	Атомарно пустой; обращение к элементам недопустимо
Инициализация	Автоматическая при объявлении	Посредством конструктора, выборки, присваивания	Посредством конструктора, выборки, присваивания
В PL/SQL на элементы можно ссылаться при помощи	BINARY_INTEGER (–2147483647… 2147483647) или символьная строка (VARCHAR2)	Положительное целое от 1 до 2147483647	Положительное целое от 1 до 2147483647
Разреженная?	Да	Изначально – нет; после удалений – да	Нет
Ограниченная?	Нет	Может быть расширена	Да
Можно присваивать значение любому элементу в любой момент времени?	Да	Нет; сначала может потребоваться выполнить процедуру EXTEND	Нет; сначала может потребоваться выполнить EXTEND, причем расширение возможно только до верхней границы
Средства расширения	Присвоить значение элементу с новым индексом	Использовать встроенную процедуру EXTEND или TRIM для уплотнения, без предопределенного максимума	EXTEND или TRIM, но только до объявленного максимального размера
Может сравниваться на предмет равенства	Нет	Нет	Нет
Элементы сохраняют порядковую позицию и индекс при извлечении из БД	Коллекция не может храниться в БД	Нет	Да

Ассоциативный массив

```
CREATE [OR REPLACE] TYPE имя_типа IS TABLE OF тип_элемента [NOT NULL]
    INDEX BY BINARY_INTEGER| PLS_INTEGER | VARCHAR2(n);
```

Одномерная неограниченная разреженная коллекция однородных элементов, которая доступна только в PL/SQL, но не в БД. Ассоциативный массив может индексироваться при помощи типов BINARY_INTEGER, PLS_INTEGER или VARCHAR2.

Ключевые слова

имя_типа

Любой разрешенный идентификатор, который впоследствии будет использован для объявления коллекции.

тип_элемента

Тип элементов коллекции. Все элементы должны быть одного типа; обычно это бывает скалярный тип данных, объектный тип или ссылочный объектный тип. Если элементы представляют собой объекты, то сам объектный тип не может иметь в качестве атрибута коллекцию. К явно запрещенным типам данных для элементов коллекций относятся BOOLEAN, NCHAR, NCLOB, NVARCHAR2, REF CURSOR, TABLE и VARRAY.

максимально_элементов

Максимальное количество элементов данного ассоциативного массива.

NOT NULL

Означает, что в коллекции данного типа не может быть элементов со значением NULL. Однако она может быть атомарно пустой (неинициализированной).

BINARY_INTEGER|PLS_INTEGER|VARCHAR(n)

Ассоциативный массив может индексироваться при помощи типов BINARY_INTEGER, PLS_INTEGER или VARCHAR2.

Вложенная таблица

```
[CREATE [OR REPLACE]] TYPE имя_типа IS TABLE OF тип_элемента [NOT NULL];
```

Одномерная неограниченная коллекция однородных элементов, доступная как в PL/SQL, так и в БД в виде столбца или таблицы. Вложенные таблицы изначально являются плотными (имеют последовательные индексы), но могут стать разреженными в результате удалений.

Ключевые слова

имя_типа

Любой разрешенный идентификатор, который впоследствии будет использован для объявления коллекции.

тип_элемента

Тип элементов коллекции. Все элементы должны принадлежать к одному типу; обычно это бывает скалярный тип данных, объектный тип или ссылочный объектный тип. Если элементы являются объектами, то сам объектный тип не может иметь в качестве атрибута коллекцию. К явно запрещенным типам данных для элементов коллекций относятся BOOLEAN, NCHAR, NCLOB, NVARCHAR2, REF CURSOR, TABLE и VARRAY.

NOT NULL

Означает, что в коллекции данного типа не может быть элементов со значением NULL. Однако она может быть атомарно пустой (неинициализированной).

VARRAY

```
[CREATE [OR REPLACE]] TYPE имя_типа IS VARRAY |
   VARYING ARRAY (максимально_элементов) OF тип_элемента [NOT NULL];
```

Одномерная ограниченная коллекция однородных элементов, которая доступна как в PL/SQL, так и в БД. Массивы VARRAY всегда ограничены и никогда не бывают разреженными. В отличие от вложенных таблиц порядок их элементов сохраняется при сохранении их в БД и извлечении оттуда.

Ключевые слова

имя_типа

Любой разрешенный идентификатор, который впоследствии будет использован для объявления коллекции.

максимально_элементов

Максимальное количество элементов данного VARRAY

тип_элемента

Тип элементов коллекции. Все элементы должны иметь один тип; обычно это скалярный тип данных, объектный тип или ссылочный объектный тип. Если элементы являются объектами, то сам объектный тип не может иметь в качестве атрибута коллекцию. К явно запрещенным типам данных для элементов коллекций относятся BOOLEAN, NCHAR, NCLOB, NVARCHAR2, REF CURSOR, TABLE и VARRAY.

NOT NULL

Означает, что в коллекции данного типа не может быть элементов со значением NULL. Однако она может быть атомарно пустой (неинициализированной).

Создание коллекций

После объявления любой из представленных выше коллекций необходимо создать экземпляр коллекции следующим образом:

```
TYPE index_table IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
indtab index_table;
```

где indtab – это экземпляр index_table.

При помощи атрибута %TYPE коллекции можно связать с ней тип данных другой коллекции, например:

```
TYPE varray1 IS VARRAY(20) OF NUMBER;
v1 varray1;
v2 v1%type;
```

Если определение v1 изменится, то вместе с ним изменится и определение v2.

Исключения

Исключение генерируется, когда в коде PL/SQL возникает ошибка. В этом случае контроль переходит от секции выполнения к секции обработки исключений текущего блока PL/SQL.

Некоторые исключения для сервера Oracle предопределены в пакете STANDARD, можно также объявить собственное исключение с типом данных EXCEPTION следующим образом:

```
DECLARE
   имя_исключения EXCEPTION;
```

Исключение объявляется один раз в пределах блока, но вложенные блоки могут объявить исключение с тем же именем, что и в объемлющем блоке. Если объявлений несколько, то при обработке исключения объявление во внутреннем блоке имеет более высокий приоритет, чем глобальное объявление.

Все объявленные исключения имеют код ошибки 1 и выводят сообщение «User-defined exception», если только не применялась директива EXCEPTION_INIT (о директивах в PL/SQL мы поговорим чуть позже, в соответствующем разделе).

Посредством команды PRAGMA EXCEPTION_INIT пользователь может сопоставить объявленному исключению номер ошибки:

```
DECLARE
   имя_исключения EXCEPTION;
   PRAGMA EXCEPTION_INIT (имя_исключения, номер_ошибки);
```

где *номер_ошибки* – это литеральная величина (переменные здесь не разрешены). Это может быть номер ошибки Oracle (например, –1855) или же номер из пользовательского диапазона от –20000 до –20999.

Предварительное объявление

Программы должны быть объявлены, прежде чем их можно будет использовать. PL/SQL поддерживает взаимную рекурсию, при которой программа A вызывает программу B, после чего программа B вызывает программу A. Для реализации такой рекурсии необходимо использовать *предварительное объявление* (*forward declaration*) программ. В этом случае программа объявляется до того, как определяется, благодаря чему становится возможным ее использование другими программами. Предварительное объявление – это заголовок программы вплоть до ключевого слова IS/AS:

```
PROCEDURE perform_calc(year_in IN NUMBER)
IS
   /* Предварительное объявление функции total_cost. */
   FUNCTION total_cost (...) RETURN NUMBER;

   /* Теперь функция net_profit function может использовать функцию total_cost. */
   FUNCTION net_profit(...) RETURN NUMBER
   IS
   BEGIN
      RETURN total_sales(...) - total_cost(...);
   END;

   /* Функция total_cost вызывает net_profit. */
   FUNCTION total_cost (...) RETURN NUMBER
   IS
```

```
   BEGIN
      IF net_profit(...) < 0
      THEN
         RETURN 0;
         ELSE
         RETURN...;
      END IF;
   END;
BEGIN /* процедура perform_calc */
   ...
END perform_calc;
```

Секция выполнения

Эта секция содержит код, реализующий функциональность блока. Код может представлять собой как любую команду SQL, так и некую управляющую структуру PL/SQL.

Команды

Секция выполнения блока PL/SQL состоит из одной или более логических команд. Команда завершается разделителем – точкой с запятой. Физический маркер конца строки в программе PL/SQL игнорируется компилятором (воспринимается только как завершение однострочного комментария, который начинается символами --).

Условные управляющие операторы

PL/SQL включает в себя три разновидности команды IF-THEN-ELSE, а также команду и выражение CASE. Синтаксис CASE появился только в версии Oracle9*i*.

IF–THEN–ELSE

Конструкция IF-THEN:

```
IF условие THEN
   выполняемая команда(ы)
END IF;
```

Конструкция IF-THEN-ELSE:

```
IF условие THEN
   последовательность выполняемых команд
ELSE
   последовательность выполняемых команд
END IF;
```

Конструкция IF-THEN-ELSIF:

```
IF условие-1 THEN
   команды-1
ELSIF условие-N THEN
   команды-N
[ELSE
   else_команды]
END IF;
```

Позволяет реализовать в программах условную логику. Существует три варианта структуры IF-THEN-ELSE.

CASE (Команда)

```
CASE выражение-переключатель
    WHEN выражение_для_значения THEN действие;
    [WHEN выражение_для_значения THEN действие; . . .]
    [ELSE
        действие;]
END CASE;
```

Позволяет выбрать для выполнения одну последовательность команд из множества предложенных последовательностей. Команда CASE похожа на команду IF-THEN-ELSIF. Но в команде CASE выражение-переключатель следует непосредственно за ключевым словом CASE. Выражение вычисляется и сравнивается со значением из каждой инструкции WHEN. Как только операция сравнения возвращает значение TRUE, управление передается команде, следующей за END CASE.

Если выражение-переключатель вычислено как NULL, то ему будет соответствовать только инструкция ELSE. Инструкция WHEN NULL не будет соответствовать никакому значению, т. к. сервер Oracle выполняет проверку равенства выражений.

И команда, и выражение CASE (см. следующий раздел) должны включать в себя инструкцию ELSE, которая будет выполнена в случае, если ни одна из инструкций WHEN не будет оценена как TRUE, т. к. в случае отсутствия совпадения процессор PL/SQL выдаст ошибку.

CASE (Выражение)

```
CASE
    WHEN выражение = выражение_для_значения THEN
        действие;
    [WHEN выражение = выражение_для_значения THEN
        действие; . . .]
       [ELSE
           действие;]
END CASE;
```

В отличие от команды CASE выражение CASE не содержит выражения-переключателя; вместо этого каждая инструкция WHEN содержит полное булево выражение. Выполняется первое подходящее выражение WHEN, затем управление переходит к команде, следующей за END CASE.

Команды управления последовательностью выполнения

Существуют два типа команд управления последовательностью выполнения: GOTO и NULL.

GOTO

```
GOTO имя метки;
```

Выполняет безусловный переход к указанной метке. Обычно применять команду GOTO не рекомендуется. Если она используется, то за указанной меткой должна следовать хотя бы одна команда (при необходимости этой единственной выполняемой командой может быть команда NULL).

Ограничения:

На применение команды GOTO накладываются следующие ограничения:

- Невозможен переход из команды IF, LOOP или вложенного блока
- Невозможен переход в команду IF, LOOP или вложенный блок
- Невозможен переход из одной секции команды IF в другую (из секции IF/THEN в секцию ELSE)
- Невозможен переход из и в подпрограмму
- Невозможен переход из секции обработки исключений в выполняемую секцию блока PL/SQL
- Невозможен переход из выполняемой секции в секцию обработки исключений блока PL/SQL, хотя такой переход осуществляется командой RAISE

NULL

```
NULL;
```

Команда, которая ничего не делает. Команда NULL полезна, когда нужна выполняемая, но ничего не делающая команда после метки. Также она может применяться для улучшения читаемости структуры IF-THEN-ELSE.

Команды цикла

Структура LOOP позволяет многократно выполнять последовательность команд. Существуют три вида циклов: простой цикл и циклы WHILE и FOR; при этом FOR имеет две разновидности: цикл со счетчиком и цикл по курсору.

Все циклы могут иметь метку или включать в себя команду EXIT.

Циклы могут быть дополнительно (необязательно) помечены для того, чтобы улучшить читаемость и контроль за их выполнением. Метка должна стоять непосредственно перед командой начала цикла.

Команда EXIT применяется для прерывания цикла и передачи управления команде, следующей за END LOOP. Синтаксис команды EXIT:

```
EXIT [WHEN логическое_условие];
```

Если в команду EXIT не включена инструкция WHEN, то цикл будет немедленно завершен командой. В противном случае цикл закончится только в случае, если значение выражения *логическое_условие* будет вычислено как TRUE. Команда EXIT является необязательной и может присутствовать в любой точке цикла.

Простой цикл

```
LOOP
   выполняемая команда(ы)
END LOOP;
```

Цикл простейшей конструкции. Такой цикл состоит только из команд цикла LOOP и END, между которыми должна быть заключена хотя бы одна выполняемая команда. Простой цикл обычно применяется, если необходимо гарантировать хотя бы од-

нократное выполнение тела цикла. Такой цикл завершится только когда в его теле будет выполнена команда EXIT или EXIT WHEN (или не будет обработано порожденное в нем исключение). Если этого не произойдет, то цикл будет выполняться бесконечно.

Цикл FOR со счетчиком

```
FOR переменная_цикла IN [REVERSE] наименьшее_число..
   наибольшее_число
LOOP
   выполняемая команда(ы)
END LOOP;
```

Традиционный цикл со счетчиком, в котором указывается количество повторений. Процессор PL/SQL автоматически объявляет переменную цикла с типом PLS_INTEGER; помните об этом и никогда не объявляйте сами переменную с тем же именем. Значения *наименьшее число* и *наибольшее число* могут быть переменными, но вычисляются они только один раз – при первом входе в цикл. Ключевое слово REVERSE означает, что выполнение начнется со значения *наибольшее число* и будет происходить по убыванию вплоть до значения *наименьшее число*.

Цикл FOR по курсору

```
FOR переменная_запись IN [имя_курсора | (SELECT команда)]
LOOP
   выполняемая команда(ы)
END LOOP;
```

Цикл, определяемый явным курсором или командой SELECT. Этот тип цикла автоматически открывает курсор, выбирает все определяемые им строки и закрывает курсор. Можно поместить команду SELECT непосредственно в цикл FOR.

Процессор PL/SQL автоматически объявляет переменную цикла как запись типа *имя_курсора*%ROWTYPE; помните об этом и никогда не объявляйте сами переменную с тем же именем.

Цикл WHILE

```
WHILE условие
LOOP
   выполняемая команда(ы)
END LOOP;
```

Цикл с условием, который выполняется до тех пор, пока логическое условие истинно. Цикл WHILE применяется в тех случаях, когда заранее неизвестно, сколько раз следует повторять операции. Кроме того, цикл WHILE можно применять вместо простого цикла, если не нужно, чтобы тело цикла выполнилось даже единожды.

Явные курсоры

Курсор – это указатель на результирующее множество в базе данных. Явный курсор – это курсор, который вы явно называете в программе и явно манипулируете им. Явный курсор является статическим; команда SQL определяется в момент компиля-

ции. Статические курсоры применяются только для команд DML (SELECT, INSERT, UPDATE, DELETE, MERGE или SELECT FOR UPDATE).

Синтаксис, применяемый при работе с явными курсорами, рассматривается далее.

OPEN

```
OPEN имя_курсора [(аргумент [,аргумент ...])];
```

Открывает явный курсор. Прежде чем извлекать строки из явного курсора, его необходимо открыть. Обработка открытого курсора включает в себя такие этапы выполнения команды SQL, как разбор, связывание, открытие и исполнение.

Курсор, используемый в цикле FOR, открывается неявно. При попытке открытия уже открытого курсора PL/SQL порождает исключение «ORA-06511: PL/SQL: cursor already open».

Ключевые слова

имя_курсора

Имя курсора, с которым он объявлен в секции объявлений.

аргумент

Необходим, если определение курсора включает в себя список параметров.

FETCH

```
FETCH имя_курсора INTO запись_или_список_переменных;
```

Выбирает данные из явного курсора. Помещает содержимое текущей строки в локальные переменные. Для извлечения всех строк результирующего множества необходимо выполнить эту команду для каждой строки.

Ключевое слово

имя_курсора

Имя курсора, с которым он объявлен и открыт.

CLOSE

```
CLOSE имя_курсора;
```

Закрывает явный курсор. После того как все строки извлечены, курсор должен быть закрыт. Закрытие курсора позволяет механизму оптимизации памяти PL/SQL освободить выделенную память в нужный момент.

Если курсор объявляется в локальном анонимном блоке функции или процедуры, то такой курсор будет автоматически закрыт по завершении блока. Курсоры, объявленные в пакете, должны быть закрыты явно, иначе они останутся открытыми до завершения сеанса. Попытка закрытия курсора, который не был открыт, приводит к порождению исключения INVALID CURSOR.

Ключевое слово

имя_курсора

Имя курсора, с котором он объявлен и открыт.

Атрибуты явного курсора

```
имя_курсора%атрибут
```

Курсору сопоставлены четыре атрибута: ISOPEN, FOUND, NOTFOUND и ROWCOUNT.

Ключевые слова

имя_курсора

Имя явного курсора.

атрибут

Одно из четырех ключевых слов:

%ISOPEN

Возвращает TRUE, если курсор открыт; FALSE – если курсор не открыт.

%FOUND

Возвращает NULL перед первым извлечением, TRUE – при успешном извлечении записи, FALSE – если не было возвращено ни одной строки. Генерируется исключение INVALID_CURSOR в случае, если курсор не был открыт или уже закрыт.

%NOTFOUND

Возвращает NULL перед первым извлечением, FALSE – при успешном извлечении записи, TRUE – если не было возвращено ни одной строки. Генерируется исключение INVALID_CURSOR в случае, если курсор не был открыт или уже закрыт.

%ROWCOUNT

Возвращает количество записей, извлеченных из курсора. Генерируется исключение INVALID_CURSOR в случае, если курсор не был открыт или уже закрыт.

Неявные курсоры

PL/SQL объявляет неявный курсор и работает с ним при каждом выполнении команды DML (INSERT, UPDATE, DELETE или MERGE) или команду SELECT INTO, которая возвращает одну строку из базы данных непосредственно в структуру данных PL/SQL. Каждый раз, когда вы напрямую запускаете команду SQL в выполняемой секции или секции обработки исключений блока команд PL/SQL, вы работаете с неявными курсорами. В отличие от явных курсоров, неявные не надо объявлять, открывать, извлекать или закрывать.

Как и явный курсор, неявный соответствует статической команде SQL (о динамических командах мы поговорим в следующем разделе).

Синтаксис

```
команда_sql
    [RETURNING значение[, значение . . .]
        INTO переменная[, переменная . . .] ]
    %атрибут
```

Ключевые слова

команда_sql

Команда INSERT, UPDATE, DELETE, MERGE или SELECT INTO.

RETURNING

Инструкция RETURNING применяется в командах INSERT, UPDATE и DELETE для получения данных, измененных соответствующей командой DML. Применение этой инструкции позволяет избежать дополнительной команды SELECT для запроса результатов выполнения команды DML, например:

```
BEGIN
   UPDATE activity SET last_accessed := SYSDATE
   WHERE UID = user_id
   RETURNING last_accessed, cost_center
   INTO timestamp, chargeback_acct;
```

атрибут

Ключевое слово, следующее непосредственно за последней строкой команды SQL без пробелов. Возможные значения:

SQL%ISOPEN

Всегда возвращает FALSE для неявных курсоров, т. к. курсор открывается неявно и закрывается сразу же после выполнения команды.

SQL%FOUND

Возвращает TRUE, если одна или более строк были обновлены, вставлены, удалены или объединены, или если была выбрана всего одна строка; FALSE – если ни одна строка не была обработана.

SQL%NOTFOUND

Возвращает NULL перед выполнением команды; TRUE – если ни одной строки не было выбрано, обновлено, вставлено, удалено или объединено; FALSE – если одна или несколько строк были обработаны.

SQL%ROWCOUNT

Возвращает количество строк, обработанных курсором.

SQL%BULK_ROWCOUNT

Псевдоасссоциативный массив, содержащий количество записей, измененных командой FORALL (см. раздел «FORALL» далее в этой главе) для каждого элемента коллекции.

SQL%BULK_EXCEPTIONS

Псевдоасссоциативный массив, содержащий порядковые номера обработанных записей, вызвавших исключение и соответствующие коды ошибок. Для продолжения обработки после возникновения исключения применяется параметр SAVE EXCEPTIONS.

Динамические курсоры и SQL

Для динамического курсора команда SQL определяется во время исполнения. Динамические курсоры применяются для любых видов разрешенных команд SQL, таких как CREATE, TRUNCATE, ALTER, GRANT и REVOKE.

EXECUTE IMMEDIATE

```
EXECUTE IMMEDIATE команда_sql
    [INTO {переменная [,переменная ...] | запись| объектная_переменная}]
    [USING [IN | OUT | IN OUT] связанный_аргумент
        [,[IN | OUT | IN OUT] связанный_аргумент...] ]
    [{RETURNING | RETURN} INTO связанный_аргумент [,связанный_аргумент]...];
```

Осуществляет синтаксический анализ команды SQL и ее выполнение за один шаг. Может применяться для любой команды SQL, кроме многострочных запросов. В Oracle9*i* посредством команды EXECUTE IMMEDIATE можно выполнять массовое извлечение, а также массовую вставку или обновление, даже если при массовой обработке были обнаружены ошибки.

Для применения динамического SQL в запросах, возвращающих несколько строк, применяется переменная CURSOR, которая описана в следующем разделе.

Команда EXECUTE IMMEDIATE должна завершаться точкой с запятой, тогда как *команда_sql* не должна содержать завершающей точки с запятой.

Ключевые слова

Ключевые слова совпадают со стандартными ключевыми словами SQL и ключевыми словами для явных курсоров.

Использование курсорных переменных

Начиная с версии Oracle8*i* динамический SQL поддерживает курсорные переменные.

Курсорная переменная может быть объявлена в секции объявлений программы PL/SQL как экземпляр типа данных REF CURSOR. Для того чтобы использовать переменную CURSOR, откройте ее с помощью описанной ниже команды OPEN. . .FOR. После того как курсорная переменная открыта, можно применять FETCH и CLOSE с тем же синтаксисом, что и для явных курсоров.

OPEN. . .FOR

```
OPEN имя_курсора FOR select_команда;
```

Открывает ранее объявленную курсорную переменную.

Использование курсорных выражений

Появившееся в Oracle9*i* *курсорное выражение* (*cursor expression*) – это курсор, используемый как выражение для столбца в списке выбора для явного курсора.

Курсорные выражения могут уменьшить объем ненужных данных, возвращаемых в вызывающую программу. В связи с этим их можно применять вместо приемов, требующих соединения таблиц. Курсорное выражение открывается автоматически при извлечении родительской строки. Курсорные выражения могут быть вложенными. Такие вложенные курсоры закрываются, когда происходит одно из событий:

- Вложенный курсор явно закрывается программой.
- Закрывается родительский курсор.
- Заново выполняется родительский курсор.
- Во время извлечений родительской строки порождается исключение.

CURSOR

```
CURSOR (подзапрос)
```

Возвращает вложенный курсор из запроса.

Использование записей

На отдельные поля записей можно ссылаться посредством точечной нотации:

```
имя_записи.имя_поля
```

Отдельные поля записи могут быть прочитаны и записаны. Они могут стоять как слева, так и справа от оператора присваивания. Запись целиком может быть присвоена другой записи того же типа, но сравнение двух записей при помощи логического оператора невозможно. Например, такое присваивание разрешено:

```
shipto_address_rec := customer_address_rec
```

Но приведенное ниже сравнение уже не разрешено (вместо этого следует сравнивать отдельные поля записей):

```
IF shipto_address_rec = customer_address_rec
THEN
...
END IF;
```

Присваивание значений записям и полям записей может выполняться четырьмя различными способами:

- Можно применять оператор присваивания для присваивания значения полю:

```
new_emp_rec.hire_date := SYSDATE;
```

- Можно выполнить SELECT INTO для целой записи или отдельных полей:

```
SELECT emp_id,dept,title,hire_date,college_recruit
  INTO new_emp_rec
  FROM emp
 WHERE surname = 'LI'
```

- Можно выполнить FETCH INTO для целой записи или отдельных полей:

```
FETCH emp_cur INTO new_emp_rec;
FETCH emp_cur INTO new_emp_rec.emp_id,
   new_emp_rec.name;
```

- Можно присвоить все поля одной переменной записи другой переменной записи того же типа:

```
IF rehire THEN
 new_emp_rec := former_emp_rec;
ENDIF;
```

Агрегатное присваивание поддерживается только для записей, объявленных одинаковыми командами TYPE.

Использование коллекций

В этом разделе описана работа с коллекциями. Типы коллекций (ассоциативные массивы, вложенные таблицы и массивы переменной длины) были описаны ранее в разделе «Секция объявлений».

Функции и методы коллекций

Действия с коллекциями можно выполнять посредством следующих функций:

CAST

Преобразует тип одной коллекции к типу другой коллекции.

```
SELECT column_value
FROM TABLE(SELECT CAST(colors AS color_tab_t)
           FROM color_models_a
          WHERE model_type ='RGB');
```

MULTISET

Отображает таблицу БД на коллекцию. Применяя MULTISET и CAST, можно извлекать строки из таблицы БД в виде столбца с типом коллекции. Можно применять CAST и MULTISET для извлечения вложенных столбцов и использования их в триггерной логике (поговорим об этом чуть позже).

```
SELECT b.genus ,b.species,
      CAST(MULTISET(SELECT bh.country
                    FROM bird_habitats bh
                   WHERE bh.genus = b.genus
                     AND bh.species = b.species)
          AS country_tab_t)
FROM birds b;
```

TABLE

Отображает коллекцию на таблицу БД; обратная функция к MULTISET.

```
SELECT *
  FROM color_models c
 WHERE 'RED' IN (SELECT * FROM TABLE(c.colors));
```

Посредством TABLE() можно преобразовывать нерезидентную коллекцию:

```
DECLARE
   birthdays Birthdate_t :=
      Birthdate_t('24-SEP-1984', '19-JUN-1993');
BEGIN
   FOR the_rec IN
      (SELECT COLUMN_VALUE
         FROM TABLE(CAST(birthdays AS Birthdate_t)))
```

Коллекции также поддерживают ряд методов, описанных в разделе «Типы данных коллекций» в приложении А:

COUNT
DELETE
EXISTS
EXTEND

FIRST
LAST
LIMIT
PRIOR
NEXT
TRIM

Для этих методов применяется такой синтаксис:

```
имя_коллекции.имя_метода[(параметры)]
```

Инициализация коллекций

Объявление ассоциативного массива означает также и его инициализацию. Инициализация вложенной таблицы и массива VARRAY также может быть выполнена следующим образом:

- Явно, при помощи конструктора
- Неявно, посредством извлечения из БД
- Неявно, посредством прямого присваивания другой переменной коллекции

Конструктор – это встроенная функция, имя которой совпадает с именем коллекции. Она создает коллекцию из переданных ей элементов. Например, можно создать вложенную таблицу цветов и явно инициализировать ее тремя элементами при помощи конструктора:

```
DECLARE
   TYPE colors_tab_t IS TABLE OF VARCHAR2(30);

   colors_tab_t('RED','GREEN','BLUE');
BEGIN
```

Применяя второй способ инициализации, можно создать вложенную таблицу цветов и неявно инициализировать ее при помощи извлечения данных из БД:

```
-- Создать хранимый в БД тип вложенной таблицы.
CREATE TYPE colors_tab_t IS TABLE OF VARCHAR2(32);

-- Инициализировать коллекцию цветов данными из таблицы.
DECLARE
   basic_colors colors_tab_t;
BEGIN
   SELECT colors INTO basic_colors
     FROM color_models
    WHERE model_type = 'RGB';
...
END;
```

Опробуем и третий метод инициализации – инициализируем таблицу неявно, используя присваивание из существующей коллекции:

```
DECLARE
   basic_colors Color_tab_t :=
      Color_tab_t ('RED','GREEN','BLUE');

   my_colors Color_tab_t;
BEGIN
   my_colors := basic_colors;
   my_colors(2) := 'MUSTARD';
```

Добавление и удаление элементов

Можно добавить элементы в ассоциативный массив, просто сославшись на новые индексы. Для добавления элементов во вложенные таблицы или массивы VARRAY необходимо сначала увеличить коллекцию с помощью функции EXTEND, а затем присвоить новому элементу значение посредством одного из ранее представленных методов.

Для удаления элемента вложенной таблицы, независимо от его позиции, применяется функция DELETE. Удалять элементы позволяет и функция TRIM, но только для удаления из конца коллекции. Не стоит использовать для одной коллекции и DELETE, и TRIM – результаты могут быть совсем не теми, которых вы ожидали.

Привилегии

Как и для других типов в БД, для того чтобы создавать тип коллекции на основе типа из другой схемы, необходимо обладать привилегией EXECUTE на тип из другой схемы.

Вложенные коллекции

Вложенные коллекции – это коллекции, содержащиеся в элементах, которые сами являются членами коллекции. Вложение коллекций представляет собой мощное средство реализации объектно-ориентированных программных конструкций в программах PL/SQL. Обсуждение данного вопроса выходит за рамки этой книги (эта возможность подробно описана в книге «Oracle PL/SQL Programming»).

Приведем простой пример определения вложенной коллекции:

```
CREATE TYPE books IS TABLE OF VARCHAR2(64);
CREATE TYPE our_books IS TABLE OF books;
```

Массовое связывание

Коллекции можно применять для улучшения производительности многократно выполняемых операций SQL за счет *массового связывания* (*bulk binds*). Массовое связывание уменьшает количество переключений контекста от процессора PL/SQL к процессору БД и обратно. Кроме того, связывание может применяться для реализации динамического SQL в Oracle9*i*.

Массовое связывание реализуется двумя конструкциями PL/SQL: FORALL и BULK COLLECT INTO.

FOR ALL

```
FORALL индекс IN нижняя_граница..верхняя_граница [SAVE EXCEPTIONS]
   {EXECUTE IMMEDIATE команда_sql | команда_sql };
```

Когда PL/SQL обрабатывает эту команду, на обработку серверу БД отправляется вся коллекция, а не каждый ее элемент по отдельности. Можно применять инструкцию FOR ALL, как и BULK COLLECT INTO, обращаясь к синтаксису EXECUTE IMMEDIATE для динамического SQL.

Ключевые слова

индекс

Целая переменная. Может применяться только для *команды_sql* и только как индекс коллекции.

нижняя_граница..верхняя_граница

Ограничения для элементов коллекции, которые будут обрабатываться процессором БД.

SAVE EXCEPTIONS

Позволяет продолжить массовую обработку даже в случае ошибок в некоторых отдельных операциях. После завершения обработки цикла подробную информацию о возникших проблемах можно будет получить в %BULK_EXCEPTIONS.

BULK COLLECT INTO

```
команда_sql BULK COLLECT INTO список_имен_коллекций;
```

Когда PL/SQL обрабатывает эту команду, сервер БД возвращает для обработки всю коллекцию целиком, а не каждый ее элемент в отдельности. Можно включать инструкцию BULK COLLECT INTO в команды SELECT INTO, FETCH INTO и RETURNING INTO. Ее можно применять с динамическим SQL, выполняемым при помощи команды EXECUTE IMMEDIATE.

Для операций массового связывания можно использовать атрибут курсора SQL%BULK_ROWCOUNT. Этот атрибут подобен ассоциативному массиву, содержащему количество строк, измененных при выполнении команд массового связывания. Элемент SQL%BULK_ROWCOUNT с номером n содержит количество строк, измененных n-м выполнением команды SQL.

Передать SQL%BULK_ROWCOUNT как параметр в другую программу нельзя, как нельзя и использовать агрегатное присваивание другой коллекции. %ROWCOUNT содержит сумму всех элементов %BULK_ROWCOUNT. Атрибуты %FOUND и %NOTFOUND отражают результаты самого последнего выполнения команды SQL.

Ключевые слова

команда_sql

Команда SQL, такая как SELECT INTO, FETCH INTO, RETURNING INTO или EXECUTE IMMEDIATE.

список_имен_коллекций

Разделенный запятыми список коллекций, по одной для каждого столбца списка SELECT. Коллекции записей не могут быть предметом назначения инструкции BULK COLLECT INTO. Однако сервер Oracle поддерживает извлечение набора объектов одного типа и массовую сборку их в коллекцию объектов.

Управление транзакциями

Модель транзакций реляционной СУБД Oracle основана на «единице работы» (*unit of work*). Язык PL/SQL поддерживает большую часть операций над транзакциями (но не все).[1] Транзакция неявно начинается с началом сеанса или при выполнении первой команды SQL после последней команды COMMIT или ROLLBACK (фактически, при первом изменении данных). Транзакция заканчивается при выполнении команды COMMIT или ROLLBACK.

[1] Например, нельзя использовать ROLLBACK FORCE.

Транзакции не зависят от блоков PL/SQL. Транзакции могут охватывать несколько блоков, или в одном блоке может быть несколько транзакций (о так называемых автономных транзакциях рассказано в следующем разделе). PL/SQL поддерживает следующие команды для транзакций: COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION и LOCK TABLE.

COMMIT

```
COMMIT [WORK] [COMMENT текст];
```

Делает изменения в базе данных постоянными и видимыми для других сеансов. Имейте в виду, что PL/SQL не поддерживает команду COMMIT FORCE для распределенных транзакций.

Ключевые слова

WORK

Необязательное ключевое слово. Применяется только для улучшения читаемости.

COMMENT текст

Необязательный комментарий. Может иметь длину до 50 символов.

ROLLBACK

```
ROLLBACK [WORK] [TO [SAVEPOINT] имя_точки_сохранения];
```

Отменяет незафиксированные изменения, сделанные в текущей транзакции, вплоть до начала транзакции или до указанной точки сохранения.

Ключевое слово

имя_точки_сохранения

Имя точки сохранения внутри транзакции, которая создается командой SAVEPOINT.

SAVEPOINT

```
SAVEPOINT имя_точки_сохранения;
```

Устанавливает точку сохранения (именованную точку обработки) для текущей транзакции. Установка точки сохранения делает возможным выполнение частичного отката.

Ключевое слово

имя_точки_сохранения

Необъявленный идентификатор. Внутри транзакции может быть установлено несколько точек сохранения. Если повторно использовать имя точки сохранения, то точка передвинется на новую позицию и откат к исходной позиции данной точки сохранения будет невозможен.

SET TRANSACTION

```
SET TRANSACTION тип_транзакции  NAME имя
```

Управляет типом транзакции.

Ключевые слова

тип_транзакции

Может принимать следующие значения:

READ ONLY

Означает начало транзакции только для чтения. Указывает СУБД, что следует обеспечить целостность по чтению БД в пределах транзакции (по умолчанию – для команды). Транзакцию заканчивает команда COMMIT или ROLLBACK. Внутри такой транзакции разрешены только команды LOCK TABLE, SELECT, SELECT INTO, OPEN, FETCH, CLOSE, COMMIT и ROLLBACK. Попытка выполнения в транзакции только для чтения других команд, таких как INSERT или UPDATE, приводит к появлению ошибки ORA-1456.

READ WRITE

Обозначает начало транзакции READ WRITE; это тип по умолчанию.

ISOLATION LEVEL SERIALIZABLE

Аналогично транзакции READ ONLY обеспечивается целостность по чтению в пределах транзакции вместо поведения по умолчанию – целостности по чтению на уровне команды. Сериализуемые транзакции не позволяют изменять данные. Дополнительная информация об уровнях изоляции была приведена в главе 3.

ISOLATION LEVEL READ COMMITTED

Если транзакции необходимы строки, заблокированные другими транзакциями, она будет ждать снятия блокировки.

USE ROLLBACK SEGMENT имя_сегмента_отката

Указывает СУБД, что следует использовать указанный сегмент отката. Полезна, когда лишь один сегмент отката имеет большой размер, и известно, что программе необходим большой сегмент отката (например, при операциях закрытия в конце месяца).

имя Имя транзакции, доступное на протяжении ее выполнения.

LOCK TABLE

```
LOCK TABLE список_таблиц IN режим_блокировки MODE [NOWAIT];
```

Обходит неявные блокировки БД на уровне строк, явно блокируя целиком одну или несколько таблиц в указанном режиме.

Ключевые слова

список_таблиц

Список таблиц, разделенных запятыми.

режим_блокировки

Возможны значения ROW SHARE, ROW EXCLUSIVE, SHARE UPDATE, SHARE, SHARE ROW EXCLUSIVE или EXCLUSIVE.

NOWAIT

Указывает, что СУБД не должна ждать освобождения блокировки. Если встречается блокировка (и указано ключевое слово NOWAIT), то СУБД генерирует исключение:

```
ORA-00054: resource busy and acquire with NOWAIT specified
```

По умолчанию при встрече блокировки СУБД бесконечно ждет ее освобождения.

Автономные транзакции

Появившиеся в Oracle8*i* *автономные транзакции* (*autonomous transactions*) выполняются внутри блока кода как отдельные транзакции из внешней (главной) транзакции. Изменения, осуществленные в автономной транзакции, могут быть зафиксированы или подвержены откату без фиксации/отката основной транзакции. Изменения, зафиксированные в автономной транзакции, видны главной транзакции, даже если они произошли после начала основной транзакции. Изменения, зафиксированные в автономной транзакции, видны и другим транзакциям. СУБД приостанавливает основную транзакцию на время выполнения автономной.

Автономные транзакции также позволяют фиксировать и откатывать транзакции в триггерах, не вмешиваясь в работу самого триггера.

Изменения, выполненные в главной транзакции, не видны в автономной, и если главная транзакция удерживает блокировки, освобождения которых ожидает автономная транзакция, то возникает тупиковая ситуация – взаимная блокировка. Использование параметра NOWAIT для команды UPDATE в автономной транзакции помогает минимизировать вероятность таких ситуаций. Функции и процедуры, триггеры БД, анонимные блоки PL/SQL верхнего уровня и объектные методы могут быть объявлены автономными при помощи директивы компилятора PRAGMA AUTONOMOUS_TRANSACTION.

Применяя автономные транзакции, необходимо следовать таким требованиям:

- Использовать директиву PRAGMA AUTONOMOUS_TRANSACTION
- Указывать COMMIT или ROLLBACK в каждой точке выхода автономной программы. В противном случае будет сгенерирована ошибка:

```
ORA-06519: active autonomous transaction detected and rolled back
```

Рассмотрим пример, учитывающий оба этих требования:

```
PROCEDURE main IS
BEGIN
   UPDATE ...-- Здесь начинается главная транзакция.
   DELETE ...
   at_proc;  -- Вызов автономной транзакции.
   SELECT ...
   INSERT ...
   COMMIT;   -- Здесь заканчивается главная транзакция.
END;

PROCEDURE at_proc IS
   PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN        -- Здесь приостанавливается главная транзакция.
   SELECT ...
   INSERT ...-- Здесь начинается автономная транзакция.
   UPDATE ...
   DELETE ...
```

```
   COMMIT;    -- Здесь заканчивается автономная транзакция.
END;          -- Здесь возобновляется основная транзакция.
```

Секция обработки исключений

PL/SQL предоставляет пользователям гибкие и мощные средства генерации и обработки ошибок (исключений). Каждый блок PL/SQL может иметь собственную секцию обработки исключений, в которой исключения будут перехватываться и обрабатываться. Когда в блоке PL/SQL порождается исключение, его секция выполнения сразу же завершает работу. Управление переходит к секции обработки исключений. Каждое исключение в PL/SQL имеет номер ошибки и сообщение об ошибке; у некоторых исключений есть и имена.

Когда возникает (или порождается) исключение, поток выполнения покидает выполняемую секцию и управление передается непосредственно в секцию обработки исключений текущего блока PL/SQL. Если исключение не обработано в секции исключений, то управление передается за пределы блока.

Генерация исключений

Исключение может генерироваться тремя разными способами:

- Процессором PL/SQL
- Явной командой RAISE в коде
- Вызовом встроенной функции RAISE_APPLICATION_ERROR

RAISE

```
RAISE имя_исключения;
```

Явно генерирует исключение. Если использовать команду RAISE внутри обработчика исключений, то для повторного порождения текущего исключения имя исключения можно не указывать. За пределами секции исключений такой синтаксис некорректен.

Ключевое слово

имя_исключения

Имя объявленного пользователем исключения или имя исключения, объявленного в пакете STANDARD.

RAISE_APPLICATION_ERROR

```
RAISE_APPLICATION_ERROR (
   номер BINARY_INTEGER,
   сообщение VARCHAR2,
   стек_ошибок BOOLEAN DEFAULT FALSE);
```

Исключение можно сгенерировать вызовом этой встроенной функции.

Ключевые слова

номер

Номер ошибки, целое число в диапазоне от –20999 до –20000.

сообщение

Соответствующее сообщение об ошибке.

стек_ошибок

Управляет содержимым стека ошибок.

Область видимости исключений

Обработчик исключений обрабатывает или пытается обработать только исключения, порожденные в выполняемой секции блока PL/SQL. Исключения, генерируемые в объявлении или в секции исключений, автоматически передаются внешнему блоку. Можно ограничить область видимости исключения, поместив код PL/SQL в отдельный блок с собственной секцией обработки исключений.

Распространение исключений

Исключения, сгенерированные в блоке PL/SQL, распространяются во внешний блок, если они не были обработаны в секции обработки исключений или были в ней повторно сгенерированы.

Когда возникает исключение, PL/SQL ищет в текущем блоке обработчик исключений, обрабатывающий соответствующую ошибку (или инструкцию WHEN OTHERS, которая описана далее). Если совпадение не обнаружено, то PL/SQL передает исключение во внешний блок или вызывающую программу. Распространение продолжается до тех пор, пока исключение не обработано или не передано вне самого внешнего блока обратно в вызывающую программу. В последнем случае исключение остается необработанным и приводит к остановке вызывающей программы.

Если исключение порождается в блоке PL/SQL, оно не приводит к откату текущей транзакции, даже если сам блок выдавал команду INSERT, UPDATE или DELETE. Для того чтобы очистить транзакцию в случае возникновения исключения, следует явно выдать команду ROLLBACK.

Однако если исключение осталось необработанным (передано за пределы самого внешнего блока), то многие среды выполняют автоматический безусловный откат всех незафиксированных изменений для сеанса.

Обработанное исключение дальше не распространяется. Для того чтобы перехватить исключение, вывести содержательное сообщение об ошибке и передать исключение во внешний блок как ошибку, его необходимо сгенерировать заново. Заново сгенерировать текущее или новое исключение позволяет описанная ранее команда RAISE.

Инструкция WHEN OTHERS

```
EXCEPTION
   WHEN OTHERS
      THEN...
```

Инструкция в обработчике исключений. Может применяться как ловушка для всех исключений, не обрабатываемых специальными отдельными инструкциями WHEN в секции обработки исключений. Если эта инструкция присутствует, она должна быть последней в секции обработки исключений.

SQLCODE

Встроенная функция, которая возвращает код ошибки SQL для текущего исключения. Пользователь может включить эту функцию внутрь инструкции WHEN OTHERS для обработки определенных ошибок по номеру. Директива компилятора EXCEPTION_INIT позволяет обрабатывать ошибки по имени. Пример применения функции приведен ниже в разделе «SQLERRM».

SQLERRM

Встроенная функция, которая возвращает сообщение об ошибке SQL для текущего исключения. Пользователь может включить эту функцию внутрь инструкции WHEN OTHERS для обработки определенных ошибок по номеру. Директива компилятора EXCEPTION_INIT позволяет обрабатывать ошибки по имени.

Рассмотрим пример использования функций SQLERRM и SQLCODE:

```
CREATE TABLE err_test
   (widget_name   VARCHAR2(100)
   ,widget_count  NUMBER
   ,CONSTRAINT no_small_numbers CHECK
      (widget_count > 1000));
BEGIN
   INSERT INTO err_test (widget_name, widget_count)
   VALUES ('Athena',2);
EXCEPTION
   WHEN OTHERS THEN
   IF SQLCODE = -2290
      AND SQLERRM LIKE '%NO_SMALL_NUMBERS%'
   THEN
      DBMS_OUTPUT.PUT_LINE('widget_count is too small');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Exception not handled,'
         ||'SQLcode='||SQLCODE);
      DBMS_OUTPUT.PUT_LINE(SQLERRM);
   END IF;
END;
```

Когда PL/SQL выполняет этот код, выводится строка:

```
widget_count is too small
```

Директивы компилятора

Ключевое слово PRAGMA позволяет давать инструкцию компилятору. В PL/SQL существует четыре типа директив:

EXCEPTION_INIT

Сообщает компилятору о том, что следует сопоставить указанный номер ошибки с идентификатором, который был объявлен как EXCEPTION в текущей программе или в доступном пакете.

RESTRICT_REFERENCES

Сообщает компилятору *уровень чистоты* (*purity level*) программы пакета. Уровень чистоты показывает, в какой степени программа свободна от возможных побочных эффектов, связанных с чтением/записью таблиц БД и/или переменных пакета. С выходом версии Oracle8*i* необходимость в ней отпала.

SERIALLY_REUSABLE

Сообщает среде выполнения, что данные пакета не должны храниться в промежутках между обращениями к нему. Применяется для снижения требования к памяти, необходимой для одного пользователя, в случае если данные пакета нужны только на протяжении вызова, а не всего сеанса.

AUTONOMOUS_TRANSACTION

Сообщает компилятору о том, что функция, процедура, анонимный блок PL/SQL верхнего уровня, объектный метод или триггер БД выполняется в собственном пространстве транзакций. Появилась в Oracle8*i*.

Программные единицы

Язык программирования PL/SQL позволяет создавать разнообразные именованные программные единицы – контейнеры для кода. В их число входят:

Процедура

Программа, выполняющая одну или несколько команд.

Функция

Программа, выполняющая одну или несколько команд и возвращающая значение.

Пакет

Контейнер для процедур, функций и структур данных.

Триггер

Программа, выполняемая в ответ на изменения в БД.

Объектный тип

Oracle-версия класса в объектно-ориентированном языке; элементами объектных типов могут быть процедуры и функции. Объектный тип может содержать в своих методах код PL/SQL.

Общие атрибуты

Следующие ключевые слова могут применяться при определении процедуры или функции:

OR REPLACE

Предназначено для пересоздания существующей программной единицы с сохранением ее привилегий.

AUTHID

Определяет, с какими привилегиями будет выполняться программа, и задает имена, например, от имени владельца объекта (DEFINER) или от имени пользователя, выполняющего функцию (CURRENT_USER). До версии Oracle8*i* только встроенные пакеты DBMS_SQL и DBMS_UTILITY выполнялись от имени текущего пользователя. Значение AUTHID по умолчанию – DEFINER. Появилось в Oracle8*i*.

Хранимый SQL поддерживает две модели назначения привилегий в процессе выполнения. По умолчанию Oracle устанавливает привилегии владельца программы, т. е. пользователя, создавшего ее. При таком способе привилегии должны быть предоставлены пользователю напрямую – он не может наследовать их от роли.

Во втором варианте пользователь, вызывающий программу, делает это со своими собственными привилегиями. Анонимные блоки PL/SQL всегда выполняются с правами текущего пользователя. Чтобы создаваемая программа выполнялась с привилегиями вызвавшего ее пользователя, объявите ее с ключевыми словами AUTHID CURRENT_USER.

AGGREGATE USING

Требуется для агрегатных функций. Указывает серверу Oracle, что функция обрабатывает группу строк и возвращает единственный результат. Встроенная функция AVG представляет собой агрегатную функцию. Появилось в Oracle9*i*.

DETERMINISTIC

Требуется для *индексов по ключу-функции* (*function-based index*). Функция является детерминированной (DETERMINISTIC), если она всегда возвращает одно и то же значение при вызове с одинаковыми параметрами. Детерминированные функции не ссылаются на переменные пакета или БД. Встроенная функция INITCAP является детерминированной, а SYSDATE – нет. Появилось в Oracle8*i*.

PARALLEL_ENABLED [(PARTITION вх_парам BY {ANY | HASH | RANGE})]

Сообщает оптимизатору, что функция безопасна для параллельного выполнения. Инструкция PARTITION BY доступна только для функций, имеющих входной параметр REF CURSOR. Применяется для табличных функций и указывает оптимизатору, каким образом можно секционировать входные данные. Появилось в Oracle8*i*.

PIPELINED

Применяется для табличных функций. Сообщает серверу Oracle, что функция может начать возвращать данные по мере их генерирования вместо того, чтобы вернуть все данные сразу после завершения обработки. Появилось в Oracle9*i*.

Процедуры

```
CREATE [OR REPLACE] PROCEDURE имя
   [ (параметр [,параметр]) ]
   [AUTHID {CURRENT_USER | DEFINER} ]
   [DETERMINISTIC]
{IS | AS}
   секция_объявления
BEGIN
   секция_выполнения
[EXCEPTION
   секция_исключений]
END [имя];
```

Программная единица или модуль, который выполняет одну или более команд и может получать или возвращать значения посредством списка параметров. Процедуру можно вызвать как автономную исполняемую команду PL/SQL, например:

```
apply_discount(new_company_id, 0.15) -- скидка 15%
```

Общие ключевые слова: AGGREGATE USING, AUTHID, DETERMINISTIC, PARALLEL_ENABLE и PIPELINED.

Функции

```
CREATE [OR REPLACE] FUNCTION имя
   [ (параметр [,параметр]) ]
   RETURN return_тип_данных
   [AUTHID {CURRENT_USER | DEFINER} ]
   [DETERMINISTIC]
   [PARALLEL_ENABLED]
   [PIPELINED]
   [AGGREGATE USING]
{IS | AS}
   [секция_объявления]
BEGIN
   секция_выполнения
[EXCEPTION
   секция_исключений]
END [имя];
```

Программная единица или модуль, который выполняет одну или более команд и возвращает значение при помощи инструкции RETURN. Как и процедуры, функции могут получать одно или более значений через список параметров. Функция должна содержать в секции выполнения хотя бы одну команду RETURN. Инструкция RETURN в заголовке функции задает тип данных возвращаемого значения.

Функция может вызываться в любом месте, где возможно использование выражения того же типа. Вызывать функцию можно следующими способами:

- В команде присваивания:

```
sales95 := tot_sales(1995,'C');
```

- При задании значения по умолчанию:

```
DECLARE
 sales95 NUMBER DEFAULT tot_sales(1995,'C');
BEGIN
```

- В логическом выражении:

```
IF tot_sales(1995,'C') > 10000
THEN
...
```

- В команде SQL:

```
SELECT first_name ,surname
   FROM sellers
WHERE tot_sales(1995,'C') > 1000;
```

- Как аргумент в списке параметров другой программной единицы.

Общие ключевые слова: AUTHID, DETERMINISTIC, PARALLEL_ENABLE, AGGREGATE USING и PIPELINED.

Табличные функции

Функции, принимающие на входе коллекцию или REF CURSOR (набор строк) и возвращающие на выходе коллекцию записей (набор строк). Табличные функции появились в Oracle8*i*. Начиная с Oracle9*i* существует возможность применять команду PIPE ROW для идентификации входного и выходного потоков. Такая организация потоков позволяет организовывать конвейер из нескольких функций, избегая необходимости хранения промежуточных таблиц. Табличные функции обычно появляются в инструкции FROM запроса.

Локальные программы

Процедура или функция, определенная в секции объявлений блока PL/SQL. Объявление локальной программы должно находиться в конце секции объявлений, после всех объявлений типов, записей, курсоров, переменных и исключений. На программу, определенную в секции объявлений, можно ссылаться только внутри секций выполнения и исключений того же блока. Она не определена вне данного блока.

Следующая программа определяет локальную процедуру и локальную функцию:

```
PROCEDURE track_revenue
IS
   PROCEDURE calc_total (year_in IN INTEGER) IS
   BEGIN
      calculations here ...
   END;

   FUNCTION below_minimum (comp_id IN INTEGER)
      RETURN BOOLEAN
   IS
   BEGIN
      ...
   END;
```

Локальные программы могут быть перегружены, при этом действуют те же правила, что и для перегруженных программ пакетов.

Перегрузка программ

PL/SQL позволяет определить внутри любой секции объявлений (в том числе в спецификации или теле пакета, о чем мы поговорим в следующем разделе) несколько программ с одинаковыми именами. Такой прием называется *перегрузкой*. Если две или более программ имеют одно и то же имя, они должны отличаться чем-то другим, чтобы компилятор мог определить, которую из них следует использовать.

Рассмотрим пример перегруженных программ в одной из спецификаций встроенного пакета Oracle:

```
PACKAGE DBMS_OUTPUT
IS
   PROCEDURE PUT_LINE (a VARCHAR2);
   PROCEDURE PUT_LINE (a NUMBER);
   PROCEDURE PUT_LINE (a DATE);
END;
```

Все процедуры PUT_LINE идентичны во всем, кроме типа данных параметра. Для компилятора такого различия достаточно.

Для успешной перегрузки необходимо, чтобы было выполнено хотя бы одно из приведенных ниже условий:

- Наборы параметров должны отличаться по типам данных (числовой, символьный, дата-и-время, логический).
- Должны отличаться типы программ (можно создать функцию и процедуру с одинаковыми именами и идентичными списками параметров).
- Количество параметров должно быть разным.

Программы не удастся перегрузить, если:

- Отличаются только типы данных в инструкции RETURN для функций.
- Типы данных параметров относятся к одному семейству (CHAR и VARCHAR2, NUMBER и INTEGER и т. д.).
- Отличаются только режимом использования параметров.

Пакеты

Пакет (*package*) – это группа элементов кода PL/SQL. Пакет может включать в себя такие элементы, как процедуры, функции, константы, переменные, курсоры, имена исключений и команды TYPE (для ассоциативных массивов, записей, REF CURSOR и т. д.).

Структура пакета

Пакет может состоять из двух частей: спецификации пакета и тела пакета, которым и посвящены следующие разделы.

Спецификация пакета

```
CREATE [OR REPLACE] PACKAGE имя_пакета
[ AUTHID {CURRENT_USER | DEFINER} ]
{IS | AS}
   [определения публичных типов
  , объявления публичных переменных, типов и объектов
  ,объявления исключений
  ,прагмы
  ,объявления курсоров, процедур и функций
    ,заголовки процедур и функций]

END [имя_пакета];
```

Перечисляет все объекты, которые доступны для общего использования в приложениях. Кроме того, спецификация пакета предоставляет всю информацию, необходимую разработчику для использования объектов пакета; можно сказать, что спецификация пакета представляет собой его API (Application Programming Interface). Спецификация пакета необходима. В случае если спецификация или пакет не содержат никаких процедур и функций и нет необходимости в закрытом коде, то тело пакету не требуется.

Тело пакета

```
CREATE [OR REPLACE] PACKAGE BODY имя_пакета
{IS | AS}

   [определения закрытых типов
   ,объявления закрытых переменных, типов и объектов
   ,полные определения курсоров
   ,полные определения процедур и функций]

[BEGIN
   выполняемые_команды

[EXCEPTION
   обработчики исключений ] ]

END [имя_пакета];
```

Содержит весь код, необходимый для реализации процедур, функций и курсоров, перечисленных в спецификации, а также все закрытые объекты (если такие имеются), доступные только другим элементам, определенным в данном пакете, и необязательную секцию инициализации.

Объявления из спецификации не могут быть повторены в теле пакета. Секции выполнения и обработки исключений необязательны для тела пакета. Само тело пакета может содержать выполняемую секцию, которая следует за объявлением процедур и функций внутри тела пакета. Если секция выполнения присутствует, то она вызывает *секцию инициализации* (*initialization section*), выполняемую всего один раз – при первом обращении к какому-либо элементу пакета в рамках сеанса.

Спецификация пакета должна быть скомпилирована раньше его тела. При назначении привилегии EXECUTE для пакета другой схеме или всем пользователям (PUBLIC) доступ предоставляется только к спецификации, тело остается скрытым.

Рассмотрим пример пакета:

```
CREATE OR REPLACE PACKAGE time_pkg IS
   FUNCTION  GetTimestamp  RETURN DATE;
   PRAGMA RESTRICT_REFERENCES (GetTimestamp, WNDS);

   PROCEDURE ResetTimestamp;
END time_pkg;

CREATE OR REPLACE PACKAGE BODY time_pkg IS
   StartTimeStamp   DATE := SYSDATE;
   -- StartTimeStamp - это данные пакета.

   FUNCTION GetTimestamp RETURN DATE IS
   BEGIN
      RETURN StartTimeStamp;
   END GetTimestamp;

   PROCEDURE ResetTimestamp IS
   BEGIN
      StartTimeStamp := SYSDATE;
   END ResetTimestamp;

END time_pkg;
```

Обращение к элементам пакета

На элементы, объявленные в спецификации, из вызывающего приложения следует ссылаться при помощи точечной нотации:

```
имя_пакета.элемент_пакета
```

Например, встроенный пакет DBMS_OUTPUT содержит процедуру с именем PUT_LINE, и обращение к ней будет выглядеть следующим образом:

```
DBMS_OUTPUT.PUT_LINE('Эта строка - параметр');
```

Данные пакета

Структуры данных, объявленные в спецификации или теле пакета, но вне каких бы то ни было процедур и функций, называются *данными пакета* (*package data*). Областью действия данных пакета является весь сеанс, для них не существует границ транзакций, поэтому они по сути своей являются глобальными для вашей программы.

При работе с данными пакета не забывайте о следующем:

- На состояние переменных пакета не влияют команды COMMIT и ROLLBACK.
- Объявленный в пакете курсор – глобальный. Он остается открытым до тех пор, пока не будет закрыт явно или же до конца сеанса.
- Хорошей практикой следует считать сокрытие структур данных в теле пакета и создание программ для чтения и записи этих данных.

Директива компилятора SERIALLY_REUSABLE

Если необходимо, чтобы данные пакета существовали только на протяжении вызова функции или процедуры пакета, но не между вызовами в текущем сеансе, то вы можете сэкономить память, используя директиву компилятора SERIALLY_REUSABLE. Ее необходимо включить как в спецификацию, так и в тело пакета. После каждого вызова PL/SQL будет закрывать курсоры и освобождать занятую пакетом память.

Этот прием применим только для больших сообществ пользователей, выполняющих одну и ту же программу. Обычно требования к оперативной памяти сервера БД возрастают линейно в зависимости от количества пользователей. Если указать директиву SERIALLY_REUSABLE, этот рост может замедлиться за счет того, что рабочие области пакетов хранятся в пуле SGA и совместно расходуются всеми пользователями.

Инициализация пакета

Когда пользователь первый раз ссылается на элемент пакета, весь пакет загружается в SGA экземпляра БД, к которому подключен пользователь. Этот код совместно используется всеми сеансами, имеющими привилегию EXECUTE на пакет.

Затем все данные пакета вносятся в глобальную (User Global Area – UGA), закрытую область SGA или PGA (Program Global Area). Если тело пакета содержит секцию инициализации, этот код будет выполнен. Секция инициализации может отсутствовать и размещается в конце тела пакета, начинается с команды BEGIN и заканчивается секцией обработки исключений (если таковая присутствует) или же оператором конца пакета (END).

Триггеры

Триггер (*trigger*) – это программа, выполняемая в ответ на изменение данных или на определенное событие, происходящее в БД. Существует предопределенный набор событий, которые могут быть связаны с триггерами, что позволяет объединять пользовательскую обработку с обработкой, которую выполняет сама СУБД. Говорят, что *инициирующее событие* (*triggering event*) запускает или выполняет триггер.

Триггеры включаются при создании (посредством команды CREATE TRIGGER) и могут быть отменены (с тем, чтобы они не срабатывали) при помощи команды ALTER TRIGGER или ALTER TABLE:

```
ALTER TRIGGER имя_триггера {ENABLE | DISABLE};
ALTER TABLE имя_таблицы {ENABLE | DISABLE} ALL
   TRIGGERS;
```

Нельзя создавать триггеры для объектов, принадлежащих пользователю SYS.

CREATE TRIGGER

```
CREATE [OR REPLACE] TRIGGER имя_триггера
    {BEFORE | AFTER | INSTEAD OF} иниц_событие
ON
   [ NESTED TABLE столбец_вложенной_таблицы OF представление ]
      | ссылка_на_таблицу_или_представление | DATABASE [инструкция_ссылок]
[FOR EACH ROW [WHEN условие_триггера]]
тело_триггера;
```

Создает триггер. Тело триггера, включенное в команду CREATE TRIGGER, – это стандартный блок PL/SQL.

Ключевые слова

BEFORE|AFTER

Триггеры могут запускаться до (BEFORE) или после (AFTER) инициирующего события. Триггеры AFTER чуть более эффективны, чем триггеры BEFORE.

INSTEAD OF

Обычно используется с представлениями для того, чтобы разрешить обновление базовых таблиц представления при помощи команды INSERT, UPDATE или DELETE.

иниц_событие

Одно из перечисленных ниже событий:

INSERT

Срабатывает при каждом добавлении строки в таблицу или представление, заданные параметром *ссылка_на_таблицу_или_представление*.

UPDATE

Срабатывает при каждом изменении таблицы или представления, заданного параметром *ссылка_на_таблицу_или_представление* посредством UPDATE. UPDATE-триггеры могут дополнительно содержать инструкцию OF для того, чтобы запретить срабатывание при обновлении определенных (указанных в этой инструкции) столбцов.

DELETE

Срабатывает при каждом удалении строки из таблицы или представления, заданного параметром *ссылка_на_таблицу_или_представление*. Не срабатывает в случае удаления командой TRUNCATE.

CREATE

Срабатывает при каждом добавлении нового объекта в БД посредством команды CREATE. В данном контексте объекты понимаются как, например, таблицы или пакеты (все то, что находится в ALL_OBJECTS). Может применяться к одной схеме или ко всей БД.

ALTER

Срабатывает при каждом изменении объекта БД посредством команды ALTER. В данном контексте объекты понимаются как, например, таблицы или пакеты (все то, что находится в ALL_OBJECTS). Может применяться к одной схеме или ко всей БД.

DROP

Срабатывает при каждом удалении объекта из БД посредством команды DROP. В данном контексте объекты понимаются как, например, таблицы или пакеты (все то, что находится в ALL_OBJECTS). Может применяться к одной схеме или ко всей БД.

SERVERERROR

Срабатывает при записи серверного сообщения об ошибке. Для таких событий разрешены только триггеры AFTER.

LOGON

Срабатывает при создании сеанса (подключении пользователя к БД). Для таких событий разрешены только триггеры AFTER.

LOGOFF

Срабатывает при закрытии сеанса (отключении пользователя от БД). Для таких событий разрешены только триггеры BEFORE.

STARTUP

Срабатывает при открытии БД. Для таких событий разрешены только триггеры AFTER.

SHUTDOWN

Срабатывает при закрытии БД. Для таких событий разрешены только триггеры BEFORE.

инструкция_ссылок

Разрешена только для таких событий, как INSERT, UPDATE и DELETE. Позволяет дать имя не по умолчанию для старой и новой псевдозаписей. Эти псевдозаписи обеспечивают для программы видимость значений, которые были до обновления БД и которые будут после обновления БД. Применяется для триггеров, действующих на уровне строк. Такие записи определяются как записи %ROWTYPE с той лишь разницей, что запрещены ссылки на столбцы типа LONG и LONG RAW. Для них применяется точечная нотация, в теле триггера они предваряются двоеточием. В отличие от других записей, для полей разрешено только индивидуальное присваивание. Для триггеров INSERT все старые поля содержат NULL, для триггеров DELETE все новые поля содержат NULL.

FOR EACH ROW

Означает, что триггер действует на уровне строк. Такие триггеры запускаются по одному разу для каждой обработанной строки. По умолчанию триггеры действуют на уровне команд, т. е. запускаются по одному разу для каждой инициирующей команды.

WHEN условие_триггера

Задает условие, которое должно быть выполнено, для того чтобы триггер сработал. В условии для запуска триггера не могут применяться хранимые процедуры и функции.

тело_триггера

Стандартный блок PL/SQL.

Последовательности событий

Из предыдущего раздела вы узнали, что для одной таблицы может быть определено множество различных видов триггеров. Необходимо понимать, в каком порядке они срабатывают.

Основное влияние на последовательность выполнения оказывает конструкция FOR EACH ROW. Ключевые слова FOR EACH ROW указывают, для строки или для таблицы будет запускаться триггер. Другие факторы также влияют на порядок выполнения триггеров, о чем и будет рассказано в следующих разделах.

События DML

К событиям DML относятся команды INSERT, UPDATE и DELETE для таблицы или представления. Для этих событий могут быть определены триггеры уровня команды (только для таблиц) или уровня строки и запускаться они могут как до, так и после инициирующего события. Триггер BEFORE может изменять данные в обрабатываемых строках и обычно применяется для того, чтобы определить, должна ли выполняться инициировавшая его команда. Триггеры AFTER не выполняют это дополнительное логическое чтение, поэтому выполняются немного быстрее, но зато они не могут изменять новые значения.

Триггеры DML, если они определены, запускаются в следующем порядке:

1. Триггер BEFORE уровня команды
2. Для каждой строки, обработанной командой
3. Триггер BEFORE уровня строки
4. Инициирующая команда
5. Триггер AFTER уровня строки
6. Триггер AFTER уровня команды

События DDL

К событиям DDL относятся CREATE, ALTER и DROP. Эти триггеры срабатывают при каждом выполнении соответствующей команды DDL. DDL-триггеры могут применяться к какой-то одной схеме или же ко всей БД. Появились в Oracle8*i*.

События БД

К событиям базы данных относятся SERVERERROR, LOGON, LOGOFF, STARTUP и SHUTDOWN. Для событий LOGOFF и SHUTDOWN разрешены только триггеры BEFORE. Для событий LOGON, STARTUP и SERVERERROR разрешены только триггеры AFTER. Триггер SHUTDOWN срабатывает в случае событий SHUTDOWN NORMAL и SHUTDOWN IMMEDIATE, но не SHUTDOWN ABORT. Появились в Oracle8*i*.

Предикаты триггера

Если один триггер применяется для нескольких событий, то в условии триггера следует применять предикаты INSERTING, UPDATING и DELETING для идентификации инициирующего события. Рассмотрим пример, иллюстрирующий применение предикатов триггера:

```
CREATE OR REPLACE TRIGGER emp_log_t
   AFTER INSERT OR UPDATE OR DELETE ON emp
   FOR EACH ROW
DECLARE
   dmltype  CHAR(1);
BEGIN
   IF INSERTING THEN
      dmltype := 'I';
      INSERT INTO emp_log (emp_no, who, operation)
         VALUES (:new.empno, USER, dmltype);
   ELSIF UPDATING  THEN
      dmltype := 'U';
      INSERT INTO emp_log (emp_no, who, operation)
         VALUES (:new.empno, USER, dmltype);
   END IF;
END;
```

Вызов функций PL/SQL в SQL

Хранимые функции могут вызываться из команд SQL подобно тому, как вызываются встроенные функции, такие как DECODE, NVL или RTRIM. При осуществлении таких вызовов необходимо помнить о ряде мер предосторожности и ограничений, которым будут посвящены следующие разделы.

Определение функции

Предложенный ниже формат применяется для вызова собственной функции из SQL. Такая возможность позволяет «переделать» язык SQL в соответствии с требованиями конкретного приложения.

Синтаксис

```
[имя_схемы.][имя_пакета.]имя_функции[@db_link]
   [список_параметров]
```

Ключевые слова

имя_схемы Пользователь/владелец функции или пакета (может отсутствовать).

имя_пакета Пакет, содержащий вызываемую функцию (может отсутствовать).

имя_функции Имя функции.

db_link

Имя канала связи с удаленной БД, содержащей функцию (не обязательно).

список_параметров

Список параметров функции (не обязателен).

Требования и ограничения

К вызову хранимых функций в SQL предъявляется ряд требований:

- Все параметры должны быть только входными, параметры IN OUT и OUT не разрешены.
- Типы данных параметров функции и возвращаемый тип (RETURN) должны быть совместимы с типами данных СУБД. Аргументы и возвращаемое значение не могут относиться к типу BOOLEAN, типу пользовательских записей, быть ассоциативным массивом и т. д.
- Для передаваемых функции параметров должно использоваться позиционное представление; представление по имени не поддерживается.
- Функция должна храниться в БД, а не в локальной программе.

Директива компилятора RESTRICT_REFERENCES

До выхода Oracle8*i* было необходимо указывать *уровень чистоты* (*purity level*) для процедуры или функции, которая прямо или косвенно используется в команде SQL. Начиная с Oracle8*i*, если уровень чистоты не указан, процессор PL/SQL определяет его автоматически. Директива компилятора RESTRICT REFERENCES все еще поддерживается для обеспечения обратной совместимости версий, но применять ее в Oracle9*i* не рекомендуется.

Директива компилятора RESTRICT_REFERENCES указывает уровень чистоты и имеет такой синтаксис:

```
PRAGMA RESTRICT_REFERENCES (имя_программы | DEFAULT, уровень_чистоты);
```

Ключевое слово DEFAULT применяется ко всем методам объектного типа или ко всем программам пакета.

Можно определить от одного до пяти уровней чистоты в любом порядке в списке через запятую. Уровень чистоты указывает, насколько программа или метод свободны от побочных эффектов (перечень возможных побочных эффектов и соответствующих уровней чистоты предложен в табл. 9.3).

Таблица 9.3. Уровни чистоты и побочные эффекты

Уровень чистоты	Описание	Ограничение
WNDS	Не пишет в БД	Не выполняет команды INSERT, UPDATE и DELETE
RNDS	Не читает БД	Не выполняет команду SELECT
WNPS	Не пишет в пакет	Не изменяет переменные пакета
RNPS	Не читает пакет	Не читает переменные пакета
TRUST		Не вводит объявленные ограничения, но благодаря ему компилятор считает, что они выполнены

Тонкости вызова функций PL/SQL из SQL

Основная трудность, связанная с выполнением хранимых функций из SQL, состоит в том, что по умолчанию они не соответствуют модели целостности по чтению на уровне команд для БД. Если команда SQL и все хранимые функции этой команды не включены в одну и ту же транзакцию с обеспечением целостности по чтению, то каждое исполнение хранимой функции может обращаться к различным согласованным по времени наборам данных. Для того чтобы избежать возможных неприятностей, необходимо программно гарантировать целостность по чтению, выдав команду SET TRANSACTION READ ONLY или SET TRANSACTION ISOLATION LEVEL SERIALIZABLE перед выполнением команды SQL, содержащей хранимую функцию. Для завершения этой целостной по чтению транзакции необходимо после команды SQL выполнить команду COMMIT или ROLLBACK.

Если имя функции совпадает с именем столбца таблицы в команде SELECT и у функции нет параметров, то приоритет столбца выше приоритета функции. Для того чтобы вынудить СУБД распознать имя функции, предварите его именем схемы.

Компиляция PL/SQL в двоичный код

Начиная с Oracle9*i* работу многих программ PL/SQL можно ускорить за счет компиляции хранимых программ в двоичный код. Сервер Oracle преобразует программу PL/SQL в код C и создаст из него разделяемую библиотеку (DLL для Windows). Для поддержки компиляции необходимо, чтобы на машине сервера БД был установлен поддерживаемый компилятор C.

Далее приводятся основные этапы компиляции программы PL/SQL в двоичный код, а за дополнительной информацией по этому вопросу можно обратиться к документации Oracle:

1. Отредактировать make-файл *spnc_makefile.mk*, который должен храниться в подкаталоге *$ORACLE HOME/plsql*.

   Установить параметр инициализации PLSQL_COMPILER_FLAGS = 'NATIVE'. Кроме того, может потребоваться установка следующих дополнительных параметров (см. главу 2):

   ```
   PLSQL_NATIVE_C_COMPILER
   PLSQL_NATIVE_LINKER
   PLSQL_NATIVE LIBRARY_DIR
   PLSQL_NATIVE_MAKE_UTILITY
   PLSQL_NATIVE_MAKE_FILE_NAME
   ```

   Их можно установить в файлах *SPFILE* или *INIT.ORA* либо при помощи команды ALTER SYSTEM.

2. Создать или заменить хранимые программы.
3. Проверить компиляцию, запрашивая представление словаря данных USER_STORED_SETTINGS и определяя местоположение разделяемой библиотеки или DLL в файловой системе сервера БД.

Внешние процедуры

Внешние процедуры обеспечивают механизм вызова программы, не относящейся к БД (такой как DLL для Windows или разделяемая библиотека для Unix), из программы PL/SQL. Для каждого сеанса, вызывающего внешнюю процедуру, прослушивающий процесс запускает собственный процесс extproc. Этот процесс extproc запускается при первом вызове внешней процедуры и завершается при закрытии сеанса. Для разделяемой библиотеки должна быть создана соответствующая библиотека в БД.

Рассмотрим вкратце те шаги, которые необходимо выполнить для создания внешней процедуры. Подробные сведения и примеры внешних процедур приведены в главе 23 книги «Oracle PL/SQL Programming».

1. Настроить прослушивающий процесс.

   Для внешних процедур необходим прослушивающий процесс. Если вы используете прослушивающий процесс Oracle Net Services, он может применяться и как прослушивающий процесс extproc, хотя для обеспечения большей безопасности можно разделить эти два прослушивающих процесса и запускать последний от имени пользователя с ограниченным набором привилегий. Подробная информация о конфигурировании прослушивающего процесса содержится в руководствах «Oracle9*i* Administrators' Guide» (Oracle9*i*. Руководство администратора) и «Oracle9*i* Net Services Administrators' Guide» (Oracle9*i*. Руководство администратора Net Services).

2. Определить или создать разделяемую библиотеку или DLL.

   Этот этап не имеет ничего общего с PL/SQL и может не иметь ничего общего и с СУБД. Необходимо написать собственные программы на C и собрать их в разделяемую библиотеку или DLL или же использовать функции или процедуры существующей библиотеки. Далее будет рассмотрен простой пример существующих вызовов генератора случайных чисел, имеющегося в операционной системе.

3. Создать библиотеку в БД.

   Создать в БД библиотеку для разделяемой библиотеки или DLL при помощи команды CREATE LIBRARY (рассматривалась в главе 7).

   Для удаления библиотек из БД применяется команда DROP LIBRARY.

   Для обращения к исполняемой функции rand из библиотеки C не надо писать программу на C, т. к. функция уже включена в разделяемую библиотеку и типы ее аргументов имеют прямые соответствия в PL/SQL. Если функция rand находится в стандартной разделяемой библиотеке */lib/libc.so*, то, как в случае с Solaris, надо выполнить следующую команду CREATE LIBRARY:

   ```
   CREATE OR REPLACE LIBRARY libc_l AS
      '/lib/libc.so';  -- Ссылка на библиотеку C.
   ```

   Аналогичная команда в Windows имеет такой вид:

   ```
   CREATE OR REPLACE LIBRARY libc_l AS
      'C:\WINDOWS\SYSTEM32\CRTDLL.DLL';
   ```

4. Создать обертку PL/SQL для внешней процедуры.

Для реализации этого шага необходимо специальным образом применить команду CREATE PROCEDURE, описываемую далее.

CREATE PROCEDURE

```
CREATE [OR REPLACE] PROCEDURE имя_процедуры
   [список_параметров]
{IS | AS} LANGUAGE C
   [NAME внешнее_имя]
   LIBRARY имя_библиотеки
   [AGENT IN (имя_агента)]
   [WITH CONTEXT]
   [PARAMETERS (список_внешних_параметров)];
```

Создает процедуру, в данном случае обертку для внешней процедуры.

Ключевые слова

имя_процедуры

Имя создаваемой процедуры-обертки.

LANGUAGE C

Язык, на котором написана внешняя программа (по умолчанию – C).

имя_библиотеки

Имя библиотеки, создаваемой при помощи команды CREATE LIBRARY.

внешнее_имя

Имя внешней программы, с которым она появляется в библиотеке. По умолчанию совпадает с именем пакета обертки. Имена пакетов PL/SQL обычно сохраняются в верхнем регистре, поэтому может потребоваться заключение внешнего имени в двойные кавычки (для сохранения регистра).

AGENT IN (имя_агента)

Указывает имя агента в виде формального параметра PL/SQL в спецификации вызова.

WITH CONTEXT

Применяется для передачи указателя контекста во внешнюю программу, чтобы она могла совершить обратный вызов к БД через Oracle Call Interface (OCI).

PARAMETERS

Список внешних параметров, разделяемых запятыми, определяющий позицию и тип параметров, передаваемых внешней программе. Назначение этой инструкции в том, чтобы разобраться с несоответствиями в способах обработки переменных PL/SQL и C. Для каждого параметра списка применяется такой синтаксис:

```
{CONTEXT | RETURN | параметр_имя [свойство]}
   [BY REFERENCE] [внешний_тип_данных]
```

где:

CONTEXT

Указывает позицию в списке параметров, в которую передается указатель контекста. Необходимо, если для передачи указателя контекста вызываемой программе применяется инструкция WITH CONTEXT. Параметр CONTEXT должен быть первым в списке внешних параметров. Если он указан, то необязательные разделы *свойство*, BY REFERENCE и *внешний_тип_данных* не действуют.

RETURN

Указывает, что описания относятся к возвращаемому из внешней программы значению. По умолчанию этот параметр передается по значению. Для передачи параметра по ссылке (как указателя) указываются ключевые слова BY REFERENCE.

имя_параметра

Имя формального параметра PL/SQL. По умолчанию входные формальные параметры передаются по значению. Для передачи параметра по ссылке (как указателя) указываются ключевые слова BY REFERENCE. Формальные параметры IN OUT и OUT всегда передаются по ссылке.

свойство

Разворачивается в следующую конструкцию:

```
INDICATOR [STRUCT | TDO ] | LENGTH | MAXLEN | CHARSETID | CHARSETFORM | SELF
```

INDICATOR указывает, содержит ли соответствующий параметр NULL. В C-программе, если индикатор равен константе OCI_IND_NULL, то параметр – NULL. Если индикатор равен константе OCI_IND_NOTNULL, то параметр – не-NULL. Для параметров IN передача INDICATOR осуществляется по значению (поведение по умолчанию). Для параметров IN OUT, OUT и RETURN передача INDICATOR осуществляется по ссылке.

Можно передать внешней процедуре пользовательский тип. Для этого обычно передаются три параметра: фактическое значение объекта, параметр TDO (Type Descriptor Object), как он определен в C посредством Oracle Type Translator; и параметр INDICATOR STRUCT, который предназначен для того, чтобы проверить, содержит ли объект значение NULL.

Аргументы LENGTH и MAXLEN служат для передачи текущей и максимальной длины строки или типа RAW. Для входных параметров LENGTH передается по значению (по умолчанию). Для IN OUT, OUT и RETURN параметров LENGTH передается по ссылке. MAXLEN не действует для параметров IN. Для параметров IN OUT, OUT и RETURN MAXLEN передается по ссылке и доступно только для чтения.

CHARSETID и CHARSETFORM применяются для поддержки национальных наборов символов. Они аналогичны OCI-атрибутам OCI_ATTR_CHARSET_ID и OCI_ATTR_CHARSET_FORM. Для входных параметров CHARSETID и CHARSETFORM передаются по значению (по умолчанию) и доступны только для чтения. Для IN OUT, OUT и RETURN параметров CHARSETID и CHARSETFORM передаются по ссылке и доступны только для чтения.

SELF указывается, если функция-член объекта реализована как внешний вызов, а не как программа PL/SQL.

Функция-обертка PL/SQL обычно находится в пакете.

Основываясь на предыдущем примере с генератором случайных чисел, можно создать пакет следующим образом:

```
CREATE OR REPLACE PACKAGE random_utl
AS
   FUNCTION rand RETURN PLS_INTEGER;
   PRAGMA RESTRICT_REFERENCES(rand,WNDS,RNDS,WNPS,RNPS);

   PROCEDURE srand (seed IN PLS_INTEGER);
```

```
   PRAGMA RESTRICT_REFERENCES(srand,WNDS,RNDS,WNPS,RNPS);
END random_utl;

CREATE PACKAGE BODY random_utl
AS
   FUNCTION rand RETURN PLS_INTEGER
   IS
      LANGUAGE C     -- Язык программы.
      NAME "rand"    -- Имя функции в библиотеке.
      LIBRARY libc_l -- Созданная выше библиотека.

   PROCEDURE srand (seed IN PLS_INTEGER)
   IS
      LANGUAGE C
      NAME "srand"           -- В этой библиотеке имя набрано в нижнем регистре.
      LIBRARY libc_l
      PARAMETERS (seed ub4); -- Отображение на целое число без знака.
END random_utl;
```

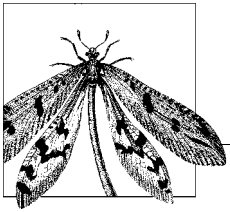
Для того чтобы использовать эту внешнюю функцию – генератор случайных чисел, просто вызываем процедуру пакета *srand* для выбора начальной точки генерации, а затем функцию пакета *rand* для получения случайных чисел:

```
DECLARE
   random_nbr  PLS_INTEGER;
   seed        PLS_INTEGER;
BEGIN
   SELECT TO_CHAR(SYSDATE,'SSSSS') INTO seed
      FROM dual;
   random_utl.srand(seed);            -- Задание начального числа для генератора.

   random_nbr := random_utl.rand;     -- Получение числа.
   DBMS_OUTPUT.PUT_LINE('number='||random_nbr);

   random_nbr := random_utl.rand;     -- Получение числа.
   DBMS_OUTPUT.PUT_LINE('number='||random_nbr);
END;
```

Конечно, то же самое можно сделать при помощи встроенного пакета DBMS_RANDOM, описанного в главе 10.

Java и PL/SQL

Информация о применении Java совместно с PL/SQL приведена в главе 11.



10
Пакеты PL/SQL

Пакет (*package*) представляет собой набор сгруппированных вместе объектов PL/SQL или Java. Пакеты могут содержать процедуры, функции, переменные, константы и прочие объекты.

Применение механизма пакетов позволяет Oracle расширять функциональность базы данных с помощью встроенных пакетов (написанных на PL/SQL), к которым разработчики могут обращаться из своих PL/SQL-программ. Многие из новых возможностей Oracle реализуются в форме пакетов. Например, компонент Advanced Queuing использует пакеты DBMS_AQ и DBMS_AQADM. Некоторые пакеты, такие как DBMS_STATISTICS, предлагают улучшенную реализацию часто применяемых возможностей.

О создании пользователями собственных пакетов рассказывается в главе 9.

Эта глава содержит краткое описание процедур и функций пакетов, поставляемых с СУБД Oracle. Пакеты перечислены в алфавитном порядке. Для каждого пакета приведен перечень интерфейсов процедур и функций с кратким описанием передаваемых данному модулю параметров. В этой книге необязательные параметры обычно заключены в квадратные скобки ([ ]). В данной главе параметры, применяемые только в Oracle9*i*, обозначаются специальным образом: они взяты в квадратные скобки, за которыми следует знак решетки (#) – это отличает их от обычных необязательных параметров.

Более подробные сведения о перечисленных пакетах и их применении можно найти в документации Oracle. Пакеты, вошедшие в Oracle8, прекрасно описаны в книге «Oracle Built-in Packages» (Встроенные пакеты Oracle) Стивена Фейерштейна (Steven Feuerstein), Чарльза Дая (Charles Dye) и Джона Бересниевича (John Beresniewicz), вышедшей в издательстве O'Reilly & Associates.

DBMS_ALERT

Предоставляет средство синхронизации – основанный на транзакциях механизм уведомления сеансов о наступлении определенных событий в базе данных.

Вызовы

```
PROCEDURE DBMS_ALERT.REGISTER
   (name IN VARCHAR2);
```

Регистрирует вызывающий сеанс в качестве получателя сигнала *name*.

```
PROCEDURE DBMS_ALERT.REMOVE
   (name IN VARCHAR2);
```

Отменяет регистрацию вызывающего сеанса в качестве получателя сигнала *name*.

```
PROCEDURE DBMS_ALERT.REMOVEALL;
```

Отменяет регистрацию вызывающего сеанса в качестве получателя для всех сигналов. Появился в Oracle8*i*.

```
PROCEDURE DBMS_ALERT.SET_DEFAULTS
   (sensitivity IN NUMBER);
```

Устанавливает период опроса в секундах для вызывающего сеанса.

```
PROCEDURE DBMS_ALERT.SIGNAL
   (name IN VARCHAR2,
    message IN VARCHAR2);
```

Посылает сигнал *name*, присоединяя к нему сообщение *message*. Сеансы, зарегистрировавшиеся для получения сигнала *name*, получают уведомление, только если сгенерировавшая сигнал транзакция зафиксирована.

```
PROCEDURE DBMS_ALERT.WAITANY
   (name OUT VARCHAR2,
    message OUT VARCHAR2,
    status OUT INTEGER,
    timeout IN NUMBER DEFAULT MAXWAIT);
```

Ждет в течение *timeout* секунд уведомления о каких-либо сигналах, в качестве получателя которых зарегистрирован данный сеанс. Если *status* = 0, то параметры *name* и *message* содержат информацию о сигнале. Если *status* = 1, это означает, что в течение *timeout* секунд не было никаких уведомлений о сигналах.

```
PROCEDURE DBMS_ALERT.WAITONE
   (name IN VARCHAR2,
    message OUT VARCHAR2,
    status OUT INTEGER,
    timeout IN NUMBER DEFAULT MAXWAIT);
```

Ждет в течение *timeout* секунд уведомления о сигнале *name*.

DBMS_APPLICATION_INFO

Позволяет приложениям зарегистрировать их текущий статус выполнения в различных виртуальных таблицах V$ словаря данных Oracle.

Вызовы

```
PROCEDURE DBMS_APPLICATION_INFO.READ_CLIENT_INFO
   (client_info OUT VARCHAR2);
```

Возвращает зарегистрированную на настоящий момент для сеанса информацию о клиенте *client_info*.

```
PROCEDURE DBMS_APPLICATION_INFO.READ_MODULE
   (module_name OUT VARCHAR2,
    action_name OUT VARCHAR2);
```

Возвращает зарегистрированные на настоящий момент для сеанса *module_name* и *action_name*.

```
PROCEDURE DBMS_APPLICATION_INFO.SET_ACTION
   (action_name IN VARCHAR2);
```

Регистрирует для сеанса *action_name* в V$SESSION и V$SQLAREA как текущее действие.

```
PROCEDURE DBMS_APPLICATION_INFO.SET_CLIENT_INFO
   (client_info IN VARCHAR2);
```

Регистрирует для сеанса *client_info* в V$SESSION как текущую информацию о клиенте.

```
PROCEDURE DBMS_APPLICATION_INFO.SET_MODULE
   (module_name IN VARCHAR2,
    action_name IN VARCHAR2);
```

Регистрирует для сеанса *module_name* и *action_name* в V$SESSION и V$SQLAREA как текущие модуль и действие.

```
PROCEDURE DBMS_APPLICATION_INFO.SET_SESSION_LONGOPS
   (rindex IN OUT BINARY_INTEGER,
    slno IN OUT BINARY_INTEGER,
    op_name IN VARCHAR2 DEFAULT NULL,
    target IN BINARY_INTEGER DEFAULT 0,
    context IN BINARY_INTEGER DEFAULT 0,
    sofar IN NUMBER DEFAULT 0,
    totalwork IN NUMBER DEFAULT 0,
    target_desc IN VARCHAR2 DEFAULT 'unknown target',
    units IN VARCHAR2 DEFAULT NULL);
```

Вставляет или обновляет данные времени выполнения для долго выполняющихся операций в виртуальной таблице V$SESSION_LONGOPS. Строки идентифицируются значением *rindex*. Чтобы была вставлена новая строка, значение *rindex* должно быть равно SET_SESSION_LONGOPS_NOHINT. Кроме того, к созданию новой строки может привести уникальная комбинация остальных параметров. Параметр *op_name* указывает имя долго выполняющейся операции; *target* – это имя объекта; *context*, *sofar* и *totalwork* – это параметры выполнения, определяемые клиентом в единицах *units*. Появилась в Oracle8*i*.

DBMS_APPLY_ADM

Предлагает процедуры, которые позволяют запускать, останавливать и изменять Apply-процессы. Apply-процессы используются механизмом Oracle Streams, появившимся в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_APPLY_ADM.ALTER_APPLY
    (apply_name IN VARCHAR2,
    rule_set_name IN VARCHAR2 DEFAULT NULL,
    remove_rule_set IN BOOLEAN DEFAULT FALSE,
    message_handler IN VARCHAR2 DEFAULT NULL,
    remove_message handler IN BOOLEAN DEFAULT FALSE,
    ddl_handler IN VARCHAR2 DEFAULT NULL,
    remove_ddl_handler IN BOOLEAN DEFAULT FALSE,
    apply_user IN VARCHAR2 DEFAULT NULL,
    apply_tag IN RAW DEFAULT NULL,
    remove_apply_tag IN BOOLEAN DEFAULT FALSE);
```

Изменяет характеристики Apply-процесса, изменяя или удаляя набор правил, обработчик сообщений, обработчик DDL или тег; или изменяя пользователя, применяющего все изменения DML и DDL и запускающего обработчики применения. Apply-процесс останавливается и запускается вновь при изменении любой характеристики, за исключением изменения пользователя.

```
PROCEDURE DBMS_APPLY_ADM.CREATE_APPLY
    (queue_name IN VARCHAR2,
    apply_name IN VARCHAR2,
    rule_set_name IN VARCHAR2 DEFAULT NULL,
    message_handler IN VARCHAR2 DEFAULT NULL,
    ddl_handle IN VARCHAR2 DEFAULT NULL,
    apply_user IN VARCHAR2 DEFAULT NULL,
    apply_database_link IN VARCHAR2 DEFAULT NULL,
    apply_tag IN RAW DEFAULT '00',
    apply_captured IN BOOLEAN DEFAULT FALSE);
```

Создает Apply-процесс.

```
PROCEDURE DBMS_APPLY_ADM.DELETE_ALL_ERRORS
   (apply_name IN VARCHAR2 DEFAULT NULL);
```

Удаляет все ошибочные транзакции для указанного Apply-процесса.

```
PROCEDURE DBMS_APPLY_ADM.DELETE_ERROR
   (local_transaction_id IN VARCHAR2);
```

Удаляет указанную ошибочную транзакцию.

```
PROCEDURE DBMS_APPLY_ADM.DROP_APPLY
   (apply_name IN VARCHAR2);
```

Удаляет указанный Apply-процесс.

```
PROCEDURE DBMS_APPLY_ADM.EXECUTE_ALL_ERRORS
    (apply_name IN VARCHAR2 DEFAULT NULL,
    execute_as_user IN BOOLEAN DEFAULT FALSE);
```

Применяется для повторного выполнения ошибочной транзакции в контексте безопасности текущего или исходного пользователя.

```
PROCEDURE DBMS_APPLY_ADM.EXECUTE_ERROR
    (local_transaction_name IN VARCHAR2,
    execute_as_user IN BOOLEAN DEFAULT FALSE);
```

Применяется для повторного выполнения указанной ошибочной транзакции.

```
PROCEDURE DBMS_APPLY_ADM.GET_ERROR_MESSAGE
    (message_number IN NUMBER,
    local_transaction_id IN VARCHAR2)
   RETURN Sys.Anydata;
```

Возвращает сообщение по указанному номеру и очереди транзакций.

```
PROCEDURE DBMS_APPLY_ADM.SET_DML_HANDLER
    (object_name IN VARCHAR2,
    object_type IN VARCHAR2,
    operation _name IN VARCHAR2,
    error_handler IN BOOLEAN DEFAULT FALSE,
    user_procedure IN VARCHAR2,
    apply_database_link IN VARCHAR2 DEFAULT NULL);
```

Устанавливает DML-обработчик для указанного объекта.

```
PROCEDURE DBMS_APPLY_ADM.SET_GLOBAL_INSTANTIATION_SCN
    (source_database_name IN VARCHAR2,
    instantiation_scn IN NUMBER,
    apply_database_link IN VARCHAR2 DEFAULT NULL);
```

Устанавливает системный номер изменения (System Change Number – SCN) при создании экземпляра указанной исходной БД.

```
PROCEDURE DBMS_APPLY_ADM.SET_KEY_COLUMNS
   (object_name IN VARCHAR2,
   {column_list IN VARCHAR2 | column_table IN DBMS_UTILITY.NAME_ARRAY},
   apply_database_link IN VARCHAR2 DEFAULT NULL);
```

Указывает, какие столбцы будут взяты для замещения столбцов первичного ключа объекта.

```
PROCEDURE DBMS_APPLY_ADM.SET_PARAMETER
    (apply_name IN VARCHAR2,
    parameter IN VARCHAR2,
    value IN VARCHAR2);
```

Указывает значение параметра Apply-процесса.

```
PROCEDURE DBMS_APPLY_ADM.SET_SCHEMA_INSTANTIATION_SCN
    (source_schema_name IN VARCHAR2,
    source_database_name IN VARCHAR2,
    instantiation_scn IN NUMBER,
    apply_database_link IN VARCHAR2 DEFAULT NULL);
```

Устанавливает системный номер изменения (SCN) для указанной схемы.

```
PROCEDURE DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN
    (source_object_name IN VARCHAR2,
    source_database_name IN VARCHAR2,
    instantiation_scn IN NUMBER,
    apply_database_link IN VARCHAR2 DEFAULT NULL);
```

Устанавливает системный номер изменения (SCN) для указанной таблицы.

```
PROCEDURE DBMS_APPLY_ADM.SET_UPDATE_CONFLICT_HANDLER
    (object_name IN VARCHAR2,
    method_name IN VARCHAR2,
    resolution_column IN VARCHAR2,
    column_list IN DBMS_UTILITY.NAME_ARRAY,
    apply_database_link IN VARCHAR2 DEFAULT NULL);
```

Управляет обработчиком конфликтов обновлений.

```
PROCEDURE DBMS_APPLY_ADM.START_APPLY
    (apply_name IN VARCHAR2);
```

Запускает Apply-процесс.

```
PROCEDURE DBMS_APPLY_ADM.STOP_APPLY
    (apply_name IN VARCHAR2,
    force IN BOOLEAN DEFAULT FALSE);
```

Останавливает Apply-процесс. Параметр *force* определяет, следует ли незамедлительно остановить Apply-процесс или же дождаться завершения применяемых транзакций.

DBMS_AQ

Содержит процедуры, выполняющие постановку сообщений в очереди и удаление их из очередей, создаваемых с применением компонента Advanced Queuing. Advanced Queuing предоставляет инфраструктуру обмена сообщениями Oracle, которая применяется для постановки заданий в очередь и обмена информацией.

Вызовы

```
PROCEDURE DBMS_AQ.ENQUEUE
    (queue_name IN VARCHAR2,
    enqueue_options IN DBMS_AQ.ENQUEUE_OPTIONS_T,
    message_properties IN DBMS_AQ.MESSAGE_PROPERTIES_T,
    payload IN <payload_type>,
    msgid OUT RAW);
```

Добавляет сообщение *payload* в очередь *queue_name*, применяя параметры, указанные в записи *enqueue_options*. Параметр *payload_type* – это тип *RAW* или имя объектного типа. Возвращает указатель на сообщение в *msgid*.

```
PROCEDURE DBMS_AQ.DEQUEUE
    (queue_name IN VARCHAR2,
    dequeue_options IN DBMS_AQ.DEQUEUE_OPTIONS_T,
    message_properties OUT DBMS_AQ.MESSAGE_PROPERTIES_T,
    payload OUT <payload_type>,
    msgid OUT RAW);
```

Значения параметров совпадают с описанными для DBMS_AQ.ENQUEUE.

```
DBMS_AQ.LISTEN
    (agent_list IN AQ$_AGENT_LIST_T,
    wait IN BINARY_INTEGER DEFAULT DBMS_AQ.FOREVER,
    agent OUT SYS.AQ$_AGENT);
```

Прослушивает одну или несколько очередей от имени агентов, указанных в списке. Появилась в Oracle8*i*.

```
DBMS_AQ.REGISTER
    (reg_list IN SYS.AQ$_REG_INFO_LIST,
    count IN NUMBER);
```

Регистрирует адрес электронной почты, процедуру PL/SQL или URL для уведомления о сообщениях. Появилась в Oracle9*i*.

```
DBMS_AQ.UNREGISTER
    (reg_list IN SYS.AQ$_REG_INFO_LIST,
    count IN NUMBER);
```

Отменяет регистрацию получателя уведомлений о сообщениях. Появилась в Oracle9*i*.

```
DBMS_AQ.POST
    (post_list IN SYS.AQ$_POST_INFO_LIST,
    count IN NUMBER);
```

Инициирует рассылку для списка анонимных подписок, отправляющую уведомления всем зарегистрированным клиентам. Появилась в Oracle9*i*.

```
DBMS_AQ.BIND_AGENT
    (agent IN SYS.AQ$_AGENT,
    certificate IN VARCHAR2 DEFAULT NULL);
```

Создает запись для агента AQ в каталоге LDAP. Появилась в Oracle9*i*.

```
DBMS_AQ.UNBIND_AGENT
    (agent IN SYS.AQ$_AGENT);
```

Удаляет запись для агента AQ из каталога LDAP. Появилась в Oracle9*i*.

DBMS_AQADM

Предлагает набор программ, которые можно использовать для создания и удаления очередей и таблиц очередей, а также управления ими при работе с компонентом Oracle Advanced Queuing.

Вызовы

```
PROCEDURE DBMS_AQADM.CREATE_QUEUE_TABLE
    (queue_table IN VARCHAR2,
    queue_payload_type IN VARCHAR2,
    storage_clause IN VARCHAR2 DEFAULT NULL,
    sort_list IN VARCHAR2 DEFAULT NULL,
    multiple_consumers IN BOOLEAN DEFAULT FALSE,
    message_grouping IN BINARY_INTEGER DEFAULT NONE,
    comment IN VARCHAR2 DEFAULT NULL,
    auto_commit IN BOOLEAN DEFAULT TRUE,
    primary_instance IN BINARY_INTEGER DEFAULT 0,
    secondary_instance IN BINARY_INTEGER DEFAULT 0,
    compatible IN VARCHAR2 DEFAULT NULL);
```

Создает таблицу очередей *queue_table* типа *queue_payload_type* (RAW или имя объектного типа). Параметры *primary_instance*, *secondary_instance* и *compatible* появились в Oracle8*i*.

```
PROCEDURE DBMS_AQADM.ALTER_QUEUE_TABLE
    (queue_table IN VARCHAR2,
    comment IN VARCHAR2 DEFAULT NULL,
    primary_instance IN BINARY_INTEGER DEFAULT NULL,
    secondary_instance IN BINARY_INTEGER DEFAULT NULL);
```

Изменяет свойства существующей очереди. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_AQADM.DROP_QUEUE_TABLE
    (queue_table IN VARCHAR2,
    force IN BOOLEAN DEFAULT FALSE,
    auto_commit IN BOOLEAN DEFAULT TRUE);
```

Удаляет таблицу очередей *queue_table*.

```
PROCEDURE DBMS_AQADM.CREATE_QUEUE
    (queue_name IN VARCHAR2,
    queue_table IN VARCHAR2,
    queue_type IN BINARY_INTEGER DEFAULT NORMAL_QUEUE,
    max_retries IN NUMBER DEFAULT NULL,
    retry_delay IN NUMBER DEFAULT 0,
    retention_time IN NUMBER DEFAULT 0,
    dependency_tracking IN BOOLEAN DEFAULT FALSE,
    comment IN VARCHAR2 DEFAULT NULL,
    auto_commit IN BOOLEAN DEFAULT TRUE);
```

Создает очередь *queue_name* в таблице очередей *queue_table*.

```
PROCEDURE DBMS_AQADM.CREATE_NP_QUEUE
    (queue_name IN VARCHAR2,
```

```
    multiple_consumers IN BOOLEAN DEFAULT FALSE,
    comment IN VARCHAR2 DEFAULT NULL);
```

Создает несохраняющуюся очередь. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_AQADM.ALTER_QUEUE
    (queue_name IN VARCHAR2,
    max_retries IN NUMBER DEFAULT NULL,
    retry_delay IN NUMBER DEFAULT NULL,
    retention_time IN NUMBER DEFAULT NULL,
    auto_commit IN BOOLEAN DEFAULT TRUE,
    comment IN VARCHAR2 DEFAULT NULL);
```

Изменяет указанные характеристики очереди *queue_name*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_AQADM.DROP_QUEUE
    (queue_name IN VARCHAR2,
    auto_commit IN BOOLEAN DEFAULT TRUE);
```

Удаляет очередь *queue_name*.

```
PROCEDURE DBMS_AQADM.START_QUEUE
    (queue_name IN VARCHAR2,
    enqueue IN BOOLEAN DEFAULT TRUE,
    dequeue IN BOOLEAN DEFAULT TRUE);
```

Запускает очередь *queue_name* с возможностью постановки в очередь и/или удаления из очереди.

```
PROCEDURE DBMS_AQADM.STOP_QUEUE
    (queue_name IN VARCHAR2,
    enqueue IN BOOLEAN DEFAULT TRUE,
    dequeue IN BOOLEAN DEFAULT TRUE,
    wait IN BOOLEAN DEFAULT TRUE);
```

Прекращает постановку в очередь и/или вывод из очереди для очереди *queue_name* с ожиданием завершения незафиксированных транзакций или незамедлительно.

```
PROCEDURE DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE
    (privilege IN VARCHAR2,
    grantee IN VARCHAR2,
    admin_option IN BOOLEAN := FALSE);
```

Предоставляет системные привилегии AQ пользователям и ролям. Изначально с процедурой могут работать только пользователи SYS и SYSTEM. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_AQADM.REVOKE_SYSTEM_PRIVILEGE
    (privilege IN VARCHAR2,
    grantee IN VARCHAR2);
```

Отзывает системные привилегии AQ у пользователей и ролей. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_AQADM.GRANT_QUEUE_PRIVILEGE
    (privilege IN VARCHAR2,
    queue_name IN VARCHAR2,
    grantee IN VARCHAR2,
    admin_option IN BOOLEAN := FALSE);
```

Предоставляет привилегии на очередь пользователям и ролям. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_AQADM.REVOKE_QUEUE_PRIVILEGE
    (privilege IN VARCHAR2,
```

```
    queue_name IN VARCHAR2,
    grantee IN VARCHAR2);
```

Отзывает привилегии на очередь у пользователей и ролей. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_AQADM.ADD_SUBSCRIBER
    (queue_name IN VARCHAR2,
    subscriber IN SYS.AQ$_AGENT,
    rule IN VARCHAR2 DEFAULT NULL
    [,transformation IN VARCHAR2 DEFAULT NULL]#);
```

Добавляет агента *subscriber* в очередь *queue_name*. Параметр *rule* появился в Oracle8*i*. Параметр *transformation* появился в Oracle9*i*.

```
PROCEDURE DBMS_AQADM.ALTER_SUBSCRIBER
    (queue_name IN VARCHAR2,
    subscriber IN SYS.AQ$_AGENT,
    rule IN VARCHAR2
    [,transformation IN VARCHAR2]#);
```

Изменяет свойства существующего подписчика. Процедура появилась в Oracle8*i*. Параметр *transformation* появился в Oracle9*i*.

```
PROCEDURE DBMS_AQADM.REMOVE_SUBSCRIBER
    (queue_name IN VARCHAR2,
    subscriber IN SYS.AQ$_AGENT);
```

Удаляет агента *subscriber* из очереди *queue_name*.

```
PROCEDURE DBMS_AQADM.SCHEDULE_PROPAGATION
    (queue_name IN VARCHAR2,
    destination IN VARCHAR2 DEFAULT NULL,
    start_time IN DATE DEFAULT SYSDATE,
    duration IN NUMBER DEFAULT NULL,
    next_time IN VARCHAR2 DEFAULT NULL,
    latency IN NUMBER DEFAULT 60);
```

Задает расписание тиражирования сообщений из очереди *queue_name* по направлению *destination*, свойства которого определяются остальными параметрами.

```
PROCEDURE DBMS_AQADM.UNSCHEDULE_PROPAGATION
    (queue_name IN VARCHAR2,
    destination IN VARCHAR2 DEFAULT NULL);
```

Отменяет ранее заданное расписание тиражирования сообщений из очереди *queue_name*.

```
PROCEDURE DBMS_AQADM.VERIFY_QUEUE_TYPES
    (src_queue_name IN VARCHAR2,
    dest_queue_name IN VARCHAR2,
    destination IN VARCHAR2 DEFAULT NULL,
    rc OUT BINARY_INTEGER);
```

Проверяет идентичность типов для исходной очереди и очереди назначения.

```
PROCEDURE DBMS_AQADM.ALTER_PROPAGATION_SCHEDULE
    (queue_name IN VARCHAR2,
    destination IN VARCHAR2 DEFAULT NULL,
    duration IN NUMBER DEFAULT NULL,
    next_time IN VARCHAR2 DEFAULT NULL,
    latency IN NUMBER DEFAULT 60);
```

Изменяет свойства существующего расписания тиражирования для очереди *queue_name*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_AQADM.ENABLE_PROPAGATION_SCHEDULE
    (queue_name IN VARCHAR2,
    destination IN VARCHAR2 DEFAULT NULL);
```

Вводит в действие расписание тиражирования, отмененное ранее. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_AQADM.DISABLE_PROPAGATION_SCHEDULE
    (queue_name IN VARCHAR2,
    destination IN VARCHAR2 DEFAULT NULL);
```

Отменяет расписание тиражирования. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_AQADM.MIGRATE_QUEUE_TABLE
    (queue_table IN VARCHAR2,
    compatible IN VARCHAR2);
```

Переход от очереди Oracle8 к очереди Oracle8*i* (*compatible* = '8.1') или наоборот (*compatible* = '8.0'). Появилась в Oracle8*i*.

```
PROCEDURE DBMS_AQADM.CREATE_AQ_AGENT
    (agent_name IN VARCHAR2,
    certificate_location IN VARCHAR2 DEFAULT NULL,
    enable_http IN BOOLEAN DEFAULT FALSE,
    enable_smtp IN BOOLEAN DEFAULT FALSE,
    enable_anyp IN BOOLEAN DEFAULT FALSE);
```

Регистрирует агента с именем *agent_name* для доступа к очереди по протоколам HTTP, SMTP или обоим. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_AQADM.ALTER_AQ_AGENT
    (agent_name IN VARCHAR2,
    certificate_location IN VARCHAR2 DEFAULT NULL,
    enable_http IN BOOLEAN DEFAULT FALSE,
    enable_smtp IN BOOLEAN DEFAULT FALSE,
    enable_anyp IN BOOLEAN DEFAULT FALSE);
```

Изменяет характеристики существующего агента AQ. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_AQADM.DROP_AQ_AGENT
    (agent_name IN VARCHAR2);
```

Удаляет существующего агента AQ. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_AQADM.ENABLE_DB_ACCESS
    (agent_name IN VARCHAR2,
    db_username IN VARCHAR2);
```

Предоставляет агенту AQ привилегии указанного пользователя БД. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_AQADM.DISABLE_DB_ACCESS
    (agent_name IN VARCHAR2,
    db_username IN VARCHAR2);
```

Отзывает привилегии указанного пользователя БД у агента AQ. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_AQADM.ADD_ALIAS_TO_LDAP
    (alias IN VARCHAR2,
    obj_location IN VARCHAR2);
```

Создает в каталоге LDAP псевдоним *alias* для очереди, агента или JMSConnectionFactory. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_AQADM.DEL_ALIAS_TO_LDAP
   (alias IN VARCHAR2);
```

Удаляет существующий псевдоним из каталога LDAP. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_AQADM.GRANT_TYPE_ACCESS
   (user_name IN VARCHAR2);
```

Предоставляет пользователю *user_name* возможность создавать очереди, работающие с несколькими потребителями. Эта процедура не поддерживается начиная с Oracle8*i*.

```
FUNCTION DBMS_AQADM.QUEUE_SUBSCRIBERS
   (queue_name IN VARCHAR2)
   RETURN AQ$_SUBSCRIBER_LIST_T;
```

Возвращает список подписчиков очереди *queue_name*. Эта процедура не поддерживается начиная с Oracle8*i*.

```
PROCEDURE DBMS_AQADM.START_TIME_MANAGER;
```

Запускает процесс Queue Monitor. Эта процедура не поддерживается начиная с Oracle8.

```
PROCEDURE DBMS_AQADM.STOP_TIME_MANAGER;
```

Останавливает процесс Queue Monitor. Эта процедура не поддерживается начиная с Oracle8.

DBMS_AQELM

Содержит процедуры для управления асинхронным уведомлением в Advanced Queuing. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_AQELM.SET_MAILHOST
   (mailhost IN VARCHAR2);
```

Устанавливает имя хоста SMTP-сервера.

```
PROCEDURE DBMS_AQELM.GET_MAILHOST
   (mailhost OUT VARCHAR2);
```

Получает имя хоста SMTP-сервера.

```
PROCEDURE DBMS_AQELM.SET_MAILPORT
   (mailport IN NUMBER);
```

Указывает порт SMTP-сервера.

```
PROCEDURE DBMS_AQELM.GET_MAILPORT
   (mailport OUT NUMBER);
```

Получает номер порта SMTP-сервера.

```
PROCEDURE DBMS_AQELM.SET_SENDFROM
   (sendfrom IN VARCHAR2);
```

Устанавливает адрес, с которого осуществляется отправка.

```
PROCEDURE DBMS_AQELM.GET_SENDFROM
   (sendfrom OUT VARCHAR2);
```

Получает адрес, с которого осуществляется отправка.

```
PROCEDURE DBMS_AQELM.SET_PROXY
    (proxy IN VARCHAR2,
    no_proxy_domains IN VARCHAR2 DEFAULT NULL);
```

Задает имя прокси-сервера для HTTP-запросов, за исключением запросов от хостов, перечисленных в параметре *no_proxy_domains*.

```
PROCEDURE DBMS_AQELM.GET_PROXY
    (proxy OUT VARCHAR2,
    no_proxy_domains OUT VARCHAR2 DEFAULT NULL);
```

Возвращает имя прокси-сервера для HTTP-запросов и список хостов, работающих в обход прокси-сервера, в параметре *no_proxy_domains*.

DBMS_CAPTURE_ADM

Включает в себя процедуры для управления процессами захвата при работе с механизмом Oracle Streams. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_CAPTURE_ADM.ABORT_GLOBAL_INSTANTIATION;
```

Удаляет информацию словаря данных о тиражировании БД. Эта процедура противоположна по отношению к PREPARE_GLOBAL_INSTANTIATION.

```
PROCEDURE DBMS_CAPTURE_ADM.ABORT_SCHEMA_INSTANTIATION
    (schema_name IN VARCHAR2);
```

Удаляет информацию словаря данных о тиражировании схемы. Эта процедура противоположна по отношению к PREPARE_SCHEMA_INSTANTIATION.

```
PROCEDURE DBMS_CAPTURE_ADM.ABORT_TABLE_INSTANTIATION
   (table_name IN VARCHAR2);
```

Удаляет информацию словаря данных о тиражировании таблицы. Эта процедура противоположна по отношению к PREPARE_TABLE_INSTANTIATION.

```
PROCEDURE DBMS_CAPTURE_ADM.ALTER_CAPTURE
    (capture_name IN VARCHAR2,
    rule_set_name IN VARCHAR2 DEFAULT NULL,
    remove_rule_set IN BOOLEAN DEFAULT FALSE,
    start_scn IN NUMBER DEFAULT NULL);
```

Изменяет характеристики существующего процесса захвата.

```
PROCEDURE DBMS_CAPTURE_ADM.CREATE_CAPTURE
    (queue_name IN VARCHAR2,
    capture_name IN VARCHAR2,
    rule_set_name IN VARCHAR2 DEFAULT NULL,
    start_scn IN NUMBER DEFAULT NULL);
```

Создает процесс захвата.

```
PROCEDURE DBMS_CAPTURE_ADM.DROP_CAPTURE
    (capture_name IN VARCHAR2);
```

Удаляет существующий процесс захвата.

```
PROCEDURE DBMS_CAPTURE_ADM.PREPARE_GLOBAL_INSTANTIATION;
```

Выполняет синхронизацию, необходимую для тиражирования всех таблиц одной БД в другую БД.

```
PROCEDURE DBMS_CAPTURE_ADM.PREPARE_SCHEMA_INSTANTIATION
    (schema_name IN VARCHAR2);
```

Выполняет синхронизацию, необходимую для тиражирования всех таблиц схемы в другую БД.

```
PROCEDURE DBMS_CAPTURE_ADM.PREPARE_TABLE_INSTANTIATION
    (table_name IN VARCHAR2);
```

Выполняет синхронизацию, необходимую для тиражирования таблицы в другую БД.

```
PROCEDURE DBMS_CAPTURE_ADM.SET_PARAMETER
    (capture_name IN VARCHAR2,
    parameter IN VARCHAR2,
    value IN VARCHAR2);
```

Задает значение параметра процесса захвата.

```
PROCEDURE DBMS_CAPTURE_ADM.START_CAPTURE
    (capture_name IN VARCHAR2);
```

Запускает процесс захвата.

```
PROCEDURE DBMS_CAPTURE_ADM.STOP_CAPTURE
    (capture_name IN VARCHAR2,
    force IN BOOLEAN DEFAULT FALSE);
```

Останавливает процесс захвата. Если задан параметр *force*, то процесс захвата останавливается незамедлительно; в противном случае он останавливается после захвата текущей транзакции.

DBMS_DDL

Содержит процедуры для повторной компиляции хранимого кода, анализа и вычисления статистики для объектов БД, а также для реорганизации ссылок на идентификаторы объектов в Oracle.

Вызовы

```
PROCEDURE DBMS_DDL.ALTER_COMPILE
    (type IN VARCHAR2,
    schema IN VARCHAR2,
    name IN VARCHAR2);
```

Повторно компилирует хранимый объект PL/SQL *name* (имя чувствительно к регистру), принадлежащий схеме *schema* и имеющий тип *type*. Значение NULL в параметре *schema* соответствует текущей схеме. Разрешенные значения для *type*: PROCEDURE, FUNCTION, PACKAGE, PACKAGE BODY и PACKAGE SPECIFICATION.

```
PROCEDURE DBMS_DDL.ALTER_TABLE_NOT_REFERENCEABLE
    (table_name IN VARCHAR2,
    table_schema IN VARCHAR2 DEFAULT NULL,
    affected_schema IN VARCHAR2 DEFAULT NULL);
```

Освобождает ссылки из схемы *affected_schema* на объекты таблицы *table_name*, принадлежащей схеме *table_schema*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_DDL.ALTER_TABLE_REFERENCEABLE
    (table_name IN VARCHAR2,
```

```
    table_schema IN VARCHAR2 DEFAULT NULL,
    affected_schema IN VARCHAR2 DEFAULT NULL);
```

Перенаправляет ссылки из схемы *affected_schema* на объекты таблицы *table_name*, принадлежащей схеме *table_schema*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_DDL.ANALYZE_OBJECT
    (type IN VARCHAR2,
    schema IN VARCHAR2,
    name IN VARCHAR2,
    method IN VARCHAR2,
    estimate_rows IN NUMBER DEFAULT NULL,
    estimate_percent IN NUMBER DEFAULT NULL,
    method_opt IN VARCHAR2 DEFAULT NULL,
    partname VARCHAR2 DEFAULT NULL);
```

Анализирует объект БД с именем *name*, принадлежащий схеме *schema* и имеющий тип *type* (TABLE, INDEX или CLUSTER), с помощью метода *method* (ESTIMATE, NULL или DELETE). Если применяется метод ESTIMATE, то необходимо указать *estimate_rows* или *estimate_percent* для определения размера выборки. В параметре *method_opt* можно указать дополнительное условие анализа: FOR TABLE, FOR ALL COLUMNS [SIZE N], FOR ALL INDEXED COLUMNS [SIZE N] и FOR ALL INDEXES. Параметр *partname* указывает имя раздела.

```
PROCEDURE DBMS_DDL.SET_TRIGGER_FIRING_PROPERTY
    (trig_owner IN VARCHAR2,
    trig_name IN VARCHAR2,
    fire_once IN BOOLEAN);
```

Устанавливает свойство запуска триггера *trig_name* в значение *fire_once*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_DDL.IS_TRIGGER_FIRE_ONCE
    (trig_owner IN VARCHAR2,
    trig_name IN VARCHAR2);
```

Возвращает TRUE, если для триггера *trig_name* установлено однократное срабатывание, и FALSE – в противном случае. Появилась в Oracle9*i*.

DBMS_DEBUG

Предоставляет API отладочного уровня PL/SQL. Для работы с таким API необходимы два активных сеанса: один для кода PL/SQL, второй для отладчика. Кроме того, необходимо включить отладочный режим сеанса при помощи следующей команды:

```
ALTER SESSION SET PLSQL_DEBUG = TRUE
```

и заново скомпилировать отлаживаемый код при помощи команд:

```
ALTER unit_type unit_name COMPILE DEBUG;
ALTER PACKAGE | TYPE unit_name COMPILE DEBUG BODY;
```

Процедуры PROBE_VERSION, SELF_CHECK и SET_TIMEOUT можно вызывать из любого сеанса. Процедуры INITIALIZE, DEBUG_ON и DEBUG_OFF необходимо вызывать из отлаживаемого сеанса. Все остальные вызовы можно выполнить из сеанса отладчика.

Этот пакет появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_DEBUG.PROBE_VERSION
    (major OUT BINARY_INTEGER,
    minor OUT BINARY_INTEGER))
```

Возвращает основной и дополнительный номера версии пакета.

```
PROCEDURE DBMS_DEBUG.SELF_CHECK
    (timeout IN  BINARY_INTEGER DEFAULT 60);
```

Если этот вызов не завершается успешно через *timeout* секунд, то это означает, что на сервере установлена неподходящая версия DBMS_DEBUG.

```
FUNCTION DBMS_DEBUG.SET_TIMEOUT
    (timeout IN BINARY_INTEGER)
    RETURN BINARY_INTEGER;
```

Устанавливает и возвращает значение таймаута (*timeout*) в секундах.

```
FUNCTION DBMS_DEBUG.INITIALIZE
    (debug_session_id IN VARCHAR2 := NULL,
    diagnostics IN BINARY_INTEGER := 0)
    RETURN VARCHAR2;
```

Инициализирует отладочный сеанс *debug_session_id*. Если значение не указано, то идентификатор генерируется автоматически. Параметр *diagnostics* может иметь значение 0 (отсутствие диагностики в файле трассировки) или 1 (включение диагностики в файл трассировки). Процедура возвращает идентификатор сеанса.

```
PROCEDURE DBMS_DEBUG.DEBUG_ON
    (no_client_side_plsql_engine BOOLEAN := TRUE,
    immediate BOOLEAN := FALSE);
```

Включает отладку на сервере, если значение первого параметра не равно FALSE. Если параметр *immediate* не равен TRUE, то сеанс включает отладку по завершении вызова; в противном случае отладка запускается незамедлительно.

```
PROCEDURE DBMS_DEBUG.DEBUG_OFF;
```

Выключает отладку.

```
PROCEDURE DBMS_DEBUG.ATTACH_SESSION
    (debug_session_id IN VARCHAR2,
    diagnostics IN BINARY_INTEGER := 0);
```

Информирует сеанс отладки об отлаживаемой программе. Если значение *diagnostics* отлично от значения по умолчанию (0), то генерируется вывод диагностики.

```
FUNCTION DBMS_DEBUG.SYNCHRONIZE
    (run_info OUT runtime_info,
    info_requested IN BINARY_INTEGER := NULL)
    RETURN BINARY_INTEGER;
```

Функция ждет, пока отлаживаемая программа не сигнализирует о событии. Если параметр *info_requested* не содержит NULL, процедура после получения сигнала о событии вызывает DBMS_DEBUG.GET_RUNTIME_INFO. Функция возвращает код, означающий успех, истечение времени ожидания или другую ошибку.

```
PROCEDURE DBMS_DEBUG.SHOW_SOURCE
    (first_line IN BINARY_INTEGER,
    last_line IN BINARY_INTEGER,
    source OUT vc2_table);
```

```
PROCEDURE DBMS_DEBUG.SHOW_SOURCE
    (first_line IN BINARY_INTEGER,
    last_line IN BINARY_INTEGER,
    window IN BINARY_INTEGER,
    print_arrow IN BINARY_INTEGER,
    buffer IN OUT VARCHAR2,
    buflen IN BINARY_INTEGER,
    pieces OUT BINARY_INTEGER);
```

Возвращает исходный текст несохраняющейся программы. Первый вариант возвращает весь код в таблицу; второй возвращает отформатированный исходный текст, но может вернуть текст не целиком.

```
PROCEDURE DBMS_DEBUG.PRINT_BACKTRACE
    (listing IN OUT VARCHAR2 | backtrace_table);
```

Выводит содержимое текущего стека вызовов.

```
FUNCTION DBMS_DEBUG.CONTINUE
    (run_info IN OUT runtime_info,
    breakflags IN BINARY_INTEGER,
    info_requested IN BINARY_INTEGER := NULL)
    RETURN BINARY_INTEGER;
```

Возвращает статус программы в *run_info*. Параметр *breakflags* содержит маску для ожидаемых событий. Параметр *info_requested* показывает, какая информация должна быть возвращена при остановке программы. Процедура возвращает код завершения, подобно процедуре DBMS_DEBUG. SYNCHRONIZE.

```
FUNCTION DBMS_DEBUG.SET_BREAKPOINT
    (program IN program_info,
    line# IN BINARY_INTEGER,
    breakpoint# OUT BINARY_INTEGER,
    fuzzy IN BINARY_INTEGER := 0,
    iterations IN BINARY_INTEGER := 0)
    RETURN BINARY_INTEGER;
```

Устанавливает точку останова в отлаживаемом коде. Параметр *fuzzy* применяется, если в указанной строке нет исполняемого кода: 1 означает поиск строки в прямом направлении, а –1 – в обратном направлении. В версии Oracle9*i* Release 2 *fuzzy* и *iterations* еще не были реализованы, но они должны появиться в последующих редакциях. Процедура возвращает код завершения, подобно процедуре DBMS_DEBUG. SYNCHRONIZE.

```
FUNCTION DBMS_DEBUG.DELETE_BREAKPOINT
    (breakpoint IN BINARY_INTEGER)
    RETURN BINARY_INTEGER;
```

Удаляет существующую точку останова. Процедура возвращает код завершения, подобно процедуре DBMS_DEBUG. SYNCHRONIZE.

```
FUNCTION DBMS_DEBUG.DISABLE_BREAKPOINT
    (breakpoint IN BINARY_INTEGER)
    RETURN BINARY_INTEGER;
```

Отключает точку останова. Процедура возвращает код завершения, подобно процедуре DBMS_DEBUG. SYNCHRONIZE.

```
FUNCTION DBMS_DEBUG.ENABLE_BREAKPOINT
    (breakpoint IN BINARY_INTEGER)
    RETURN BINARY_INTEGER;
```

Активирует точку останова. Процедура возвращает код завершения, подобно процедуре DBMS_DEBUG. SYNCHRONIZE.

```
PROCEDURE DBMS_DEBUG.SHOW_BREAKPOINTS
    (listing {IN OUT VARCHAR2|OUT breakpoint_table});
```

Возвращает список существующих точек останова.

```
FUNCTION DBMS_DEBUG.GET_VALUE
    (variable_name IN VARCHAR2,
    {frame# IN BINARY_INTEGER | handle IN program_info},
    scalar_value OUT VARCHAR2,
    format IN VARCHAR2 := NULL)
    RETURN BINARY_INTEGER;
```

Возвращает значение переменной пакета. Процедура возвращает код завершения, подобно процедуре DBMS_DEBUG. SYNCHRONIZE.

```
FUNCTION DBMS_DEBUG.SET_VALUE
    ({frame# IN BINARY_INTEGER | handle IN program_info},
    assignment_statement IN VARCHAR2)
    RETURN BINARY_INTEGER;
```

Устанавливает значение переменной пакета. Процедура возвращает код завершения, подобно процедуре DBMS_DEBUG. SYNCHRONIZE.

```
PROCEDURE DBMS_DEBUG.DETACH_SESSION;
```

Останавливает отладку программы.

```
FUNCTION DBMS_DEBUG.GET_RUNTIME_INFO
    (info_requested IN BINARY_INTEGER,
    run_info OUT runtime_info)
    RETURN BINARY_INTEGER;
```

Возвращает информацию о текущей программе. Начиная с Oracle8*i* применяется только для PL/SQL на стороне клиента.

```
FUNCTION DBMS_DEBUG.GET_INDEXES
    (varname IN VARCHAR2,
    frame# IN BINARY_INTEGER,
    handle IN program_info,
    entries OUT index_table)
    RETURN BINARY_INTEGER;
```

Возвращает множество индексов *varname*. Возвращает ошибку, если *varname* – это неиндексированная таблица.

```
PROCEDURE DBMS_DEBUG.EXECUTE
    (what IN VARCHAR2,
    frame# IN BINARY_INTEGER,
    bind_results IN BINARY_INTEGER,
    results IN OUT NOCOPY dbms_debug_vc2coll,
    errm IN OUT NOCOPY VARCHAR2);
```

Выполняет код, содержащийся в строке *what* в рамках сеанса отлаживаемой программы. Параметр *bind_results* определяет, следует ли связывать исходный текст с результатами для того, чтобы вернуть значение (значение 1) или же нет (значение 0).

```
PROCEDURE DBMS_DEBUG.PRINT_INSTANTIATIONS
    (pkgs IN OUT NOCOPY backtrace_table,
    flags IN BINARY_NUMBER);
```

Возвращает список пакетов, экземпляры которых были созданы в сеансе отлаживаемой программы. Параметр *flags* указывает область действия отчета. Появилась в Oracle9*i*.

```
FUNCTION DBMS_DEBUG.TARGET_PROGRAM_RUNNING RETURN BOOLEAN;
```

Возвращает TRUE, если в отлаживаемом сеансе в данный момент выполняется процедура. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_DEBUG.PING;
```

Посылает запросы проверки доступности отлаживаемому сеансу для предотвращения прерывания по истечении таймаута. Параметры таймаута для сеанса устанавливаются при помощи DBMS_DEBUG.SET_TIMEOUT_BEHAVIOR. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_DEBUG.SET_TIMEOUT_BEHAVIOR
    (behavior IN PLS_INTEGER);
```

Задает действие для прерывания по истечении таймаута в отлаживаемом сеансе. Появилась в Oracle9*i*.

```
FUNCTION DBMS_DEBUG.GET_TIMEOUT_BEHAVIOR
   RETURN BINARY_INTEGER;
```

Возвращает действие, выполняемое при наступлении таймаута. Появилась в Oracle9*i*.

```
FUNCTION DBMS_DEBUG.SET_OER_BREAKPOINT
    (oer IN PLS_INTEGER)
    RETURN PLS_INTEGER;
```

Устанавливает точку останова OER. (OER – это ошибка, возвращаемая из ядра Oracle). Подробно точки останова OER описаны в документации Oracle. Появилась в Oracle9*i*.

```
FUNCTION DBMS_DEBUG.DELETE_OER_BREAKPOINT
    (oer IN PLS_INTEGER)
    RETURN PLS_INTEGER;
```

Удаляет точку останова OER. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_DEBUG.SHOW_BREAKPOINTS
    (code_breakpoints OUT breakpoint_table,
    oer_breakpoints OUT oer_table);
```

Возвращает существующие точки останова.

DBMS_DEFER

Содержит процедуры, реализующие взаимодействие с механизмом отложенных вызовов тиражируемых транзакционных удаленных процедур (*replicated transactional deferred remote procedure call facility*). Появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_DEFER.CALL
    (schema_name IN VARCHAR2,
    package_name IN VARCHAR2,
    proc_name IN VARCHAR2,
    arc_count IN NATURAL,
    {node IN node_list_t | group_name IN VARCHAR DEFAULT ''});
```

Создает отложенный вызов удаленной процедуры. Последний параметр появился в Oracle8*i*.

```
PROCEDURE DBMS_DEFER.COMMIT_WORK
   (commit_work_comment IN VARCHAR2);
```

Выполняет фиксацию транзакции после проверки на корректно сформированные вызовы отложенных удаленных процедур.

```
PROCEDURE DBMS_DEFER.datatype_ARG
   (arg IN datatype);
```

Указывает тип данных параметров. Применяется после процедуры DBMS_DEFER.CALL.

```
PROCEDURE DBMS_DEFER.TRANSACTION
   (nodes IN node_list_t);
```

Определяет начало отложенной транзакции, тиражируемой в базах данных, список полных имен которых содержится в параметре *nodes*.

DBMS_DEFER_QUERY

Позволяет обращаться к данным очереди отложенных транзакций, которые не видны в обычных представлениях словаря данных.

Вызовы

```
FUNCTION DBMS_DEFER_QUERY.GET_ARG_FORM
    (callno IN NUMBER,
    arg_no IN NUMBER,
    deferred_tran_id IN VARCHAR2)
    RETURN NUMBER;
```

Возвращает константу, указывающую на вид набора символов для параметра отложенного вызова.

```
FUNCTION DBMS_DEFER_QUERY.GET_ARG_TYPE
    (callno IN NUMBER,
    arg_no IN NUMBER,
    deferred_tran_id IN VARCHAR2)
    RETURN NUMBER;
```

Определяет тип аргумента в отложенном вызове.

```
PROCEDURE DBMS_DEFER_QUERY.GET_CALL_ARGS
    (callno IN NUMBER,
    startarg IN NUMBER := 1,
    argcnt IN NUMBER,
    argsize IN NUMBER,
    tran_id IN VARCHAR2,
    date_fmt IN VARCHAR2,
    types OUT TYPE_ANY,
    forms OUT TYPE_ANY,
    vals OUT VAL_ANY);
```

Возвращает текстовую версию аргументов для *callno*.

```
FUNCTION DBMS_DEFER_QUERY.GET_datatype_ARG
    (callno IN NUMBER,
    arg_no IN NUMBER,
```

```
    deferred_tran_id IN VARCHAR2)
   RETURN datatype;
```

Возвращает значение аргумента для *callno*.

```
FUNCTION DBMS_DEFER_QUERY.GET_OBJECT_NULL_VECTOR_ARG
    (callno IN NUMBER,
    arg_no IN NUMBER,
    deferred_tran_id IN VARCHAR2)
    RETURN SYSTEM.REPCAT$_OBJECT_NULL_VECTOR;
```

Возвращает информацию о типе объекта столбца. Появилась в Oracle9*i*.

DBMS_DEFER_SYS

Содержит процедуры, позволяющие управлять узлами тиражирования, назначенными по умолчанию.

Вызовы

```
PROCEDURE DBMS_DEFER_SYS.ADD_DEFAULT_DEST
   (dblink IN VARCHAR2);
```

Добавляет полное имя БД (*dblink*) в представление словаря данных DEFDEFAULTDEST.

```
PROCEDURE DBMS_DEFER_SYS.CLEAR_PROP_STATISTICS
   (dblink IN VARCHAR2);
```

Очищает статистику тиражирования для БД *dblink* из представления словаря данных DEFSCHEDULE. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_DEFER_SYS.DELETE_DEFAULT_DEST
   (dblink IN VARCHAR2);
```

Удаляет БД *dblink* из представления словаря данных DEFDEFAULTDEST.

```
PROCEDURE DBMS_DEFER_SYS.DELETE_DEF_DESTINATION
    (destination IN VARCHAR2,
    force IN BOOLEAN := FALSE);
```

Удаляет БД *destination* из представления словаря данных DEFSCHEDULE. Если значение параметра *force* равно TRUE, то сервер Oracle игнорирует любые проверки на безопасность и удаляет БД.

```
PROCEDURE DBMS_DEFER_SYS.DELETE_ERROR
    (deferred_tran_id IN VARCHAR2,
    destination IN VARCHAR2);
```

Удаляет отложенную транзакцию *deferred_tran_id* в БД *destination* из представления словаря данных DEFERROR.

```
PROCEDURE DBMS_DEFER_SYS.DELETE_TRAN
    (deferred_tran_id IN VARCHAR2,
    destination IN VARCHAR2);
```

Удаляет отложенную транзакцию *deferred_tran_id* в БД *destination* из представления словаря данных DEFTRANDEST.

```
FUNCTION DBMS_DEFER_SYS.DISABLED
    (destination IN VARCHAR2)
    RETURN BOOLEAN;
```

Возвращает TRUE, если запрещено тиражирование из текущей БД в БД *destination*.

```
FUNCTION DBMS_DEFER_SYS.EXCLUDE_PUSH
    (timeout IN INTEGER)
    RETURN INTEGER;
```

Не допускает распространения (PUSH) отложенной транзакции на период выполнения транзакции. Если не удается получить блокировку на PUSH в течение *timeout* секунд, то возвращает 1.

```
PROCEDURE DBMS_DEFER_SYS.EXECUTE_ERROR
    (deferred_tran_id IN VARCHAR2,
    destination IN VARCHAR2);
```

Повторно выполняет отложенную транзакцию *deferred_tran_id* в БД *destination*, если проблемы были вызваны ошибкой в исходном контексте безопасности.

```
PROCEDURE DBMS_DEFER_SYS.EXECUTE_ERROR_AS_USER
    (deferred_tran_id IN VARCHAR2,
    destination IN VARCHAR2);
```

Повторно выполняет отложенную транзакцию *deferred_tran_id* в БД *destination*, если проблемы были вызваны ошибкой в контексте безопасности текущего пользователя.

```
FUNCTION DBMS_DEFER_SYS.PURGE
    (purge_method IN BINARY_INTEGER := purge_method,
    rollback_segment IN VARCHAR2 := NULL,
    startup_seconds IN BINARY_INTEGER := 0,
    execution_seconds IN BINARY_INTEGER := NULL | SECONDS_INFINITY,
    delay_seconds IN BINARY_INTEGER := 0,
    transaction_count IN BINARY_INTEGER := NULL | TRANSACTIONS_INFINITY,
    write_trace IN BOOLEAN := NULL)
    RETURN BINARY_INTEGER;
```

Удаляет продвинутые транзакции из очереди отложенных транзакций текущего главного узла тиражирования. Значение по умолчанию SECONDS_INFINITY и TRANSACTIONS_INFINITY были изменены в Oracle9*i*.

```
FUNCTION DBMS_DEFER_SYS.PUSH
    (destination IN VARCHAR2,
    parallelism IN BINARY_INTEGER := 0,
    heap_size IN BINARY_INTEGER := 0,
    stop_on_error IN BOOLEAN := FALSE,
    write_trace IN BOOLEAN := FALSE,
    startup_seconds IN BINARY_INTEGER := 0,
    execution_seconds IN BINARY_INTEGER := SECONDS_INFINITY,
    delay_seconds IN BINARY_INTEGER := 0,
    transaction_count IN BINARY_INTEGER := TRANSACTIONS_INFINITY,
    deliver_order_limit IN BOOLEAN := NULL)
    RETURN BINARY_INTEGER;
```

Инициирует распространение очереди отложенных удаленных вызовов процедур (RPC) в БД *destination*.

```
PROCEDURE DBMS_DEFER_SYS.REGISTER_PROPAGATOR
    (username IN VARCHAR2);
```

Регистрирует пользователя *username* в качестве *распространителя* (*propagator*) для локальной БД и выдает необходимые привилегии.

```
PROCEDURE DBMS_DEFER_SYS.SCHEDULE_PURGE
    (interval IN VARCHAR2,
    next_date IN DATE,
```

```
    reset IN BOOLEAN DEFAULT NULL,
    purge_method IN BINARY_INTEGER DEFAULT NULL,
    rollback_segment IN VARCHAR2 DEFAULT NULL,
    startup_seconds IN BINARY_INTEGER DEFAULT 0,
    execution_seconds IN BINARY_INTEGER := NULL | SECONDS_INFINITY,
    delay_seconds IN BINARY_INTEGER := 0,
    transaction_count IN BINARY_INTEGER := NULL | TRANSACTIONS_INFINITY,
    write_trace IN BOOLEAN := NULL);
```

Устанавливает расписание выполнения задания для очистки отложенных транзакций. Значения по умолчанию SECONDS_INFINITY и TRANSACTIONS_INFINITY были изменены в Oracle9*i*.

```
PROCEDURE DBMS_DEFER_SYS.SCHEDULE_PUSH
    (destination IN VARCHAR2,
    interval IN VARCHAR2,
    next_date IN DATE,
    reset IN BOOLEAN := FALSE,
    parallelism IN BINARY_INTEGER := NULL,
    heap_size IN BINARY_INTEGER := NULL,
    stop_on_error IN BOOLEAN := NULL,
    write_trace IN BOOLEAN := NULL,
    startup_seconds IN BINARY_INTEGER := 0,
    execution_seconds IN BINARY_INTEGER := NULL | SECONDS_INFINITY,
    delay_seconds IN BINARY_INTEGER := 0,
    transaction_count IN BINARY_INTEGER := NULL | TRANSACTIONS_INFINITY
    [,write_trace IN BOOLEAN := NULL]#);
```

Устанавливает расписание выполнения задания по продвижению очереди отложенных транзакций в БД *destination*. Значения по умолчанию SECONDS_INFINITY и TRANSACTIONS_INFINITY были изменены в Oracle9*i*. Параметр *write_trace* появился в Oracle9*i*.

```
PROCEDURE DBMS_DEFER_SYS.SET_DISABLED
    (destination IN VARCHAR2,
    disabled IN BOOLEAN := TRUE
    [,catchup IN RAW := '00']#
    [,override IN BOOLEAN := FALSE]#);
```

Включает или выключает тиражирование очереди отложенных транзакций в БД *destination*. В Oracle9*i* появились параметры: *catchup*, представляющий собою идентификатор расширения для добавления новых главных узлов тиражирования в главную группу без приостановки (quiescing) главной группы, и *override*, который может заставить сервер Oracle игнорировать состояние отключения, установленное внутренними средствами для синхронизации.

```
PROCEDURE DBMS_DEFER_SYS.UNREGISTER_PROPAGATOR
    (username IN VARCHAR2,
    timeout IN INTEGER DEFAULT DBMS_LOCK.MAXWAIT);
```

Удаляет пользователя *username* из локальной БД и отзывает выданные ему привилегии.

```
PROCEDURE DBMS_DEFER_SYS.UNSCHEDULE_PURGE();
```

Останавливает задание на автоматическую очистку продвинутых транзакций.

```
PROCEDURE DBMS_DEFER_SYS.UNSCHEDULE_PUSH
    (destination IN VARCHAR2);
```

Останавливает задание на автоматическое продвижение для БД *destination*.

DBMS_DESCRIBE

Содержит единственную процедуру, предназначенную для описания аргументов хранимой процедуры или функции PL/SQL.

Вызов

```
PROCEDURE DBMS_DESCRIBE.DESCRIBE_PROCEDURE
    (object_name IN VARCHAR2,
    reserved1 IN VARCHAR2,
    reserved2 IN VARCHAR2,
    overload OUT DBMS_DESCRIBE.NUMBER_TABLE,
    position OUT DBMS_DESCRIBE.NUMBER_TABLE,
    level OUT DBMS_DESCRIBE.NUMBER_TABLE,
    argument_name OUT DBMS_DESCRIBE.VARCHAR2_TABLE,
    datatype OUT DBMS_DESCRIBE.NUMBER_TABLE,
    default_value OUT DBMS_DESCRIBE.NUMBER_TABLE,
    in_out OUT DBMS_DESCRIBE.NUMBER_TABLE,
    length OUT DBMS_DESCRIBE.NUMBER_TABLE,
    precision OUT DBMS_DESCRIBE.NUMBER_TABLE,
    scale OUT DBMS_DESCRIBE.NUMBER_TABLE,
    radix OUT DBMS_DESCRIBE.NUMBER_TABLE,
    spare OUT DBMS_DESCRIBE.NUMBER_TABLE);
```

Возвращает информацию о параметрах и тип возвращаемого значения (для функции) для указанного объекта (процедуры или функции) в таблицы PL/SQL, типы которых описаны в этом же пакете.

DBMS_DISTRIBUTED_TRUST_ADMIN

Процедуры, применяемые для поддержки Trusted Servers. Программные единицы данного пакета работают со списком Trusted Servers. Если база данных не является доверенной, то сервер Oracle отвергает связи данной БД.

Вызовы

```
DBMS_DISTRIBUTED_TRUST_ADMIN.ALLOW_ALL;
```

Очищает список доверенных серверов и указывает, что все серверы входят в доверенный домен.

```
DBMS_DISTRIBUTED_TRUST_ADMIN.ALLOW_SERVER
    (server IN VARCHAR2);
```

Разрешает доступ к указанному серверу, даже если ранее в этом же пакете была выполнена процедура DENY_ALL.

```
DBMS_DISTRIBUTED_TRUST_ADMIN.DENY_ALL;
```

Очищает список доверенных серверов и запрещает доступ ко всем серверам.

```
DBMS_DISTRIBUTED_TRUST_ADMIN.DENY_SERVER
    (server IN VARCHAR2);
```

Запрещает доступ к указанному серверу, даже если ранее в этом же пакете была выполнена процедура ALLOW_ALL.

DBMS_FGA

Содержит процедуры, позволяющие управлять политикой детального аудита. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_FGA.ADD_POLICY
    (object_schema IN VARCHAR2,
    object_name IN VARCHAR2,
    policy_name IN VARCHAR2,
    audit_condition IN VARCHAR2,
    audit_column IN VARCHAR2,
    handler_schema IN VARCHAR2,
    handler_module IN VARCHAR2,
    enable BOOLEAN);
```

Назначает политику *policy_name* и обработчик событий *handler_module* для аудита объекта *object_name* в схеме *object_schema*.

```
PROCEDURE DBMS_FGA.DROP_POLICY
    (object_schema IN VARCHAR2,
    object_name IN VARCHAR2,
    policy_name IN VARCHAR2);
```

Удаляет политику *policy_name* для объекта *object_name* в схеме *object_schema*.

```
PROCEDURE DBMS_FGA.ENABLE_POLICY
    (object_schema IN VARCHAR2 DEFAULT NULL,
    object_name IN VARCHAR2,
    policy_name IN VARCHAR2,
    enable BOOLEAN DEFAULT TRUE);
```

Включает политику аудита *policy_name*.

```
PROCEDURE DBMS_FGA.DISABLE_POLICY
    (object_schema IN VARCHAR2 DEFAULT NULL,
    object_name IN VARCHAR2,
    policy_name IN VARCHAR2);
```

Отключает политику аудита *policy_name*.

DBMS_FLASHBACK

Содержит процедуры, позволяющие управлять ретроспективными запросами. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_FLASHBACK.ENABLE_AT_TIME
   (query_time IN TIMESTAMP);
```

Включает ретроспективный режим для всего сеанса для моментального снимка на момент времени *query_time*.

```
PROCEDURE DBMS_FLASHBACK.ENABLE_AT_SYSTEM_CHANGE_NUMBER
   (query_scn IN NUMBER);
```

Включает ретроспективный режим для всего сеанса для моментального снимка, определяемого системным номером изменения *query_scn*.

```
FUNCTION DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER()
   RETURN NUMBER;
```

Возвращает текущий системный номер изменения (SCN) для дальнейшего использования.

```
PROCEDURE DBMS_FLASHBACK.DISABLE;
```

Выключает ретроспективный режим для всего сеанса.

DBMS_HS

Содержит процедуры для работы с гетерогенными сервисами (Heterogeneous Services – HS). Не поддерживается в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_HS.ALTER_BASE_CAPS
    (CAP_number IN NUMBER,
    new_CAP_number IN NUMBER := -1e-130,
    new_CAP_description IN VARCHAR2 := '-');
```

В таблице HS$_BASE_CAPS изменяет строку, определяемую значением *CAP_number*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.ALTER_BASE_DD
    (DD_table_name IN VARCHAR2,
    new_DD_table_name IN VARCHAR := '-',
    new_DD_table_name_descr IN VARCHAR2 := '-');
```

Изменяет строку, определяемую параметром *DD_table_name*, в таблице HS$_BASE_DD. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.ALTER_CLASS_CAPS
    (FDS_class_name IN VARCHAR2,
    CAP_number IN NUMBER,
    new_FDS_class_name IN VARCHAR2 := '-',
    new_CAP_number IN NUMBER := -1e-130,
    new_context IN NUMBER := -1e-130,
    new_translation IN VARCHAR2 := '-',
    new_additional_info NUMBER := -1e-130);
```

Изменяет строку, определяемую параметром *CAP_number*, в таблице HS$_CLASS_CAPS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.ALTER_CLASS_DD
    (FDS_class_name IN VARCHAR2,
    DD_table_name IN VARCHAR2,
    new_FDS_class_name IN VARCHAR2 := '-',
    new_DD_table_name IN NUMBER := -1e-130,
    new_translation_type IN CHAR := '-',
    new_translation_text IN VARCHAR2 := '-');
```

Изменяет содержимое таблицы HS$_CLASS_DD. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.ALTER_CLASS_INIT
    (FDS_class_name IN VARCHAR2,
    init_value_name IN VARCHAR2,
    new_FDS_class_name IN VARCHAR2 := '-',
    new_init_value_name IN VARCHAR2 := '-',
    new_init_value IN VARCHAR2 := '-',
    new_init_value_type IN VARCHAR2 := '-');
```

Изменяет содержимое таблицы HS$_CLASS_INIT. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.ALTER_FDS_CLASS
    (FDS_class_name IN VARCHAR2,
```

```
    new_FDS_class_name IN VARCHAR2 := '-',
    new_FDS_class_comments IN VARCHAR2 := '-');
```

Изменяет содержимое таблицы HS$_FDS_CLASS для класса *FDS_class_name*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.ALTER_FDS_INST
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    new_FDS_inst_name IN VARCHAR2 := '-',
    new_FDS_class_name IN VARCHAR2 := '-',
    new_FDS_class_comments IN VARCHAR2 := '-');
```

Изменяет содержимое таблицы HS$_FDS_INST для экземпляра *FDS_inst_name* класса *FDS_class_name*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.ALTER_INST_CAPS
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    CAP_number IN NUMBER,
    new_FDS_inst_name IN VARCHAR2 := '-',
    new_FDS_class_name IN VARCHAR2 := '-',
    new_CAP_number IN NUMBER := -1e-130,
    new_context IN NUMBER := -1e-130,
    new_translation IN VARCHAR2 := '-',
    new_additional_info IN NUMBER := -1e-130);
```

Изменяет содержимое таблицы HS$_INST_CAPS для экземпляра *FDS_inst_name* класса *FDS_class_name*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.ALTER_INST_DD
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    DD_table_name IN VARCHAR2,
    new_FDS_inst_name IN VARCHAR2 := '-',
    new_FDS_class_name IN VARCHAR2 := '-',
    new_DD_table_name IN VARCHAR2 := '-',
    new_translation_type IN CHAR := '-',
    new_transation_text VARCHAR2 := '-');
```

Изменяет содержимое таблицы HS$_INST_DD для экземпляра *FDS_inst_name* класса *FDS_class_name* и таблицы *DD_table_name*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.ALTER_INST_INIT
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    init_value_name IN VARCHAR2,
    new_FDS_inst_name IN VARCHAR2 := '-',
    new_FDS_class_name IN VARCHAR2 := '-',
    new_init_value_name IN VARCHAR2 := '-',
    new_init_value IN VARCHAR2 := '-',
    new_init_value_type IN VARCHAR2 := '-');
```

Изменяет содержимое таблицы HS$_INST_INIT для экземпляра *FDS_inst_name* класса *FDS_class_name*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.COPY_CLASS
    (old_FDS_class_name IN VARCHAR2);
```

Копирует все из класса *old_FDS_class_name* в другой класс. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.COPY_INST
    (FDS_inst_name IN VARCHAR2,
```

```
    FDS_class_name IN VARCHAR2,
    new_FDS_inst_name IN VARCHAR2 := '-',
    new_FDS_class_comments IN VARCHAR2 := '-');
```

Создает новый экземпляр *new_FDS_inst_name* в таблице HS$_FDS_INST для экземпляра *FDS_inst_name* класса *FDS_class_name*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.CREATE_BASE_CAPS
    (CAP_number IN NUMBER,
    CAP_description IN VARCHAR2 := NULL);
```

Создает строку в таблице HS$_BASE_CAPS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.CREATE_BASE_DD
    (DD_table_name IN VARCHAR2,
    DD_table_descr IN VARCHAR2 := NULL);
```

Создает строку в таблице HS$_BASE_DD. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.CREATE_CLASS_CAPS
    (FDS_class_name IN VARCHAR2,
    CAP_number IN NUMBER,
    context IN NUMBER := NULL,
    translation IN VARCHAR2 := NULL,
    additional_info NUMBER := NULL);
```

Создает строку в таблице HS$_CLASS_CAPS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.CREATE_CLASS_DD
    (FDS_class_name IN VARCHAR2,
    DD_table_name IN VARCHAR2,
    translation_type IN CHAR,
    translation_text IN VARCHAR2);
```

Создает строку в таблице HS$_CLASS_DD. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.CREATE_CLASS_INIT
    (FDS_class_name IN VARCHAR2,
    init_value_name IN VARCHAR2,
    init_value IN VARCHAR2,
    init_value_type IN VARCHAR2);
```

Создает строку в таблице HS$_CLASS_INIT. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.CREATE_FDS_CLASS
    (FDS_class_name IN VARCHAR2,
    FDS_class_comments IN VARCHAR2);
```

Создает строку в таблице HS$_FDS_CLASS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.CREATE_FDS_INST
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    FDS_inst_comments IN VARCHAR2 := NULL);
```

Создает строку в таблице HS$_FDS_INST.

```
PROCEDURE DBMS_HS.CREATE_INST_CAPS
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    CAP_number IN NUMBER,
    context IN NUMBER := NULL,
    translation IN VARCHAR2 := NULL,
    new_additional_info IN NUMBER);
```

Создает строку в таблице HS$_INST_CAPS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.CREATE_INST_DD
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    DD_table_name IN VARCHAR2,
    translation_type IN CHAR,
    transation_text VARCHAR2 := NULL);
```

Создает строку в таблице HS$_INST_DD. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.CREATE_INST_INIT
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    init_value_name IN VARCHAR2,
    init_value IN VARCHAR2,
    init_value_type IN VARCHAR2);
```

Создает строку в таблице HS$_INST_INIT. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.DROP_BASE_CAPS
    (CAP_number IN NUMBER);
```

Удаляет строку из таблицы HS$_BASE_CAPS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.CREATE_BASE_DD
    (DD_table_name IN VARCHAR2,);
```

Удаляет строку из таблицы HS$_BASE_DD. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.DROP_CLASS_CAPS
    (FDS_class_name IN VARCHAR2,
    CAP_number IN NUMBER);
```

Удаляет строку из таблицы HS$_CLASS_CAPS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.DROP_CLASS_DD
    (FDS_class_name IN VARCHAR2,
    DD_table_name IN VARCHAR2);
```

Удаляет строку из таблицы HS$_CLASS_DD. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.DROP_CLASS_INIT
    (FDS_class_name IN VARCHAR2,
    init_value_name IN VARCHAR2);
```

Удаляет строку из таблицы HS$_CLASS_INIT. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.DROP_FDS_CLASS
    (FDS_class_name IN VARCHAR2);
```

Удаляет строку из таблицы HS$_FDS_CLASS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.DROP_FDS_INST
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2);
```

Удаляет строку из таблицы HS$_FDS_INST. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.DROP_INST_CAPS
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    CAP_number IN NUMBER);
```

Удаляет строку из таблицы HS$_INST_CAPS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.DROP_INST_DD
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
```

```
    DD_table_name IN VARCHAR2);
```

Удаляет строку из таблицы HS$_INST_DD. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.CREATE_INST_INIT
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    init_value_name IN VARCHAR2);
```

Удаляет строку из таблицы HS$_INST_INIT.

```
PROCEDURE DBMS_HS.REPLACE_BASE_CAPS
    (CAP_number IN NUMBER,
    new_CAP_number IN NUMBER := NULL,
    new_CAP_description IN VARCHAR2 := NULL);
```

Выполняет операцию создания или замены для таблицы HS$_BASE_CAPS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.REPLACE_BASE_DD
    (DD_table_name IN VARCHAR2,
    new_DD_table_name IN VARCHAR := NULL,
    new_DD_table_name_descr IN VARCHAR2 := NULL);
```

Выполняет операцию создания или замены для таблицы HS$_BASE_DD. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.REPLACE_CLASS_CAPS
    (FDS_class_name IN VARCHAR2,
    CAP_number IN NUMBER,
    new_FDS_class_name IN VARCHAR2 := NULL,
    new_CAP_number IN NUMBER := NULL,
    new_context IN NUMBER := NULL,
    new_translation IN VARCHAR2 := NULL,
    new_additional_info NUMBER := NULL);
```

Выполняет операцию создания или замены для таблицы HS$_CLASS_CAPS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.REPLACE_CLASS_DD
    (FDS_class_name IN VARCHAR2,
    DD_table_name IN VARCHAR2,
    new_FDS_class_name IN VARCHAR2 := NULL,
    new_DD_table_name IN NUMBER := NULL,
    new_translation_type IN CHAR := NULL,
    new_translation_text IN VARCHAR2 := NULL);
```

Выполняет операцию создания или замены для таблицы HS$_CLASS_DD. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.REPLACE_CLASS_INIT
    (FDS_class_name IN VARCHAR2,
    init_value_name IN VARCHAR2,
    new_FDS_class_name IN VARCHAR2 := NULL,
    new_init_value_name IN VARCHAR2 := NULL,
    new_init_value IN VARCHAR2 := NULL,
    new_init_value_type IN VARCHAR2 := NULL);
```

Выполняет операцию создания или замены для таблицы HS$_CLASS_INIT. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.REPLACE_FDS_CLASS
    (FDS_class_name IN VARCHAR2,
```

```
    new_FDS_class_name IN VARCHAR2 := NULL,
    new_FDS_class_comments IN VARCHAR2 := NULL);
```

Выполняет операцию создания или замены для таблицы HS$_FDS_CLASS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.REPLACE_FDS_INST
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    new_FDS_inst_name IN VARCHAR2 := NULL,
    new_FDS_class_name IN VARCHAR2 := NULL,
    new_FDS_class_comments IN VARCHAR2 := NULL);
```

Выполняет операцию создания или замены для таблицы HS$_FDS_INST. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.REPLACE_INST_CAPS
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    CAP_number IN NUMBER,
    new_FDS_inst_name IN VARCHAR2 := NULL,
    new_FDS_class_name IN VARCHAR2 := NULL,
    new_CAP_number IN NUMBER := NULL,
    new_context IN NUMBER := NULL,
    new_translation IN VARCHAR := NULL,
    new_additional_info IN NUMBER := NULL);
```

Выполняет операцию создания или замены для таблицы HS$_INST_CAPS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.REPLACE_INST_DD
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    DD_table_name IN VARCHAR2,
    new_FDS_inst_name IN VARCHAR2 := NULL,
    new_FDS_class_name IN VARCHAR2 := NULL,
    new_DD_table_name IN VARCHAR2 := NULL,
    new_translation_type IN CHAR := NULL,
    new_translation_text VARCHAR2 := NULL);
```

Выполняет операцию создания или замены для таблицы HS$_INST_DD. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_HS.REPLACE_INST_INIT
    (FDS_inst_name IN VARCHAR2,
    FDS_class_name IN VARCHAR2,
    init_value_name IN VARCHAR2,
    new_FDS_inst_name IN VARCHAR2 := NULL,
    new_FDS_class_name IN VARCHAR2 := NULL,
    new_init_value_name IN VARCHAR2 := NULL,
    new_init_value IN VARCHAR2 := NULL,
    new_init_value_type IN VARCHAR2 := NULL);
```

Выполняет операцию создания или замены для таблицы HS$_INST_INIT. Появилась в Oracle8*i*.

DBMS_HS_PASSTHROUGH

Управляет отправкой команд непосредственно на не-Oracle сервер без интерпретации (Heterogeneous Services). Все программы работают так, как если бы они находились на удаленном сервере.

Вызовы

```
PROCEDURE DBMS_HS_PASSTHROUGH.BIND_VARIABLE
    (c IN BINARY_INTEGER NOT NULL,
    pos IN BINARY_INTEGER NOT NULL,
    val IN {DATE | NUMBER | VARCHAR2},
    name IN VARCHAR2);
```

Связывает значение *val* с переменой курсора *c* в позиции *pos* на удаленной системе.

```
PROCEDURE DBMS_HS_PASSTHROUGH.BIND_VARIABLE_RAW
    (c IN BINARY_INTEGER NOT NULL,
    pos IN BINARY_INTEGER NOT NULL,
    val IN RAW,
    name IN VARCHAR2);
```

Связывает значение *val* типа RAW с переменой курсора *c* в позиции *pos* на удаленной системе.

```
PROCEDURE DBMS_HS_PASSTHROUGH.BIND_OUT_VARIABLE
    (c IN BINARY_INTEGER NOT NULL,
    pos IN BINARY_INTEGER NOT NULL,
    val OUT {DATE | NUMBER | VARCHAR2},
    name IN VARCHAR2);
```

Связывает выходную переменную курсора *c* в позиции *pos* с переменной *val* программы PL/SQL.

```
PROCEDURE DBMS_HS_PASSTHROUGH.BIND_OUT_VARIABLE_RAW
    (c IN BINARY_INTEGER NOT NULL,
    pos IN BINARY_INTEGER NOT NULL,
    val OUT RAW,
    name IN VARCHAR2);
```

Связывает выходную переменную типа RAW курсора *c* в позиции *pos* с переменной *val* программы PL/SQL.

```
PROCEDURE DBMS_HS_PASSTHROUGH.BIND_INOUT_VARIABLE
    (c IN BINARY_INTEGER NOT NULL,
    pos IN BINARY_INTEGER NOT NULL,
    val IN OUT {DATE | NUMBER | VARCHAR2},
    name IN VARCHAR2);
```

Связывает IN OUT значение *val* в позиции *pos* курсора *c* с переменной программы PL/SQL.

```
PROCEDURE DBMS_HS_PASSTHROUGH.BIND_INOUT_VARIABLE_RAW
    (c IN BINARY_INTEGER NOT NULL,
    pos IN BINARY_INTEGER NOT NULL,
    val IN OUT RAW,
    name IN VARCHAR2);
```

Связывает IN OUT значение *val* типа RAW в позиции *pos* курсора *c* с переменной программы PL/SQL.

```
PROCEDURE DBMS_HS_PASSTHROUGH.CLOSE_CURSOR
    (c IN BINARY_INTEGER NOT NULL);
```

Закрывает курсор *c* и освобождает память после выполнения команды в не-Oracle системе.

```
FUNCTION DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE
    (s IN VARCHAR2 NOT NULL)
    RETURN BINARY_INTEGER;
```

Осуществляет непосредственное выполнение SQL-команды *s*. Помните о том, что команды SELECT не могут быть выполнены непосредственно.

```
FUNCTION DBMS_HS_PASSTHROUGH.EXECUTE_NON_QUERY
    (c IN BINARY_INTEGER NOT NULL)
    RETURN BINARY_INTEGER;
```

Выполняет разобранную команду SQL курсора *c*. Такой командой не может быть SELECT.

```
FUNCTION DBMS_HS_PASSTHROUGH.FETCH_ROW
    (c IN BINARY_INTEGER NOT NULL,
    first IN BOOLEAN)
    RETURN BINARY_INTEGER;
```

Извлекает строки из результирующего множества курсора *c*. Если значение параметра *first* равно TRUE, то команда SELECT выполняется повторно, в противном случае извлекается следующая строка (или первая строка для запроса, который еще не выполнялся).

```
PROCEDURE DBMS_HS_PASSTHROUGH.GET_VALUE
    (c IN BINARY_INTEGER NOT NULL,
    pos IN BINARY_INTEGER NOT NULL,
    val OUT {DATE | NUMBER | VARCHAR2});
```

Извлекает или список элементов в команде SELECT, или список связанных выходных значений после выполнения команды SQL.

```
PROCEDURE DBMS_HS_PASSTHROUGH.GET_VALUE_RAW
    (c IN BINARY_INTEGER NOT NULL,
    pos IN BINARY_INTEGER NOT NULL,
    val OUT RAW);
```

Извлекает значения типа RAW. Эта процедура аналогична процедуре GET_VALUE.

```
FUNCTION DBMS_HS_PASSTHROUGH.OPEN_CURSOR
    RETURN BINARY_INTEGER;
```

Открывает курсор в не-Oracle системе.

```
PROCEDURE DBMS_HS_PASSTHROUGH.PARSE
    (c IN BINARY_INTEGER NOT NULL,
    stmt IN VARCHAR2 NOT NULL);
```

Выполняет синтаксический анализ команды *stmt* для курсора *c* в не-Oracle системе.

DBMS_IOT

Содержит процедуры, позволяющие создавать таблицы для управления ссылками на расщепленные строки или строки, нарушающие ограничение в индекс-таблицах. Появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_IOT.BUILD_CHAIN_ROWS_TABLE
    (owner IN VARCHAR2,
    iot_name IN VARCHAR2.
    chainrow_table_name IN VARCHAR2 DEFAULT 'IOT_CHAINED_ROWS');
```

Строит таблицу *chainrow_table_name* для хранения ссылок на расщепленные строки в таблице *iot_name*, принадлежащей владельцу *owner*.

```
PROCEDURE DBMS_IOT.BUILD_EXCEPTIONS_TABLE
    (owner IN VARCHAR2,
    iot_name IN VARCHAR2.
    exceptions_table_name IN VARCHAR2 DEFAULT 'IOT_EXCEPTIONS_ROWS');
```

Строит таблицу *exceptions_table_name* для хранения нарушающих ограничения строк таблицы *iot_name*, принадлежащей владельцу *owner*.

DBMS_JOB

Предоставляет интерфейс для подсистемы очереди заданий Oracle, который обеспечивает автоматическое планирование и выполнение программ PL/SQL.

Вызовы

```
PROCEDURE DBMS_JOB.BROKEN
    (job IN BINARY_INTEGER,
    broken IN BOOLEAN,
    next_date IN DATE DEFAULT SYSDATE);
```

Устанавливает или снимает логический флаг *broken* для задания *job* и (необязательно) задает следующую дату выполнения *next_date*. Задания, для которых флаг *broken* установлен в TRUE, автоматически не выполняются.

```
PROCEDURE DBMS_JOB.CHANGE
    (job IN BINARY_INTEGER,
    what IN VARCHAR2,
    next_date IN DATE,
    interval IN VARCHAR2,
    instance IN BINARY_INTEGER DEFAULT NULL,
    force IN BOOLEAN DEFAULT FALSE);
```

Изменяет один или несколько параметров, таких как *what*, *next_date*, *interval* или *instance* (появившийся в Oracle8*i*), для задания *job*. Если значение параметра *force* (также появившегося в Oracle8*i*) равно TRUE, то для *instance* допустимо любое целое положительное значение; если же значение *force* равно FALSE, то экземпляр *instance* должен быть запущен.

```
PROCEDURE DBMS_JOB.INTERVAL
    (job IN BINARY_INTEGER,
    interval IN VARCHAR2);
```

Изменяет выражение даты *interval*, которое применяется для определения момента следующего выполнения задания *job*.

```
PROCEDURE DBMS_JOB.SUBMIT
    (job IN BINARY_INTEGER,
    what IN VARCHAR2,
    next_date IN VARCHAR2,
    interval IN VARCHAR2 DEFAULT 'null',
    no_parse IN BOOLEAN DEFAULT FALSE,
    instance IN BINARY_INTEGER DEFAULT any_instance,
    force IN BOOLEAN DEFAULT FALSE);
```

Передает задание с указанным номером *job* и PL/SQL-определением *what*, выполнение которого было запланировано на время *next_date* и далее через интервалы *interval*, экземпляру *instance* (параметр появился в Oracle8*i*). Если значение параметра *no_parse* равно TRUE, то синтаксический анализ PL/SQL-кода в *what* откладывается до выполнения. Если значение параметра *force* (появившегося в Oracle8*i*)

равно TRUE, то для *instance* допустимо любое целое положительное значение; если же значение *force* равно FALSE, то экземпляр *instance* должен быть запущен.

```
PROCEDURE DBMS_JOB.NEXT_DATE
    (job IN BINARY_INTEGER,
    next_date IN DATE);
```

Изменяет следующую запланированную дату выполнения задания *job* на *next_date*.

```
PROCEDURE DBMS_JOB.REMOVE
    (job IN BINARY_INTEGER);
```

Удаляет задание *job* из очереди. Если в данный момент задание *job* выполняется, оно будет работать до нормального завершения, но дальнейшее его выполнение не будет запланировано.

```
PROCEDURE DBMS_JOB.RUN
    (job IN BINARY_INTEGER
    force IN BOOLEAN DEFAULT TRUE);
```

Незамедлительно выполняет задание *job* в текущем сеансе. Если значение параметра *force* (появившегося в Oracle8*i*) равно FALSE, то задание *job* может запускаться только как приоритетный процесс для указанного экземпляра.

```
PROCEDURE DBMS_JOB.USER_EXPORT
    (job IN BINARY_INTEGER,
    mycall IN OUT VARCHAR2
    [myinst IN OUT VARCHAR2]);
```

Возвращает в *mycall* символьную строку, которая может использоваться для повторной передачи задания *job* в очередь. Если указан параметр *myinst*, то процедура изменяет привязку экземпляра. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_JOB.WHAT
    (job IN BINARY_INTEGER,
    what IN VARCHAR2);
```

Изменяет PL/SQL-определение задания *job* на *what*.

```
PROCEDURE DBMS_JOB.INSTANCE
    (job IN BINARY_INTEGER,
    instance IN BINARY_INTEGER,
    force IN BOOLEAN DEFAULT FALSE);
```

Изменяет привязку задания к экземпляру. Если значение параметра *force* равно TRUE, то для *instance* допустимо любое целое положительное значение; если же значение *force* равно FALSE, то *instance* должен быть работающим экземпляром. Появилась в Oracle8*i*.

DBMS_LDAP

Содержит процедуры и функции, позволяющие обращаться к данным , получаемым от серверов LDAP. Пакет должен быть загружен в БД с помощью сценария *catldap.sql*. Появился в Oracle9*i*.

Вызовы

```
FUNCTION DBMS_LDAP.INIT
    (hostname IN VARCHAR2,
```

```
    portnum IN PLS_INTEGER)
    RETURN SESSION;
```

Устанавливает соединение с LDAP-сервером на хосте *hostname* через порт *portnum*.

```
FUNCTION DBMS_LDAP.SIMPLE_BIND_S
    (ld IN SESSION,
    dn IN VARCHAR2,
    passwd IN VARCHAR2)
    RETURN PLS_INTEGER;
```

Выполняет простую аутентификацию по имени пользователя (*dn*) и паролю (*passwd*). Возвращает флаг завершения.

```
FUNCTION DBMS_LDAP.BIND_S
    (ld IN SESSION,
    dn IN VARCHAR2,
    cred IN VARCHAR2,
    meth IN PLS_INTEGER)
    RETURN PLS_INTEGER;
```

Выполняет комплексную аутентификацию с помощью метода *meth*. Возвращает флаг завершения.

```
FUNCTION DBMS_LDAP.UNBIND_S
    (ld IN SESSION)
    RETURN PLS_INTEGER;
```

Закрывает активный сеанс *ld*. Возвращает флаг завершения.

```
FUNCTION DBMS_LDAP.COMPARE_S
    (ld IN SESSION,
    dn IN VARCHAR2,
    attr IN VARCHAR2,
    value IN VARCHAR2)
    RETURN PLS_INTEGER;
```

Сравнивает атрибут *attr* элемента *dn* со значением *value*. Возвращает флаг завершения.

```
FUNCTION DBMS_LDAP.SEARCH_S
    (ld IN SESSION,
    base IN VARCHAR2,
    scope IN PLS_INTEGER,
    filter IN VARCHAR2,
    attrs IN STRING_COLLECTION,
    attronly IN PLS_INTEGER,
    res OUT MESSAGE)
    RETURN PLS_INTEGER;
```

Выполняет синхронный поиск атрибутов *attrs* на сервере *ld*. Возвращает флаг завершения.

```
FUNCTION DBMS_LDAP.SEARCH_ST
    (ld IN SESSION,
    base IN VARCHAR2,
    scope IN PLS_INTEGER,
    filter IN VARCHAR2,
    attrs IN STRING_COLLECTION,
    attronly IN PLS_INTEGER,
    tv IN TIMEVAL,
    res OUT MESSAGE)
    RETURN PLS_INTEGER;
```

Выполняет синхронный поиск атрибутов *attrs* на сервере *ld* с таймаутом *tv* с клиентской стороны. Возвращает флаг завершения.

```
FUNCTION DBMS_LDAP.FIRST_ENTRY
    (ld IN SESSION,
    msg IN MESSAGE)
    RETURN MESSAGE;
```

Извлекает первый элемент результирующего множества *msg*, полученного функцией SEARCH_S или SEARCH_ST.

```
FUNCTION DBMS_LDAP.NEXT_ENTRY
    (ld IN SESSION,
    msg IN MESSAGE)
    RETURN MESSAGE;
```

Извлекает очередной элемент результирующего множества *msg*, полученного функцией SEARCH_S или SEARCH_ST.

```
FUNCTION DBMS_LDAP.COUNT_ENTRIES
    (ld IN SESSION,
    msg IN MESSAGE)
    RETURN PLS_INTEGER;
```

Возвращает количество элементов в результирующем множестве *msg*, полученном функцией SEARCH_S или SEARCH_ST.

```
FUNCTION DBMS_LDAP.FIRST_ATTRIBUTE
    (ld IN SESSION,
    msg IN MESSAGE,
    ber_elem OUT BER_ELEMENT)
    RETURN VARCHAR2;
```

Возвращает первый атрибут указанного элемента.

```
FUNCTION DBMS_LDAP.NEXT_ATTRIBUTE
    (ld IN SESSION,
    msg IN MESSAGE,
    ber_elem IN BER_ELEMENT)
    RETURN VARCHAR2;
```

Возвращает очередной атрибут указанного элемента.

```
FUNCTION DBMS_LDAP.GET_DN
    (ld IN SESSION,
    msg IN MESSAGE)
    RETURN VARCHAR2;
```

Возвращает отличительное имя указанного элемента.

```
FUNCTION DBMS_LDAP.GET_VALUES
    (ld IN SESSION,
    ldapentry IN MESSAGE,
    attr IN VARCHAR2)
    RETURN STRING_COLLECTION;
```

Возвращает все значения, связанные с атрибутом *attr* элемента *ldapentry*.

```
FUNCTION DBMS_LDAP.GET_VALUES_LEN
    (ld IN SESSION,
    ldapentry IN MESSAGE,
    attr IN VARCHAR2)
    RETURN BINVAL_COLLECTION;
```

Возвращает все значения двоичного типа, связанные с атрибутом *attr* элемента *ldapentry*.

```
FUNCTION DBMS_LDAP.DELETE_S
    (ld IN SESSION,
    entrydn IN VARCHAR2)
    RETURN PLS_INTEGER;
```

Удаляет листовой (терминальный) элемент *entrydn* из каталога *ld*. Возвращает флаг завершения.

```
FUNCTION DBMS_LDAP.MODRDN2_S
    (ld IN SESSION,
    entrydn IN VARCHAR2,
    newrdn IN VARCHAR2,
    deleteoldrdn IN PLS_INTEGER)
    RETURN PLS_INTEGER;
```

Изменяет относительное отличительное имя элемента *entrydn* на *newrdn*. Если параметр *deleteoldr* не равен 0, то старое относительное отличительное имя будет удалено. Возвращает флаг завершения.

```
FUNCTION DBMS_LDAP.ERR2STRING
    (ldap_err IN PLS_INTEGER)
    RETURN VARCHAR2;
```

Преобразует номер ошибки в сообщение об ошибке.

```
FUNCTION DBMS_LDAP.CREATE_MOD_ARRAY
    (num IN PLS_INTEGER)
    RETURN MOD_ARRAY;
```

Выделяет память для *num* элементов массива изменений, выполняемых функцией MODIFY_S или ADD_S.

```
PROCEDURE DBMS_LDAP.POPULATE_MOD_ARRAY
    (modptr IN DBMS_LDAP.MOD_ARRAY,
    mod_op IN PLS_INTEGER,
    mod_type IN VARCHAR2,
    modval IN {DBMS_LDAP.STRING_COLLECTION | DBMS_LDAP.BERVAL_ COLLECTION});
```

Передает указатель (*modptr*) для модификации атрибута, имеющего тип *mod_type*, значением *modval*; тип модификации – *mod_op*. Вызов процедуры работает для строк или двоичных значений с различными типами данных *modval*.

```
FUNCTION DBMS_LDAP.MODIFY_S
    (ld IN SESSION,
    entrydh IN VARCHAR2,
    modptr IN DBMS_LDAP.MOD_ARRAY)
    RETURN PLS_INTEGER;
```

Выполняет синхронное обновление элемента каталога LDAP определяемого массивом *modptr*. Возвращает флаг завершения.

```
FUNCTION DBMS_LDAP.ADD_S
    (ld IN SESSION,
    entrydh IN VARCHAR2,
    modptr IN DBMS_LDAP.MOD_ARRAY)
    RETURN PLS_INTEGER;
```

Выполняет синхронное добавление элемента в каталог LDAP в соответствии с данными в *modptr*. Прежде чем вызывать данную функцию, необходимо вызвать CREATE_MOD_ARRAY и POPULATE_MOD_ARRAY. Возвращает флаг завершения.

```
PROCEDURE DBMS_LDAP.FREE_MOD_ARRAY
    (modptr IN DBMS_LDAP.MOD_ARRAY);
```

Освобождает память, выделенную для *modptr*.

```
FUNCTION DBMS_LDAP.COUNT_VALUES
    (values IN DBMS_LDAP.STRING_COLLECTION_
    RETURN PLS_INTEGER;
```

Подсчитывает количество значений, возвращенных вызовом GET_VALUES.

```
FUNCTION DBMS_LDAP.COUNT_VALUES_LEN
    (values IN DBMS_LDAP.STRING_COLLECTION_
    RETURN PLS_INTEGER;
```

Подсчитывает количество значений, возвращенных вызовом GET_VALUES_LEN.

```
FUNCTION DBMS_LDAP.RENAME_S
    (ld IN SESSION,
    entrydn IN VARCHAR2,
    newrdn IN VARCHAR2,
    deleteoldrdn IN PLS_INTEGER,
    serverctrls IN LDAPCONTROL,
    clientcntrls IN LDAPCONTROL)
    RETURN PLS_INTEGER;
```

Синхронно переименовывает элемент LDAP. Если параметр *deleteoldr* не равен 0, то старое относительное отличительное имя будет удалено. Параметры *serverctrls* и *clientcntrls* не поддерживаются в Oracle9*i*. Возвращает флаг завершения.

```
FUNCTION DBMS_LDAP.EXPLODE_DN
    (dn IN VARCHAR2,
    notypes IN PLS_INTEGER)
    RETURN STRING_COLLECTION;
```

Возвращает отличительное имя *dn*, разбитое на части. Если параметр *notypes* равен 0, то включаются атрибуты тегов.

```
FUNCTION DBMS_LDAP.OPEN_SSL
    (ld IN SESSION,
    sslwr1 IN VARCHAR2,
    sslwalletpasswd IN VARCHAR2,
    sslauth IN PLS_INTEGER)
    RETURN PLS_INTEGER;
```

Устанавливает SSL-соединение поверх существующего LDAP-соединения. Возвращает флаг завершения.

DBMS_LIBCACHE

Подготавливает библиотечный кэш экземпляра Oracle, извлекая код SQL и PL/SQL из удаленного экземпляра и перекомпилируя его локально. Появился в Oracle9*i*.

Вызов

```
PROCEDURE DBMS_LIBCACHE.COMPILE_CURSORS_FROM_REMOTE
    (dblink IN VARCHAR2,
    username IN VARCHAR2,
    execution_threshold IN NUMBER,
    shared_mem_threshold IN NUMBER);
```

Выполняет групповое извлечение кода SQL из *dblink* и проводит его синтаксический анализ от имени пользователя *username*. Будут извлечены только SQL-команды, выполняемые не менее *execution_threshold* раз и требующие не менее *shared_mem_threshold* разделяемой памяти.

DBMS_LOB

Предоставляет механизм доступа к большим объектам (LOB) и работы с ними. К большим объектам относятся большие двоичные объекты (Binary Large Objects – BLOBs), большие символьные объекты (Character Large Objects – CLOBs), большие символьные объекты в национальной кодировке (National Language Support Character Large Objects – NCLOBs) и двоичные файлы BFILE. Инструкции CHARACTER SET ANY_CS в объявлениях CLOB допускают использование указателей (*locator*) CLOB или NCLOB.

Вызовы

```
PROCEDURE DBMS_LOB.APPEND
    ({dest_lob IN OUT BLOB | dest_lob IN OUT CLOB CHARACTER SET ANY_CS},
    {src_lob IN BLOB | src_lob IN CLOB CHARACTER SET dest_lob%CHARSET});
```

Добавляет содержимое исходного LOB *src_lob* в LOB назначения *dest_lob*. Оба больших объекта должны относиться к одному из типов LOB: BLOB, CLOB или NCLOB.

```
PROCEDURE DBMS_LOB.CLOSE
    ({lob_loc IN OUT NOCOPY BLOB |
    lob_loc IN OUT NOCOPY CLOB CHARACTER SET ANY_CS |
    file_loc IN OUT NOCOPY BFILE});
```

Закрывает ранее открытый BLOB, CLOB или внешний файл. Появилась в Oracle9*i*.

```
FUNCTION DBMS_LOB.COMPARE
    (lob_1 IN {BLOB | CLOB CHARACTER SET ANY_CS},
    lob_2 IN {BLOB | CLOB CHARACTER SET lob_1%CHARSET},
    amount IN INTEGER DEFAULT 4294967295,
    offset_1 IN INTEGER DEFAULT 1,
    offset_2 IN INTEGER DEFAULT 1)
RETURN INTEGER;

FUNCTION DBMS_LOB.COMPARE
    (file_1 IN BFILE,
    file_2 IN BFILE,
    amount IN INTEGER,
    offset_1 IN INTEGER DEFAULT 1,
    offset_2 IN INTEGER DEFAULT 1)
RETURN INTEGER;
```

Сравнивает не более *amount* байт входных больших объектов *lob_1* и *lob_2* или *file_1* и *file_2* начиная сравнение с позиций *offset_1* и *offset_2*. Оба входных значения должны относиться к одному из типов LOB: BLOB, CLOB, NCLOB или BFILE.

Возвращает 0, если входные объекты точно совпадают друг с другом; ненулевое значение в случае несовпадения или NULL, если *amount*, *offset_1* или *offset_2* меньше 1 или же больше LOBMAXSIZE (константа данного пакета, имеющая значение 4 294 967 295).

```
PROCEDURE DBMS_LOB.COPY
    (dest_lob IN OUT {BLOB | CLOB CHARACTER SET ANY_CS},
    src_lob IN {BLOB | CLOB CHARACTER SET dest_lob%CHARSET},
```

```
    amount IN INTEGER,
    dest_offset IN INTEGER DEFAULT 1,
    src_offset IN INTEGER DEFAULT 1);
```

Копирует *amount* байт (BLOB) или символов (CLOB), расположенных начиная с позиции *src_offset* байт или символов исходного большого объекта *lob_loc*, в объект назначения *dest_lob* начиная с позиции *dest_offset*. Оба больших объекта должны относиться к одному из типов LOB: BLOB, CLOB или NCLOB.

```
PROCEDURE DBMS_LOB.CREATETEMPORARY
    (lob_loc IN OUT NOCOPY {BLOB | CLOB CHARACTER SET ANY_CS},
    cache IN BOOLEAN,
    dur IN PLS_INTEGER DEFAULT 10);
```

Создает временную копию *lob_loc*, считывает ее в кэш буферов, если это указано параметром *cache*, и указывает, будет ли она существовать в течение сеанса (SESSION, по умолчанию) или вызова (CALL). Появилась в Oracle8*i*.

```
PROCEDURE DBMS_LOB.ERASE
    (lob_loc IN OUT {BLOB | CLOB CHARACTER SET ANY_CS},
    amount IN OUT INTEGER,
    offset IN INTEGER DEFAULT 1);
```

Стирает (заполняет нулевыми байтами или пробелами) *amount* байт (BLOB) или символов (CLOB) большого объекта *lob_loc* начиная с позиции *offset* байт или символов.

```
PROCEDURE DBMS_LOB.FILECLOSE
    (file_loc IN OUT BFILE);
```

Закрывает открытый ранее файл BFILE *file_loc*.

```
PROCEDURE DBMS_LOB.FILECLOSEALL;
```

Закрывает все открытые в текущем сеансе файлы BFILE.

```
FUNCTION DBMS_LOB.FILEEXISTS
    (file_loc IN BFILE)
    RETURN INTEGER;
```

Возвращает 1, если файл BFILE *file_loc* существует и 0 – в противном случае.

```
PROCEDURE DBMS_LOB.FILEGETNAME
    (file_loc IN BFILE,
    dir_alias OUT VARCHAR2,
    filename OUT VARCHAR2);
```

Задает псевдоним каталога (указанного ранее в команде CREATE DIRECTORY) и имя файла для указателя файла *file_loc*.

```
FUNCTION DBMS_LOB.FILEISOPEN
    (file_loc IN BFILE)
    RETURN INTEGER;
```

Возвращает 1, если файл BFILE *file_loc* открыт и 0 – в противном случае.

```
PROCEDURE DBMS_LOB.FILEOPEN
    (file_loc IN OUT BFILE,
    open_mode IN BINARY_INTEGER DEFAULT FILE_READONLY);
```

Открывает файл BFILE *file_loc* только для чтения.

```
PROCECURE DBMS_LOB.FREETEMPORARY
    (lob_loc IN OUT NOCOPY {BLOB | CLOB CHARACTER SET ANY_CS});
```

Освобождает временный BLOB или CLOB *lob_loc*, который затем помечается как недействительный. Появилась в Oracle9*i*.

```
FUNCTION DBMS_LOB.GETCHUNKSIZE
    (lob_loc IN OUT NOCOPY {BLOB | CLOB CHARACTER SET ANY_CS})
    RETURN INTEGER;
```

Возвращает размер фрагмента для *lob_loc*. Появилась в Oracle8*i*.

```
FUNCTION DBMS_LOB.GETLENGTH
    (lob_loc IN {BLOB | CHARACTER SET ANY_CS | BFILE})
    RETURN INTEGER;
```

Возвращает размер большого объекта *lob_loc* в байтах или символах.

```
FUNCTION DBMS_LOB.INSTR
    (lob_loc IN {BLOB | CHARACTER SET ANY_CS | BFILE},
    pattern IN {RAW | VARCHAR2 CHARACTER SET lob_loc%CHARSET | RAW},
    offset IN INTEGER DEFAULT 1,
    nth IN INTEGER DEFAULT 1)
    RETURN INTEGER;
```

Аналогична встроенной функции INSTR. Возвращает позицию *offset* (в байтах (BLOB) или символах (CLOB)) в большом объекте *lob_loc*, в которой встретилось *n*-е вхождение образца *pattern*. Поиск начинается с позиции *offset* байт или символов в *lob_loc*.

```
FUNCTION DBMS_LOB.ISOPEN
    ({lob_loc | file_loc} IN {BLOB | CHARACTER SET ANY_CS | BFILE})
    RETURN INTEGER;
```

Возвращает TRUE, если большой объект или файл уже открыт. Появилась в Oracle8*i*.

```
FUNCTION DBMS_LOB.ISTEMPORARY
    (lob_loc IN {BLOB | CHARACTER SET ANY_CS})
    RETURN INTEGER;
```

Возвращает TRUE, если *lob_loc* указывает на временный LOB. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_LOB.LOADFROMFILE
    (dest_lob IN OUT NOCOPY {BLOB | CLOB CHARACTER SET ANY_CS},
    src_lob IN BFILE,
    amount IN INTEGER,
    dest_offset IN INTEGER := 1,
    src_offset IN INTEGER := 1);
```

Копирует *amount* байт начиная с позиции *src_offset* байт исходного BFILE *src_lob* в позицию *dest_offset* байт файла назначения *dest_lob*. До выхода Oracle9*i* этот вызов мог работать с типом данных CLOB; в версии Oracle9*i* допустим только тип данных BLOB.

```
PROCEDURE DBMS_LOB.LOADBLOBFROMFILE
    (dest_lob IN OUT NOCOPY BLOB,
    src_lob IN BFILE,
    amount IN INTEGER,
    dest_offset IN OUT INTEGER := 1,
    src_offset IN OUT INTEGER := 1);
```

Загружает данные BFILE *src_lob* во внутренний BLOB *dest_lob*. Аналогична процедуре LOADFROMFILE; появилась в Oracle9*i*.

```
PROCEDURE DBMS_LOB.LOADCLOBFROMFILE
    (dest_lob IN OUT NOCOPY BLOB,
    src_lob IN BFILE,
    amount IN INTEGER,
    dest_offset IN OUT INTEGER := 1,
    src_offset IN OUT INTEGER := 1,
    lang_context IN OUT INTEGER,
    warning OUT INTEGER);
```

Загружает BFILE *src_lob* во внутренний CLOB/NCLOB *dest_lob* с выполнением необходимого преобразования набора символов. Входное значение параметра *lang_context* определяет текущий контекст языка, а выходное – контекст языка на момент окончания загрузки. Параметр *warning* возвращает значение, если в процессе загрузки возникли проблемы. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_LOB.OPEN
    (lob_loc IN OUT NOCOPY {BLOB | CLOB CHARACTER SET ANY_CS | BFILE},
    open_mode IN BINARY_INTEGER);
```

Открывает LOB в режиме *open_mode*. Для файлов BFILE поддерживается только режим file_readonly. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_LOB.READ
    (lob_loc IN {BLOB | CLOB CHARACTER SET ANY_CS | BFILE},
    amount IN OUT [NOCOPY] BINARY_INTEGER,
    offset IN INTEGER,
    buffer OUT {RAW | CHARACTER SET lob_loc%CHARSET});
```

Копирует *amount* байт (BLOB) или символов (CLOB) из исходного большого объекта *lob_loc*, начиная с позиции *offset* байт или символов, в переменную *buffer*. Возвращает в *amount* количество реально скопированных байт или символов.

```
FUNCTION DBMS_LOB.SUBSTR
    (lob_loc IN {BLOB | CLOB CHARACTER SET ANY_CS | BFILE},
    amount IN INTEGER := 32767,
    offset IN INTEGER := 1)
    RETURN {RAW | VARCHAR2 CHARACTER SET lob_loc%CHARSET};
```

Аналогична встроенной функции SUBSTR. Возвращает *amount* байт (BLOB) или символов (CLOB), находящихся в большом объекте *lob_loc* начиная с позиции *offset* байт или символов.

```
PROCEDURE DBMS_LOB.TRIM
    (lob_loc IN OUT {BLOB | CLOB CHARACTER SET ANY_CS},
    newlen IN INTEGER);
```

Усекает большой объект *lob_loc* до длины *newlen* байт (BLOB) или символов (CLOB).

```
PROCEDURE DBMS_LOB.WRITE
    (lob_loc IN OUT {BLOB | CLOB CHARACTER SET ANY_CS},
    amount IN BINARY_INTEGER,
    offset IN INTEGER,
    buffer IN {RAW | VARCHAR2 CHARACTER SET lob_loc%CHARSET);
```

Копирует *amount* байт из переменной *buffer* в большой объект *lob_loc* начиная с позиции *offset* байт (BLOB) или символов (CLOB), перезаписывая любые уже имевшиеся данные в *lob_loc*.

```
PROCEDURE DBMS_LOB.WRITEAPPEND
    (lob_loc IN OUT NOCOPY {BLOB | CLOB CHARACTER SET ANY_CS},
    amount IN BINARY_INTEGER,
```

```
    buffer IN {RAW | VARCHAR2 CHARACTER SET lob_lob%CHARSET});
```

Дописывает *amount* байт или символов из переменной *buffer* в конец большого объекта *lob_loc*. Появилась в Oracle8*i*.

DBMS_LOCK

Позволяет применять сервисы управления блокировками Oracle в приложениях, нуждающихся в специализированных, не связанных с данными блокировках.

Вызовы

```
PROCEDURE DBMS_LOCK.ALLOCATE_UNIQUE
    (lockname IN VARCHAR2,
    lockhandle OUT VARCHAR2,
    expiration_secs IN INTEGER DEFAULT 864000);
```

Выделяет уникальный дескриптор блокировки *lockhandle* для блокировки, идентифицируемой по *lockname* на период *expiration_secs* секунд. Также выполняет команду COMMIT.

```
FUNCTION DBMS_LOCK.CONVERT
    ({id IN INTEGER | lockhandle IN VARCHAR2},
    lockmode IN INTEGER,
    timeout IN NUMBER DEFAULT MAXWAIT)
    RETURN INTEGER;
```

Переводит блокировку, идентифицируемую по *id* или *lockhandle*, в режим *lockmode*, ожидая успешного завершения в течение *timeout* секунд. При этом *lockmode* должна быть допустимой константой, определенной в пакете DBMS_LOCK. Возвращаются следующие значения: 0 – успех; 1 – превышение таймаута; 2 – взаимная блокировка; 3 – ошибка, связанная с параметрами; 4 – не владелец блокировки, преобразование невозможно; 5 – некорректный дескриптор блокировки.

```
FUNCTION DBMS_LOCK.RELEASE
    ({id IN INTEGER | lockhandle IN VARCHAR2})
    RETURN INTEGER;
```

Освобождает блокировку, идентифицируемую по *id* или *lockhandle*. Возвращает те же значения, что и функция CONVERT.

```
FUNCTION DBMS_LOCK.REQUEST
    ({id IN INTEGER | lockhandle IN VARCHAR2},
    lockmode IN INTEGER DEFAULT X_MODE,
    timeout IN INTEGER DEFAULT MAXWAIT,
    release_on_commit IN BOOLEAN DEFAULT FALSE)
    RETURN INTEGER;
```

Запрашивает блокировку, идентифицируемую по *id* или *lockhandle*, в режиме *lockmode*, ожидая успешного завершения в течение *timeout* секунд. Если значение параметра *release_on_commit* равно TRUE, то блокировка автоматически освобождается в результате фиксации (COMMIT) или отката (ROLLBACK) транзакции. При этом *lockmode* должна быть допустимой константой, определенной в пакете DBMS_LOCK. Возвращает те же значения, что и функция CONVERT.

```
PROCEDURE DBMS_LOCK.SLEEP
    (seconds IN NUMBER);
```

Приостанавливает сеанс на *seconds* секунд.

DBMS_LOGMNR

Содержит процедуры, предназначенные для инициализации LogMiner. Появилась в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_LOGMINER.ADD_LOGFILE
    (logfilename IN VARCHAR2,
    options IN BINARY_INTEGER DEFAULT ADDFILE);
```

Добавляет файл в существующий или вновь создаваемый список архивных файлов.

```
PROCEDURE DBMS_LOGMINER.START_LOGMNR
    (startSCN IN NUMBER DEFAULT 0,
    endSCN IN NUMBER DEFAULT 0,
    starttime IN DATE DEFAULT '01-JAN-1988',
    endtime IN DATE DEFAULT '01-JAN-2988',
    dict_file_name IN VARCHAR2 DEFAULT '',
    options IN BINARY_INTEGER DEFAULT 0);
```

Запускает сеанс LogMiner с указанием начального и конечного системного номеров изменений (SCN) или времени начала и завершения (если указаны SCN, то значения времени игнорируются). Файл *dict_file_name* – это плоский файл, содержащий моментальный снимок каталога БД. Информацию о параметре *options* можно найти в документации Oracle.

```
PROCEDURE DBMS_LOGMINER.END_LOGMNR;
```

Завершает сеанс LogMiner.

```
FUNCTION DBMS_LOGMINER.MINE_VALUE
    (sql_redo_undo IN RAW,
    column_name IN VARCHAR2 DEFAULT '')
    RETURN VARCHAR2;
```

Извлекает данные о столбце *column_name* из столбца *sql_redo_undo* в V$LOGMNR_CONTENTS и возвращает NULL или не-NULL. Появилась в Oracle9*i*.

```
FUNCTION DBMS_LOGMINER.COLUMN_PRESENT
    (sql_redo_undo IN RAW,
    column_name IN VARCHAR2 DEFAULT '')
    RETURN NUMBER;
```

Извлекает данные о столбце *column_name* из столбца *sql_redo_undo* в V$LOGMNR_CONTENTS и возвращает 0 в случае, если столбец не найден, или 1, если найден. Появилась в Oracle9*i*.

DBMS_LOGMNR_CDC_PUBLISH

Применяется для конфигурирования системы захвата изменений данных (Change Data Capture – CDC) и публикации содержащейся в ней информации. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_LOGMNR_CDC_PUBLISH.
    (owner IN VARCHAR2,
    change_table_name IN VARCHAR2,
    change_set_name IN VARCHAR2,
```

```
    source_schema IN VARCHAR2,
    source_table IN VARCHAR2,
    column_type_list IN VARCHAR2,
    capture_values IN VARCHAR2,
    rs_id IN CHAR,
    row_id IN CHAR,
    user_id IN CHAR,
    timestamp IN CHAR,
    object_id IN CHAR,
    source_colmap IN CHAR,
    target_colmap IN CHAR,
    options_string IN VARCHAR2);
```

Создает таблицу *change_table_name* в схеме *owner* для мониторинга таблицы *source_table* в схеме *source_schema*.

```
PROCEDURE DBMS_LOGMNR_CDC_PUBLISH.ALTER_CHANGE_TABLE
    (owner IN VARCHAR2,
    change_table_name IN VARCHAR2,
    operation IN VARCHAR2,
    column_list IN VARCHAR2,
    capture_values IN VARCHAR2,
    rs_id IN CHAR,
    row_id IN CHAR,
    user_id IN CHAR,
    timestamp IN CHAR,
    object_id IN CHAR,
    source_colmap IN CHAR,
    target_colmap IN CHAR);
```

Добавляет столбцы (значение параметра *operation* равно ADD) в таблицу *change_table_name* или удаляет из нее столбцы (значение параметра *operation* равно DROP).

```
PROCEDURE DBMS_LOGMNR_CDC_PUBLISH.DROP_SUBSCRIBER_VIEW
    (subscription_handle IN NUMBER,
    source_schema IN VARCHAR2,
    source_table IN VARCHAR2);
```

Удаляет представление *subscriber_handle* из схемы *source_schema*.

```
PROCEDURE DBMS_LOGMNR_CDC_PUBLISH.DROP_SUBSCRIPTION
    (subscription_handle IN NUMBER);
```

Удаляет подписку *subscription_handle*, полученную при вызове DBMS_LOGMNR_CDC_SUBSCRIBE.GET_SUBSCRIPTION_HANDLE.

```
PROCEDURE DBMS_LOGMNR_CDC_PUBLISH.DROP_CHANGE_TABLE
    (owner IN VARCHAR2,
    change_table_name IN VARCHAR2,
    force IN CHAR);
```

Удаляет таблицу *change_table_name* из схемы *owner*. Если значение параметра *force* равно Y, то таблица удаляется, даже если существуют подписки, ссылающиеся на данную таблицу; если же значение force равно N, то процедура не будет удалять таблицу, если на нее ссылаются какие-то подписки.

```
PROCEDURE DBMS_LOGMNR_CDC_PUBLISH.PURGE;
```

Удаляет ненужные строки из таблиц изменений.

DBMS_LOGMNR_CDC_SUBSCRIBE

Содержит процедуры, позволяющие издателю просматривать и запрашивать данные об изменениях, захваченных при помощи пакета DBMS_LOGMNR_CDC_PUBLISH. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_LOGMNR_CDC_SUBSCRIBE.GET_SUBSCRIPTION_HANDLE
    (change_set IN VARCHAR2,
    description IN VARCHAR2 := NULL,
    subscription_handle OUT NUMBER);
```

Создает дескриптор *subscription_handle*, сопоставленный набору изменений *change_set*.

```
PROCEDURE DBMS_LOGMNR_CDC_SUBSCRIBE.SUBSCRIBE
    (subscription_handle IN NUMBER,
    source_schema IN VARCHAR2,
    source_table IN VARCHAR2,
    column_list IN VARCHAR2);

PROCEDURE DBMS_LOGMNR_CDC_SUBSCRIBE.SUBSCRIBE
    (subscription_handle IN NUMBER,
    publication_id IN NUMBER,
    column_list IN VARCHAR2);
```

Определяет исходные таблицы и столбцы для подписчика, указывая их расположение или задавая *publication_id* для конкретной публикации.

```
PROCEDURE DBMS_LOGMNR_CDC_SUBSCRIBE.ACTIVATE_SUBSCRIPTION
    (subscription_handle IN NUMBER);
```

Приводит дескриптор *subscription_handle* в состояние готовности к приему данных об изменениях.

```
PROCEDURE DBMS_LOGMNR_CDC_SUBSCRIBE.EXTEND_WINDOW
    (subscription_handle IN NUMBER);
```

Устанавливает границы окна для *subscription_handle*, чтобы были видны новые данные изменений.

```
PROCEDURE DBMS_LOGMNR_CDC_SUBSCRIBE.PREPARE_SUBSCRIBER_VIEW
    (subscription_handle IN NUMBER,
    source_schema IN VARCHAR2,
    source_table IN VARCHAR2,
    view_name OUT VARCHAR2);
```

Создает в схеме подписчика представление *view_name* для получения данных об изменениях из таблицы *source_table*.

```
PROCEDURE DBMS_LOGMNR_CDC_SUBSCRIBE.DROP_SUBSCRIBER_VIEW
    (subscription_handle IN NUMBER,
    source_schema IN VARCHAR2,
    source_table IN VARCHAR2);
```

Удаляет представление подписчика из схемы подписчика.

```
PROCEDURE DBMS_LOGMNR_CDC_SUBSCRIBE.PURGE_WINDOW
    (subscription_handle IN NUMBER);
```

Возвращает в исходное положение нижнюю границу для *subscription_handle*, что делает окно пустым.

```
PROCEDURE DBMS_LOGMNR_CDC_SUBSCRIBE.DROP_SUBSCRIPTION
   (subscription_handle IN NUMBER);
```

Удаляет подписку *subscription_handle*.

DBMS_LOGMNR_D

Содержит процедуры, позволяющие работать с альтернативными табличными пространствами и словарем данных. Появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_LOGMNR_D.BUILD
    (dictionary_filename IN VARCHAR2,
    dictionary_location IN VARCHAR2
    [,options IN NUMBER]#);
```

Записывает информацию словаря данных в файл *dictionary_filename*, местоположение которого определяется параметром *dictionary_location*. Это может быть плоский файл (*options* = STORE_IN_FLAT_FILE) или журнальные файлы (*options* = STORE_IN_REDO_LOGS). Параметр *options* появился в Oracle9*i*.

```
PROCEDURE DBMS_LOGMNR_D.SET_TABLESPACE
   (tablespace IN VARCHAR2);
```

Заново создает таблицы LogMiner в табличном пространстве *tablespace*, отличном от табличного пространства по умолчанию для пользователя SYS. Появилась в Oracle9*i*.

DBMS_LOGSTDBY

В Oracle9*i* Release 2 появилась новая функциональность – логическое резервирование (*logical standby*), позволяющее использовать в качестве резервной базу данных, которая не является точной физической копией активной БД. Этот пакет содержит процедуры, позволяющие управлять средой логического резервирования.

Вызовы

```
PROCEDURE DBMS_LOGSTDBY.APPLY_SET
    (parameter IN VARCHAR2,
    value IN VARCHAR2);
```

Позволяет присвоить параметру *parameter* значение *value*. Подробная информация о параметрах, которые могут быть заданы, приведена в документации Oracle.

```
PROCEDURE DBMS_LOGSTDBY.APPLY_UNSET
    (parameter IN VARCHAR);
```

Отменяет значения, установленные процедурой APPLY_SET.

```
PROCEDURE DBMS_LOGSTDBY.BUILD;
```

Применяется для сохранения информации словаря LogMiner в журналах.

```
PROCEDURE DBMS_LOGSTDBY.GUARD_BYPASS_OFF;
```

Заново включает функцию защиты БД, которая была заблокирована при помощи процедуры GUARD_BYPASS_ON.

```
PROCEDURE DBMS_LOGSTDBY.GUARD_BYPASS_ON;
```

Отключает защиту БД, которая не допускает изменения логической резервной базы данных. Обычно применяется для обслуживания резервной БД или для исправления ошибок, но не для транзакций, т. к. триггеры и ограничения не используются.

```
PROCEDURE DBMS_LOGSTDBY.INSTANTIATE_TABLE
    (table_name IN VARCHAR2,
    schema IN VARCHAR2,
    dblink IN VARCHAR2);
```

Создает таблицу *table_name* в схеме *schema* базы данных *dblink*.

```
PROCEDURE DBMS_LOGSTDBY.SKIP
    (statement_option IN VARCHAR2,
    schema_name IN VARCHAR2,
    object_name IN VARCHAR2,
    proc_name IN VARCHAR2);
```

Задает фильтр, определяемый ключевыми словами в *statement_option*, для того чтобы предотвратить применение некоторых команд SQL объекта *object_name* схемы *schema_name* к логической резервной БД на основе правил, приведенных в *proc_name*.

```
PROCEDURE DBMS_LOGSTDBY.SKIP_ERROR
    (statement_option IN VARCHAR2,
    schema_name IN VARCHAR2,
    object_name IN VARCHAR2,
    proc_name IN VARCHAR2);
```

Вызывается, когда возникают ошибки сервисов применения журналов.

```
PROCEDURE DBMS_LOGSTDBY.SKIP_TRANSACTION
    (XIDUSN NUMBER STRING,
    XIDSLT NUMBER STRING,
    XIDSQN NUMBER STRING);
```

Вызывается для того, чтобы пропустить некоторую транзакцию, определяемую номером сегмента отката, номером слота и порядковым номером транзакции.

```
PROCEDURE DBMS_LOGSTDBY.UNSKIP
    (statement_option IN VARCHAR2,
    schema_name IN VARCHAR2,
    object_name IN VARCHAR2,
```

Отменяет действие процедуры SKIP.

```
PROCEDURE DBMS_LOGSTDBY.UNDO_ERROR
    (statement_option IN VARCHAR2,
    schema_name IN VARCHAR2,
    object_name IN VARCHAR2);
```

Отменяет действие процедуры SKIP_ERROR.

```
PROCEDURE DBMS_LOGSTDBY.UNSKIP_TRANSACTION
    (XIDUSN NUMBER STRING,
    XIDSLT NUMBER STRING,
    XIDSQN NUMBER STRING);
```

Отменяет действие процедуры SKIP_TRANSACTION.

DBMS_METADATA

Позволяет извлекать полные определения объектов БД из словаря данных. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_METADATA.OPEN
    (object_type IN VARCHAR2,
    version IN VARCHAR2 DEFAULT 'COMPATIBLE',
    model IN VARCHAR2 DEFAULT 'ORACLE')
    RETURN NUMBER;
```

Указывает *object_type*, *version* и *model* для объекта и возвращает указатель контекста, который будет использоваться в последующих вызовах.

```
PROCEDURE DBMS_METADATA.SET_FILTER
    (handle IN NUMBER,
    name IN VARCHAR2,
    value IN {VARCHAR2 | BOOLEAN DEFAULT TRUE});
```

Указывает ограничения для извлекаемых объектов. Параметр *name* означает имя фильтра (см. документацию Oracle).

```
PROCEDURE DBMS_METADATA.SET_COUNT
    (handle IN NUMBER,
    value IN NUMBER);
```

Указывает максимальное количество объектов (в параметре *value*), которые могут быть извлечены процедурой FETCH_*.

```
PROCEDURE DBMS_METADATA.GET_QUERY
    (handle IN NUMBER)
    RETURN VARCHAR2;
```

Возвращает текст запросов, используемых процедурой FETCH_*.

```
PROCEDURE DBMS_METADATA.SET_PARSE_ITEM
    (handle IN NUMBER,
    name IN VARCHAR2);
```

Задает атрибут вывода *name*, который синтаксически анализируется и возвращается функцией FETCH_DDL.

```
PROCEDURE DBMS_METADATA.ADD_TRANSFORMATION
    (handle IN NUMBER,
    name IN VARCHAR2,
    encoding IN VARCHAR2 DEFAULT NULL)
    RETURN NUMBER;
```

Указывает, что к XML-представлению извлекаемых объектов должна быть применена трансформация. Возвращает указатель на трансформированное представление.

```
PROCEDURE DBMS_METADATA.SET_TRANSFORM_PARAM
    (transform_handle IN NUMBER,
    name IN VARCHAR2,
    value IN {VARCHAR2 | BOOLEAN DEFAULT TRUE});
```

Указывает параметры для таблицы стилей XSLT, определяемой дескриптором *transform_handle*, позволяя настраивать результат трансформации.

```
FUNCTION DBMS_METADATA.FETCH_XML
    (handle IN NUMBER)
    RETURN SYS.XMLTYPE;
```

Возвращает метаданные в виде XML.

```
FUNCTION DBMS_METADATA.FETCH_DDL
    (handle IN NUMBER)
    RETURN SYS.KU$_DDLS;
```

Возвращает метаданные в виде DDL.

```
FUNCTION DBMS_METADATA.FETCH_CLOB
    (handle IN NUMBER)
    RETURN CLOB;

PROCEDURE DBMS_METADATA.FETCH_CLOB
    (handle IN NUMBER,
    doc IN OUT NOCOPY CLOB);
```

Возвращает сам объект в виде CLOB.

```
PROCEDURE DBMS_METADATA.CLOSE
    (handle IN NUMBER);
```

Закрывает дескриптор *handle* и обнуляет его состояние.

```
FUNCTION DBMS_METADATA.GET_XML
    (object_type IN VARCHAR2,
    name IN VARCHAR2,
    schema IN VARCHAR2 DEFAULT NULL,
    version IN VARCHAR2 DEFAULT 'COMPATIBLE',
    model IN VARCHAR2 DEFAULT 'ORACLE',
    transform IN VARCHAR2 DEFAULT NULL)
    RETURN CLOB;
```

Возвращает метаданные для объекта *name* схемы *schema* в виде XML.

```
FUNCTION DBMS_METADATA.GET_DDL
    (object_type IN VARCHAR2,
    name IN VARCHAR2,
    schema IN VARCHAR2 DEFAULT NULL,
    version IN VARCHAR2 DEFAULT 'COMPATIBLE',
    model IN VARCHAR2 DEFAULT 'ORACLE',
    transform IN VARCHAR2 DEFAULT 'DDL')
    RETURN CLOB;
```

Возвращает метаданные для объекта *name* схемы *schema* в виде DDL.

```
FUNCTION DBMS_METADATA.GET_DEPENDENT_XML
    (object_type IN VARCHAR2,
    base_object_name IN VARCHAR2,
    base_object_schema IN VARCHAR2 DEFAULT NULL,
    version IN VARCHAR2 DEFAULT 'COMPATIBLE',
    model IN VARCHAR2 DEFAULT 'ORACLE',
    transform IN VARCHAR2 DEFAULT NULL,
    object_count IN NUMBER DEFAULT 10000)
    RETURN CLOB;
```

Возвращает метаданные для зависимых объектов в виде XML.

```
FUNCTION DBMS_METADATA.GET_DEPENDENT_DDL
    (object_type IN VARCHAR2,
    base_object_name IN VARCHAR2,
```

```
    base_object_schema IN VARCHAR2 DEFAULT NULL,
    version IN VARCHAR2 DEFAULT 'COMPATIBLE',
    model IN VARCHAR2 DEFAULT 'ORACLE',
    transform IN VARCHAR2 DEFAULT 'DDL',
    object_count IN NUMBER DEFAULT 10000)
    RETURN CLOB;
```

Возвращает метаданные для зависимых объектов в виде DDL.

```
PROCEDURE DBMS_METADATA.GET_GRANTED_XML
    (object_type IN VARCHAR2,
    grantee IN VARCHAR2 DEFAULT NULL,
    version IN VARCHAR2 DEFAULT 'COMPATIBLE',
    model IN VARCHAR2 DEFAULT 'ORACLE',
    transform IN VARCHAR2 DEFAULT NULL,
    object_count IN NUMBER DEFAULT 10000)
    RETURN CLOB;
```

Возвращает метаданные для объектов, на которые предоставлены привилегии, в виде XML.

```
PROCEDURE DBMS_METADATA.GET_GRANTED_XML
    (object_type IN VARCHAR2,
    grantee IN VARCHAR2 DEFAULT NULL,
    version IN VARCHAR2 DEFAULT 'COMPATIBLE',
    model IN VARCHAR2 DEFAULT 'ORACLE',
    transform IN VARCHAR2 DEFAULT 'DDL',
    object_count IN NUMBER DEFAULT 10000)
    RETURN CLOB;
```

Возвращает метаданные для объектов, на которые предоставлены привилегии, в виде DDL.

DBMS_MGWADM

Предоставляет процедуры административного интерфейса для системы передачи сообщений Messaging Gateway, которая входит в состав компонента Oracle Advanced Queuing. Для того чтобы использовать данный пакет, необходимо запустить сценарий *catmgw.sql*. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_MGWADM.ALTER_AGENT
    (max_connections IN BINARY_INTEGER DEFAULT NULL,
    max_memory IN BINARY_INTEGER DEFAULT NULL);
```

Устанавливает ограничения на количество соединений и оперативной памяти, используемых агентом Messaging Gateway.

```
PROCEDURE DBMS_MGWADM.DB_CONNECT_INFO
    (username IN VARCHAR2,
    password IN VARCHAR2,
    database IN VARCHAR2 DEFAULT NULL);
```

Указывает параметры, используемые агентом Messaging Gateway для соединения с Oracle.

```
PROCEDURE DBMS_MGWADM.STARTUP
    (instance IN BINARY_INTEGER DEFAULT 0,
    force IN BINARY_INTEGER DEFAULT DBMS_MGWADM.NO_FORCE);
```

Запускает агент Messaging Gateway для экземпляра *instance*. Если значение *instance* равно 0, то агент может использовать любой экземпляр. Если параметр *force* имеет значение по умолчанию, то указанный экземпляр должен быть запущен.

```
PROCEDURE DBMS_MGWADM.SHUTDOWN
    (sdmode IN BINARY_INTEGER DEFAULT DBMS_MGWADM.SHUTDOWN_NORMAL);
```

Если параметр *sdmode* имеет значение по умолчанию, то агент Messaging Gateway пытается завершить текущую работу по тиражированию. Если же значение *sdmode* равно SHUTDOWN_IMMEDIATE, то агент завершает работу немедленно.

```
PROCEDURE DBMS_MGWADM.CLEANUP_GATEWAY
    (action IN BINARY_INTEGER);
```

Выполняет операции очистки или восстановления после нештатной остановки системы. Единственным допустимым значением параметра *action* является DBMS_MGWADM.CLEAN_STARTUP_STATE.

```
PROCEDURE DBMS_MGWADM.SET_LOG_LEVEL
    (log_level IN BINARY_INTEGER);
```

Устанавливает уровень журналирования *log_level* в значение DBMS_MGWADM.BASIC_LOGGING или DBMS_MGWADM.TRACE_DEBUG_LOGGING.

```
PROCEDURE DBMS_MGWADM.CREATE_MSGSYSTEM_LINK
   (linkname IN VARCHAR2,
    properties IN SYS.MGW_MQSERIES_PROPERTIES,
    options IN SYS.MGW_PROPERTIES DEFAULT NULL,
    comment IN VARCHAR2 DEFAULT NULL);
```

Создает канал связи с системой передачи сообщений MQSeries *linkname* с указанными свойствами.

```
PROCEDURE DBMS_MGWADM.ALTER_MSGSYSTEM_LINK
    (linkname IN VARCHAR2,
    properties IN SYS.MGW_MQSERIES_PROPERTIES,
    options IN SYS.MGW_PROPERTIES DEFAULT NULL,
    comment IN VARCHAR2 DEFAULT DBMS_MGWADM.NO_CHANGE);
```

Изменяет свойства канала связи с системой передачи сообщений MQSeries *linkname*.

```
PROCEDURE DBMS_MGWADM.REMOVE_MSGSYSTEM_LINK
    (linkname IN VARCHAR2);
```

Удаляет канал связи *linkname*.

```
PROCEDURE DBMS_MGWADM.REGISTER_FOREIGN_QUEUE
    (name IN VARCHAR2,
    linkname IN VARCHAR2,
    provider_queue IN VARCHAR2 DEFAULT NULL,
    domain IN INTEGER DEFAULT NULL,
    options IN SYS.MGW_PROPERTIES DEFAULT NULL,
    comment IN VARCHAR2 DEFAULT NULL);
```

Регистрирует очередь *name* для очереди *provider_queue* домена внешней системы с каналом связи *linkname*.

```
PROCEDURE DBMS_MGWADM.UNREGISTER_FOREIGN_QUEUE
    (name IN VARCHAR2,
    linkname IN VARCHAR2);
```

Удаляет очередь *name*, использующую канал связи *linkname*.

```
PROCEDURE DBMS_MGWADM.ADD_SUBSCRIBER
    (subscriber_id IN VARCHAR2,
    propagation_type IN BINARY_INTEGER,
    queue_name IN VARCHAR2,
    destination IN VARCHAR2,
    rule IN VARCHAR2 DEFAULT NULL,
    transformation IN VARCHAR2 DEFAULT NULL,
    exception_queue IN VARCHAR2 DEFAULT NULL);
```

Добавляет подписчика *subscriber_id*, получающего сообщения из очереди *queue_name*.

```
PROCEDURE DBMS_MGWADM.ALTER_SUBSCRIBER
    (subscriber_id IN VARCHAR2,
    rule IN VARCHAR2 DEFAULT NULL,
    transformation IN VARCHAR2 DEFAULT NULL,
    exception_queue IN VARCHAR2 DEFAULT NULL);
```

Изменяет параметры подписчика *subscriber_id*.

```
PROCEDURE DBMS_MGWADM.REMOVE_SUBSCRIBER
    (subscriber_id IN VARCHAR2,
    force IN BINARY_INTEGER DEFAULT DBMS_MGWADM.NO_FORCE);
```

Удаляет подписчика *subscriber_id*. Если параметр *force* имеет значение по умолчанию, то действие не удастся выполнить без корректной очистки; если же задано значение FORCE, то действие будет выполнено успешно даже в случае, если очистка невозможна.

```
PROCEDURE DBMS_MGWADM.RESET_SUBSCRIBER
    (subscriber_id IN VARCHAR);
```

Сбрасывает состояние ошибки тиражирования для *subscriber_id*.

```
PROCEDURE DBMS_MGWADM.SCHEDULE_PROPAGATION
    (schedule_id IN VARCHAR2,
    propagation_type IN BINARY_INTEGER,
    source IN VARCHAR2,
    destination IN VARCHAR2,
    start_time IN DATE DEFAULT SYSDATE,
    duration IN NUMBER DEFAULT NULL,
    next_time IN VARCHAR2 DEFAULT NULL,
    latency IN NUMBER DEFAULT 60);
```

Задает расписание *schedule_id* для тиражирования сообщения от источника *source* к получателю *destination*.

```
PROCEDURE DBMS_MGWADM.UNSCHEDULE_PROPAGATION
    (schedule_id IN VARCHAR2);
```

Удаляет расписание *schedule_id* из графика тиражирования.

```
PROCEDURE DBMS_MGWADM.ALTER_PROPAGATION_SCHEDULE
    (schedule_id IN VARCHAR2,
    duration IN NUMBER DEFAULT NULL,
    next_time IN VARCHAR2 DEFAULT NULL,
    latency IN NUMBER DEFAULT 60);
```

Изменяет расписание тиражирования сообщений *schedule_id*.

```
PROCEDURE DBMS_MGWADM.ENABLE_PROPAGATION_SCHEDULE
    (schedule_id IN VARCHAR2);
```

Активирует расписание тиражирования *schedule_id*.

```
PROCEDURE DBMS_MGWADM.DISABLE_PROPAGATION_SCHEDULE
   (schedule_id IN VARCHAR2);
```

Блокирует расписание тиражирования *schedule_id*.

DBMS_MGWMSG

Содержит процедуры для работы с типами сообщенийMessaging Gateway, а также с типами используемых объектов. Для того чтобы работать с данным пакетом, необходимо запустить сценарий *catmgw.sql*. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_MGWMSG.NVARRAY_ADD
    (p_array IN OUT SYS.MGW_NAME_ARRAY_T,
    p_value IN SYS.MGS_NAME_VALUE_T);
```

Добавляет элемент *p_value* в массив *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_GET
    (p_array IN SYS.MGW_NAME_ARRAY_T,
    p_name IN VARCHAR2,
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN SYS.MGW_NAME_VALUE_T;
```

Возвращает элемент массива *p_array* с именем *p_name*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_GET_BOOLEAN
    (p_array IN SYS.MGW_NAME_ARRAY_T,
    p_name IN VARCHAR2,
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN INTEGER;
```

Возвращает значение типа BOOLEAN_VALUE для элемента массива *p_array* с именем *p_name*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_GET_BYTE
    (p_array IN SYS.MGW_NAME_ARRAY_T,
    p_name IN VARCHAR2,
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN INTEGER;
```

Возвращает значение типа BYTE_VALUE для элемента массива *p_array* с именем *p_name*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_GET_SHORT
    (p_array IN SYS.MGW_NAME_ARRAY_T,
    p_name IN VARCHAR2,
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN INTEGER;
```

Возвращает значение типа SHORT_VALUE для элемента массива *p_array* с именем *p_name*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_GET_INTEGER
    (p_array IN SYS.MGW_NAME_ARRAY_T,
```

```
    p_name IN VARCHAR2,
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN INTEGER;
```

Возвращает значение типа INTEGER_VALUE для элемента массива *p_array* с именем *p_name*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_GET_LONG
    (p_array IN SYS.MGW_NAME_ARRAY_T,
    p_name IN VARCHAR2,
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN INTEGER;
```

Возвращает значение типа LONG_VALUE для элемента массива *p_array* с именем *p_name*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_GET_FLOAT
    (p_array IN SYS.MGW_NAME_ARRAY_T,
    p_name IN VARCHAR2,
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN INTEGER;
```

Возвращает значение типа FLOAT_VALUE для элемента массива *p_array* с именем *p_name*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_GET_DOUBLE
    (p_array IN SYS.MGW_NAME_ARRAY_T,
    p_name IN VARCHAR2,
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN INTEGER;
```

Возвращает значение типа DOUBLE_VALUE для элемента массива *p_array* с именем *p_name*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_GET_TEXT
    (p_array IN SYS.MGW_NAME_ARRAY_T,
    p_name IN VARCHAR2,
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN VARCHAR2;
```

Возвращает значение типа TEXT_VALUE для элемента массива *p_array* с именем *p_name*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_GET_RAW
    (p_array IN SYS.MGW_NAME_ARRAY_T,
    p_name IN VARCHAR2,
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN RAW;
```

Возвращает значение типа RAW_VALUE для элемента массива *p_array* с именем *p_name*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_GET_DATE
    (p_array IN SYS.MGW_NAME_ARRAY_T,
    p_name IN VARCHAR2,
```

```
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN DATE;
```

Возвращает значение типа DATE_VALUE для элемента массива *p_array* с именем *p_name*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_FIND_NAME
    (p_array IN SYS.MGW_NAME_ARRAY_T,
    p_name IN VARCHAR2,
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN BINARY_INTEGER;
```

Возвращает позицию элемента *p_name* в массиве *p_array*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

```
FUNCTION DBMS_MGWMSG.NVARRAY_FIND_NAME_TYPE
    (p_array IN SYS.MGW_NAME_ARRAY_T,
    p_name IN VARCHAR2,
    p_type IN BINARY_INTEGER,
    p_compare IN BINARY_INTEGER DEFAULT CASE_SENSITIVE)
    RETURN BINARY_INTEGER;
```

Возвращает позицию элемента *p_name* типа *p_type* в массиве *p_array*. Параметр *p_compare* принимает значение CASE_SENSITIVE или CASE_INSENSITIVE, определяющее способ сравнения *p_name* с именами из *p_array*.

DBMS_MVIEW

Содержит процедуры и функции, предназначенные для управления материализованными представлениями. До выхода версии Oracle9*i* пакет назывался DBMS_SNAPSHOT.

Вызовы

```
PROCEDURE DBMS_MVIEW.BEGIN_TABLE_REORGANIZATION
    (tabowner IN VARCHAR2,
    tabname IN VARCHAR2);
```

Сохраняет данные материализованного представления перед реорганизацией таблицы *tabname* в схеме *tabowner*.

```
PROCEDURE DBMS_MVIEW.END_TABLE_REORGANIZATION
    (tabowner IN VARCHAR2,
    tabname IN VARCHAR2);
```

Вызывается после реорганизации таблицы *tabname* в схеме *tabowner* для того, чтобы убедиться, что данные материализованного представления корректны и таблица *tabname* находится в соответствующем состоянии.

```
PROCEDURE DBMS_MVIEW.EXPLAIN_MVIEW
    (mv IN VARCHAR2,
    {statement_id IN VARCHAR2 := NULL |
    msg_array OUT SYS.ExplainMVArrayType});
```

Возвращает информацию о характеристиках материализованного представления *mv*. Если указывается параметр *statement_id*, то вывод попадает в таблицу MV_CAPABILITIES_TABLE, которая должна существовать. Предварительно необходимо запустить сценарий *utlxmv.sql*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_MVIEW.EXPLAIN_REWRITE
    (query IN VARCHAR2,
    mv IN VARCHAR2,
    {statement_id IN VARCHAR2 := NULL |
    msg_array OUT SYS.RewriteArrayType});
```

Возвращает информацию о перезаписи запроса при помощи материализованного представления *mv*. Если указывается параметр *statement_id*, то вывод попадает в таблицу REWRITE_TABLE, создаваемую при запуске сценария *utlxrw.sql*. Появилась в Oracle9*i*.

```
FUNCTION DBMS_MVIEW.I_AM_A_REFRESH()
    RETURN BOOLEAN;
```

Возвращает TRUE, если все локальные триггеры тиражирования для материализованных представлений отключены, и FALSE – в противном случае. Функция, которая сама себя идентифицирует – это хорошо!

```
FUNCTION DBMS_MVIEW.PMARKER
    (rid IN ROWID)
    RETURN NUMBER;
```

Возвращает маркер раздела для указанного *rid* типа ROWID, который применяется для Partition Change Tracking. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_MVIEW.PURGE_DIRECT_LOAD_LOG;
```

Удаляет записи из журнала прямого загрузчика. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_MVIEW.PURGE_LOG
    (master IN VARCHAR2,
    num IN BINARY_INTEGER := 1,
    flag IN VARCHAR2 := 'NOP');
```

Удаляет строки из журнала материализованного представления *master*. Параметр *num* определяет количество последних обновленных материализованных представлений. Если значение параметра *flag* равно 'DELETE', то строки будут гарантированно удалены как минимум для одного материализованного представления.

```
PROCEDURE DBMS_MVIEW.PURGE_{SNAPSHOT | MVIEW}_FROM_LOG
    ({snapshot_id | mview IN BINARY_INTEGER |
    {snapowner | mviewowner} IN VARCHAR2},
    {snapname | mviewname} IN VARCHAR2,
    {snapsite | mviewsite} IN VARCHAR2);
```

Удаляет строки из представлений словаря данных, связанных с обновлениями материализованных представлений. Материализованное представление определяется идентификатором *mview_id* или владельцем, именем и местоположением (узлом). Появилась в Oracle8*i*.

```
PROCEDURE DBMS_MVIEW.REFRESH
    ({list IN VARCHAR2 | tab IN OUT DBMS_UTILITY.UNCL_ARRAY},
    method IN VARCHAR2 := NULL,
    rollback_seg IN VARCHAR2 := NULL,
    push_deferred_rpc IN BOOLEAN := TRUE,
    refresh_after_errors IN BOOLEAN := FALSE,
    purge_option IN BINARY_INTEGER := 1,
    parallelism IN BINARY_INTEGER := 0,
    heap_size IN BINARY_INTEGER := 0,
    atomic_refresh IN BOOLEAN := TRUE);
```

Обновляет список материализованных представлений из списка *list* или *tab*. Параметр *atomic_refresh* появился в Oracle8*i*.

```
PROCEDURE DBMS_MVIEW.REFRESH_ALL_MVIEWS
    (number_of_failures OUT BINARY_INTEGER,
    method IN VARCHAR2 := NULL,
    rollback_seg IN VARCHAR2 := NULL,
    refresh_after_errors IN BOOLEAN := FALSE,
    atomic_refresh IN BOOLEAN := TRUE);
```

Обновляет все материализованные представления, не обновлявшиеся с момента последнего изменения в главной таблице. Все материализованные представления и все главные таблицы должны быть локальными и находиться в представлении DBA_MVIEWS. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_MVIEW.REFRESH_DEPENDENT
    ([number_of_failures OUT BINARY_INTEGER,]
    {list IN VARCHAR2 | tab IN OUT DBMS_UTILITY.UNCL_ARRAY},
    method IN VARCHAR2 := NULL,
    rollback_seg IN VARCHAR2 := NULL,
    refresh_after_errors IN BOOLEAN := FALSE,
    atomic_refresh IN BOOLEAN := TRUE);
```

Обновляет все материализованные представления, зависящие от главной таблицы или материализованного представления из списка *list* или *tab* и не обновлявшиеся с момента последнего изменения в главной таблице. Все материализованные представления и все главные таблицы должны быть локальными и находиться в представлении DBA_MVIEWS. Параметр *number_of_failures* не поддерживается в Oracle9*i*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_MVIEW.REGISTER_{SNAPSHOT | MVIEW}
    (snapowner | mviewowner} IN VARCHAR2,
    {snapname | mviewname} IN VARCHAR2,
    {snapsite | mviewsite} IN VARCHAR2,
    {snapid | mview_id} IN {DATE | BINARY_INTEGER},
    {flag IN BINARY_INTEGER,
    qry_txt IN VARCHAR2,
    rep_type IN BINARY_INTEGER := DBMS_{SNAPSHOT | MVIEW.REG_UNKNOWN};
```

Применяется для регистрации индивидуальных материализованных представлений.

```
PROCEDURE DBMS_SNAPSHOT.SET_I_AM_A_REFRESH
   (value IN BOOLEAN);
```

Устанавливает состояние пакета I_AM_A_REFRESH в значение *value*. Не поддерживается в версиях выше Oracle8.

```
PROCEDURE DBMS_MVIEW.UNREGISTER_{SNAPSHOT | MVIEW}
    ({snapowner | mviewowner} IN VARCHAR2,
    {snapname | mviewname} IN VARCHAR2,
    {snapsite | mviewsite} IN VARCHAR2);
```

Отменяет регистрацию индивидуальных материализованных представлений.

DBMS_OBFUSCATION_TOOLKIT

Содержит процедуры, позволяющие приложению шифровать данные согласно стандарту DES или Triple DES. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_OBFUSCATION_TOOLKIT.DESENCRYPT
    (input IN {RAW | VARCHAR2},
    key IN {RAW | VARCHAR2},
    encrypted_data OUT {RAW | VARCHAR2});
```

Шифрует данные *input* при помощи ключа *key* и возвращает их в *encrypted_data*.

```
PROCEDURE DBMS_OBFUSCATION_TOOLKIT.DESDECRYPT
    (input IN {RAW | VARCHAR2},
    key IN {RAW | VARCHAR2},
    decrypted_data OUT {RAW | VARCHAR2});
```

Расшифровывает данные *input* при помощи ключа *key* и возвращает их в *decrypted_data*.

```
PROCEDURE DBMS_OBFUSCATION_TOOLKIT.DES3ENCRYPT
    (input IN {RAW | VARCHAR2},
    key IN {RAW | VARCHAR2},
    encrypted_data OUT {RAW | VARCHAR2},
    which IN PLS_INTEGER);
```

Возвращает в *encrypted_data* данные, дважды или трижды зашифрованные с ключом *key*, при этом параметр *which* равен 0 (для режима TwoKeyMode – два ключа) или 1 (ThreeKeyMode – три ключа шифрования).

```
PROCEDURE DBMS_OBFUSCATION_TOOLKIT.DESDECRYPT
    (input IN {RAW | VARCHAR2},
    key IN {RAW | VARCHAR2},
    decrypted_data OUT {RAW | VARCHAR2});
```

Возвращает в *decrypted_data* данные, дважды или трижды расшифрованные с ключом *key*, при этом параметр *which* равен 0 (для режима TwoKeyMode – два ключа) или 1 (ThreeKeyMode – три ключа шифрования).

DBMS_ODCI

Возвращает используемую расширяемыми командами оптимизатора стоимость процессора для пользовательской функции, основанную на продолжительности ее выполнения. Появился в Oracle9*i*.

Вызов

```
FUNCTION DBMS_ODCI.ESTIMATE_CPU_UNITS
    (elapsed_time NUMBER)
    RETURN NUMBER;
```

Возвращает стоимость функции, рассчитанную на основе значения *elapsed_time* в секундах и быстродействия процессора данной машины.

DBMS_OFFLINE_OG

Содержит процедуры, служащие для управления автономным созданием *экземпляров главных групп* (*instantiations of master groups*), которые используются в среде с несколькими главными узлами.

Вызовы

```
PROCEDURE DBMS_OFFLINE_OG.BEGIN_INSTANTIATION
    (gname IN VARCHAR2,
```

```
    new_site IN VARCHAR2,
    fname IN VARCHAR2);
```

Запускает автономное создание экземпляров на узле *new_site* для группы тиражирования *gname*. Параметр *fname* предназначен только для внутреннего использования.

```
PROCEDURE DBMS_OFFLINE_OG.BEGIN_LOAD
    (gname IN VARCHAR2,
    new_site IN VARCHAR2);
```

Отключает триггеры на время импортирования данных на главный узел тиражирования *new_site*. Должна вызываться с главного узла тиражирования.

```
PROCEDURE DBMS_OFFLINE_OG.END_INSTANTIATION
    (gname IN VARCHAR2,
    new_site IN VARCHAR2);
```

Завершает создание экземпляров на узле *new_site*. Должна вызываться с главного узла тиражирования.

```
PROCEDURE DBMS_OFFLINE_OG.END_LOAD
    (gname IN VARCHAR2,
    new_site IN VARCHAR2,
    fname IN VARCHAR2);
```

Заново включает триггеры на узле *new_site* по завершении автономного создания экземпляров и загрузки. Параметр *fname* предназначен только для внутреннего использования.

```
PROCEDURE DBMS_OFFLINE_OG.RESUME_SUBSET_OF_MASTERS
    (gname IN VARCHAR2,
    new_site IN VARCHAR2,
    override IN BOOLEAN := FALSE);
```

Возобновляет тиражирование для существующих узлов в процессе тиражирования для узла *new_site*. Если значение параметра *override* (появился в Oracle8*i*) равно TRUE, то тиражирование на каждом главном узле восстанавливается как можно скорее; если же значение *override* равно FALSE, то тиражирование начинается лишь тогда, когда ни на одном главном узле не останется ожидающих исполнения запросов *gname*.

DBMS_OFFLINE_SNAPSHOT

Содержит процедуры для управления автономным созданием экземпляров материализованных представлений.

Вызовы

```
PROCEDURE DBMS_OFFLINE_SNAPSHOT.BEGIN_LOAD
    (gname IN VARCHAR2,
    sname IN VARCHAR2,
    master_site IN VARCHAR2,
    snapshot_oname IN VARCHAR2,
    storage_c IN VARCHAR2 := '',
    comment IN VARCHAR2 := '',
    min_communication IN BOOLEAN := TRUE);
```

Подготавливает материализованное представление в схеме *sname* для импорта нового материализованного представления *snapshot_oname*. Если значение пара-

метра *min_communications* равно TRUE, то обновление отправляет новое значение столбца, только если оно изменено командой UPDATE.

```
PROCEDURE DBMS_OFFLINE_SNAPSHOT.END_LOAD
    (gname IN VARCHAR2,
    sname IN VARCHAR2,
    snapshot_oname IN VARCHAR2);
```

Завершает автономное создание экземпляров *snapshot_oname*.

DBMS_OLAP

Содержит набор процедур и функций анализа материализованных представлений и Summary Advisor (консультант по сводным данным). Появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_OLAP.ADD_FILTER_ITEM
    (filter_id IN NUMBER,
    filter_name IN VARCHAR2,
    string_list IN VARCHAR2,
    number_min IN NUMBER,
    number_max IN NUMBER,
    date_min IN VARCHAR2,
    date_max IN VARCHAR2);
```

Добавляет новый элемент в существующий фильтр *filter_id*. Параметр *filter_name* описывает тип фильтра, а остальные параметры задаются для фильтров разных типов. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.CREATE_ID
    (id OUT NUMBER);
```

Создает уникальный идентификатор *id*, служащий для идентификации фильтра, рабочей нагрузки, результатов работы Summary Advisor или проверки корректности. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.ESTIMATE_{SUMMARY_SIZE | MVIEW_SIZE}
    (stmt_id IN VARCHAR2,
    select_clause IN VARCHAR2,
    num_rows OUT NUMBER,
    num_bytes OUT NUMBER);
```

Возвращает количество строк *num_rows* и байт *num_bytes*, которые должны содержаться в материализованном представлении, созданном запросом *select_clause*. ESTIMATE_SUMMARY_SIZE применяется в Oracle8*i*, а ESTIMATE_MVIEW_SIZE – в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.EVALUATE_UTILIZATION;
```

Измеряет частоту использования материализованных представлений. Выходные данные попадают в MVIEW$_EVALUATIONS. Только для Oracle8*i*.

```
PROCEDURE DBMS_OLAP.EVALUATE_UTILIZATION_W;
```

Измеряет частоту использования материализованных представлений на основе рабочей нагрузки схемы по умолчанию. Только для Oracle8*i*.

```
PROCEDURE DBMS_OLAP.EVALUATE_MVIEW_STRATEGY
    (run_id IN NUMBER,
    workload_id IN NUMBER,
    filter_id IN NUMBER);
```

Измеряет частоту использования материализованного представления по заданным нагрузке *workload_id* и фильтру *filter_id* и возвращает идентификатор для результатов запуска *run_id*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.GENERATE_MVIEW_REPORT
    (filename IN VARCHAR2,
    id IN NUMBER
    flags IN NUMBER);
```

Создает файл *filename* с отчетом о работе Summary Advisor (*id*). Параметр *flags* управляет представлением информации в отчете. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.GENERATE_MVIEW_SCRIPT
    (filename IN VARCHAR2,
    id IN NUMBER
    tspace IN VARCHAR2);
```

Генерирует в файле *filename* команды SQL, реализующие рекомендации Summary Advisor. Параметр *tspace* – это имя табличного пространства, используемого при создании материализованных представлений. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.LOAD_WORKLOAD_CACHE
    (workload_id IN NUMBER,
    flags IN NUMBER,
    filter_id IN NUMBER,
    application IN VARCHAR2,
    priority IN NUMBER);
```

Загружает нагрузку SQL-кэша *workload_id*. Параметр *flags* может иметь значение WORKLOAD_OVERWRITE, WORKLOAD_APPEND или WORKLOAD_NEW. Параметр *filter_id* указывает на ранее определенный фильтр; *application* – это бизнес-приложение по умолчанию. Параметр *priority* задает бизнес-приоритет для запросов при указанной нагрузке. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.LOAD_WORKLOAD_TRACE
    (collection_id IN NUMBER,
    flags IN NUMBER,
    filter_id IN NUMBER,
    application IN VARCHAR2,
    priority IN NUMBER,
    owner_name IN VARCHAR2);
```

Загружает нагрузку SQL Trace *collection_id*. Параметр *flags* может иметь значение WORKLOAD_OVERWRITE, WORKLOAD_APPEND или WORKLOAD_NEW. Параметр *filter_id* указывает на ранее определенный фильтр; *application* – это бизнес-приложение по умолчанию. Параметр *priority* задает бизнес-приоритет для запросов при указанной нагрузке. Параметр *owner_name* задает схему, которая содержит данные SQL Trace. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.LOAD_WORKLOAD_USER
    (workload_id IN NUMBER,
    flags IN NUMBER,
    filter_id IN NUMBER,
    owner_name IN VARCHAR2,
    table_name IN VARCHAR2);
```

Загружает пользовательскую нагрузку *workload_id*. Параметр *flags* может иметь значение WORKLOAD_OVERWRITE, WORKLOAD_APPEND или WORKLOAD_NEW. Параметр *filter_id* указывает на ранее определенный фильтр. Параметр

owner_name задает схему, которая содержит таблицу *table_name*, хранящую корректные данные о нагрузке. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.PURGE_FILTER
    (filter_id IN NUMBER);
```

Удаляет фильтр *filter_id*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.PURGE_RESULTS
    (run_id IN NUMBER);
```

Удаляет результаты рекомендаций, оценки или проверки корректности, идентифицируемых *run_id*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.PURGE_WORKLOAD
    (workload_id IN NUMBER);
```

Удаляет *workload_id*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.RECOMMEND_MV[_W]
    (fact_table_filter IN VARCHAR2,
    storage_in_bytes IN NUMBER,
    retention_list IN VARCHAR2,
    retention_pct IN NUMBER := 50);
```

Рекомендует, какие материализованные представления следует создать, сохранить или удалить. Параметр *fact_table_filter* содержит список имен таблиц фактов, разделенных запятыми; значение *storage_in_bytes* определяет максимальный объем, который может использоваться для хранения материализованных представлений. Параметр *retention_list* содержит список разделенных запятыми имен таблиц материализованных представлений, которые исключаются из листа удаления, а *retention_pct* – это объем (в процентном отношении) существующего хранилища материализованных представлений, который должен быть сохранен. RECOMMEND_MV_W основывается на рабочей нагрузке, собранной средствами Oracle Trace. Только для Oracle8*i*.

```
PROCEDURE DBMS_OLAP.RECOMMEND_MVIEW_STRATEGY
    (run_id IN NUMBER,
    workload_id IN NUMBER,
    filter_id IN NUMBER,
    storage_in_bytes IN NUMBER,
    retention_pct IN NUMBER,
    retention_list IN VARCHAR2,
    fact_table_filter IN VARCHAR2);
```

Формирует рекомендации относительно существующих и предполагаемых материализованных представлений. Перед запуском данной процедуры необходимо выполнить процедуру DBMS_STATS.GATHER_TABLE_STATS. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.SET_CANCELLED
    (run_id IN NUMBER);
```

Останавливает выполнение Summary Advisor с данным *run_id*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.VALIDATE_DIMENSION
    (dimension_name IN VARCHAR2,
    dimension_owner IN VARCHAR2,
    incremental IN BOOLEAN,
    check_nulls IN BOOLEAN,
    run_id IN NUMBER);
```

Проверяет, корректны ли указанные в измерении *dimension_name*, принадлежащем *dimension_owner*, отношения иерархии, атрибута и соединения. Параметры *incremental* и *check_nulls* относятся к проверке корректности, а *run_id* – это идентификатор, формируемый DBMS_OLAP.CREATE_ID.

```
PROCEDURE DBMS_OLAP.VALIDATE_WORKLOAD_CACHE
    (valid OUT NUMBER,
    error OUT VARCHAR2);
```

Проверяет корректность рабочей нагрузки SQL-кэша перед загрузкой и возвращает флаг успеха в *valid* или же ошибки – в *error*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.VALIDATE_WORKLOAD_TRACE
    (owner_name IN VARCHAR2,
    valid OUT NUMBER,
    error OUT VARCHAR2);
```

Проверяет корректность рабочей нагрузки SQL Trace перед загрузкой и возвращает флаг успеха в *valid* или же ошибки – в *error*. Параметр *owner_name* определяет владельца таблицы нагрузки SQL Trace. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_OLAP.VALIDATE_WORKLOAD_USER
    (owner_name IN VARCHAR2,
    table_name IN VARCHAR2,
    valid OUT NUMBER,
    error OUT VARCHAR2);
```

Проверяет корректность рабочей нагрузки, предоставленной пользователем, для таблицы *table_name* перед загрузкой и возвращает флаг успеха в *valid* или же ошибки – в *error*. Параметр *owner_name* определяет владельца таблицы нагрузки. Появилась в Oracle9*i*.

DBMS_ORACLE_TRACE_AGENT

Содержит процедуру, позволяющую собирать данные трассировки от другого сеанса. Обычно пакет доступен только пользователю SYS. Появился в Oracle8*i*.

Вызов

```
PROCEDURE DBMS_ORACLE_TRACE_AGENT.SET_ORACLE_TRACE_IN_SESSION
    (sid NUMBER DEFAULT 0,
    serial# NUMBER DEFAULT 0,
    on_off IN BOOLEAN DEFAULT FALSE,
    collection_name IN VARCHAR2 DEFAULT '',
    facility_name IN VARCHAR2 DEFAULT '');
```

Собирает данные SQL Trace для сессии *sid* с номером *serial#*. Параметр *on_off* включает и выключает трассировку, а параметры *collection_name* и *facility_name* определяют сущности SQL Trace.

DBMS_ORACLE_TRACE_USER

Обеспечивает пользователям доступ к инструментарию SQL Trace. Появился в Oracle8*i*.

Вызов

```
PROCEDURE DBMS_ORACLE_TRACE_USER.SET_ORACLE_TRACE
    (on_off IN BOOLEAN DEFAULT FALSE,
```

```
    collection_name IN VARCHAR2 DEFAULT '',
    facility_name IN VARCHAR2 DEFAULT '');
```

Параметр *on_off* включает и выключает трассировку, а параметры *collection_name* и *facility_name* определяют сущности SQL Trace.

DBMS_OUTLN

Обеспечивает возможность работы с хранимыми планами выполнения. В Oracle8*i* процедуры DROP_UNUSED, DROP_BY_CAT и UPDATE_BY_CAT входили в пакет OUTLN_PKG.

Вызовы

```
PROCEDURE DBMS_OUTLN.DROP_BY_CAT
    (cat VARCHAR2);
```

Удаляет планы выполнения, принадлежащие категории *cat*.

```
PROCEDURE DBMS_OUTLN.DROP_COLLISION;
```

Удаляет план выполнения, для которого значение OL$.HINTCOUNT не соответствует количеству подсказок для него в OL$HINTS.

```
PROCEDURE DBMS_OUTLN.DROP_EXTRA;
```

Удаляет лишние строки подсказок, не принимаемые во внимание в OL$.HINTCOUNT.

```
PROCEDURE DBMS_OUTLN.DROP_UNREFD_HINTS;
```

Удаляет строки подсказок, не имеющие соответствующего плана выполнения в таблице OL$.

```
PROCEDURE DBMS_OUTLN.DROP_UNUSED;
```

Удаляет планы выполнения, ни разу не примененные при компиляции команд SQL.

```
PROCEDURE DBMS_OUTLN.UPDATE_BY_CAT
    (oldcat VARCHAR2 DEFAULT 'DEFAULT',
    newcat VARCHAR2 DEFAULT 'DEFAULT');
```

Меняет категорию всех планов выполнения категории *oldcat* на *newcat*.

```
PROCEDURE DBMS_OUTLN.GENERATE_SIGNATURE
    (sqltxt IN VARCHAR2,
    signature OUT RAW);
```

Формирует сигнатуру для *sqltext*.

DBMS_OUTLN_EDIT

Применяется для редактирования хранимых планов выполнения. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_OUTLN_EDIT.CHANGE_JOIN_POS
    (name VARCHAR,
    hintno NUMBER,
    newpos NUMBER);
```

Изменяет позицию соединения в подсказке *hintno* для плана выполнения с именем *name* на *newpos*.

```
PROCEDURE DBMS_OUTLN_EDIT.CREATE_EDIT_TABLES;
```

Создает таблицы редактирования плана выполнения в схеме пользователя.

```
PROCEDURE DBMS_OUTLN_EDIT.DROP_EDIT_TABLES;
```

Удаляет таблицы редактирования плана выполнения в схеме пользователя.

```
PROCEDURE DBMS_OUTLN_EDIT.REFRESH_PRIVATE_OUTLINE
    (name IN VARCHAR2);
```

Обновляет копию плана выполнения *name* в оперативной памяти для синхронизации ее с изменениями, выполненными в подсказках.

DBMS_OUTPUT

Обеспечивает механизм вывода данных на выходное устройство сеанса из программы PL/SQL. Может применяться как простой отладчик или средство трассировки.

Вызовы

```
PROCEDURE DBMS_OUTPUT.DISABLE;
```

Запрещает вывод из пакета и очищает буфер DBMS_OUTPUT.

```
PROCEDURE DBMS_OUTPUT.ENABLE
    (buffer_size IN INTEGER DEFAULT 20000);
```

Разрешает вывод из пакета и устанавливает размер выходного буфера равным *buffer_size* байт.

```
PROCEDURE DBMS_OUTPUT.GET_LINE
    (line OUT VARCHAR2,
    status OUT INTEGER);
```

Получает следующую строку из буфера и помещает ее в *line*. Значение *status*, равное 0, свидетельствует об успешном извлечении; 1 означает ошибку.

```
PROCEDURE DBMS_OUTPUT.GET_LINES
    (lines OUT DBMS_OUTPUT.CHARARR,
    numlines IN OUT INTEGER);
```

Получает из буфера *numlines* строк и помещает их в массив *lines*.

```
PROCEDURE DBMS_OUTPUT.NEW_LINE;
```

Выводит в буфер DBMS_OUTPUT символ разделителя строк. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_OUTPUT.PUT
    (a IN DATE|NUMBER|VARCHAR2);
```

Помещает данные, содержащиеся в *a*, в буфер DBMS_OUTPUT, не добавляя символа разделителя строк.

```
PROCEDURE DBMS_OUTPUT.PUT_LINE
    (a IN DATE|NUMBER|VARCHAR2);
```

Помещает данные, содержащиеся в *a*, в буфер DBMS_OUTPUT, добавляя в конец символ разделителя строк.

DBMS_PCLXUTIL

Обеспечивает параллелизм внутри раздела при создании локальных индексов разделов. Появился в Oracle8*i*.

Вызов

```
PROCEDURE DBMS_PCLXUTIL.BUILD_PART_INDEX
    ([jobs_per_batch IN NUMBER DEFAULT 1,]
    procs_per_job IN NUMBER DEFAULT 1,
    tab_name IN VARCHAR2 DEFAULT NULL,
    idx_name IN VARCHAR2 DEFAULT NULL,
    force_opt IN BOOLEAN DEFAULT FALSE);
```

Создает локальный индекс *idx_name* для секционированной таблицы *tab_name*, основываясь на количестве *procs_per_job* параллельных запросов. Если значение параметра *force_opt* равно TRUE, то происходит перестроение всех секционированных индексов; в противном случае перестраиваются только разделы, помеченные как UNUSABLE. Параметр *jobs_per_batch* поддерживался только до выхода Oracle8*i*.

DBMS_PIPE

Обеспечивает обмен сообщениями между сеансами БД, используя структуры в оперативной памяти. Это взаимодействие асинхронное, нетранзакционное и может продолжаться после завершения сеанса.

Вызовы

```
FUNCTION DBMS_PIPE.CREATE_PIPE
    (pipename IN VARCHAR2,
    maxpipesize IN INTEGER DEFAULT 8192,
    private IN BOOLEAN DEFAULT TRUE)
    RETURN INTEGER;
```

Создает канал (*pipe*) с именем *pipename*, с максимальным размером *maxpipesize* и возвращает 0 в случае успеха. Если значение параметра *private* равно FALSE, то канал является общедоступным.

```
FUNCTION DBMS_PIPE.NEXT_ITEM_TYPE
    RETURN INTEGER;
```

Возвращает целое число, определяющее тип данных следующего элемента в буфере сообщений сеанса.

```
PROCEDURE DBMS_PIPE.PACK_MESSAGE
    (item IN {VARCHAR2 | NUMBER | DATE});
```

Упаковывает данные *item* в буфер сообщений сеанса.

```
PROCEDURE DBMS_PIPE.PACK_MESSAGE_RAW
    (item IN RAW);
```

Упаковывает данные типа RAW из *item* в буфер сообщений сеанса.

```
PROCEDURE DBMS_PIPE.PACK_MESSAGE_ROWID
    (item IN ROWID);
```

Упаковывает данные типа ROWID из *item* в буфер сообщений сеанса.

```
PROCEDURE DBMS_PIPE.PURGE
    (pipename IN VARCHAR2);
```

Удаляет все сообщения из канала *pipename*.

```
FUNCTION DBMS_PIPE.RECEIVE_MESSAGE
    (pipename IN VARCHAR2,
    timeout IN INTEGER DEFAULT MAXWAIT)
    RETURN INTEGER;
```

Получает сообщение из канала *pipename* в буфер сообщений сеанса, ожидая успешного завершения в течение *timeout* секунд. Возвращает 0 при успешном завершении и 1 в противном случае.

```
FUNCTION DBMS_PIPE.REMOVE_PIPE
    (pipename IN VARCHAR2)
    RETURN INTEGER;
```

Удаляет канал *pipename* и освобождает выделенную для него память, возвращая 0.

```
PROCEDURE DBMS_PIPE.RESET_BUFFER;
```

Сбрасывает индикаторы упаковки и распаковки для буфера сообщений сеанса, фактически удаляя все содержимое.

```
FUNCTION DBMS_PIPE.SEND_MESSAGE
    (pipename IN VARCHAR2,
    timeout IN INTEGER DEFAULT MAXWAIT,
    maxpipesize IN INTEGER DEFAULT 8192)
    RETURN INTEGER;
```

Отправляет содержимое буфера сообщений сеанса в канал *pipename*, ожидая успешного завершения в течение *timeout* секунд и (необязательно) устанавливая максимальный размер канала в *maxpipesize*. Возвращает 0 при успешном завершении и 1 в противном случае.

```
FUNCTION DBMS_PIPE.UNIQUE_SESSION_NAME
   RETURN VARCHAR2;
```

Возвращает уникальный строковый идентификатор сеанса длиной до 30 байт.

```
PROCEDURE DBMS_PIPE.UNPACK_MESSAGE
   (item OUT {VARCHAR2 | NUMBER | DATE});
```

Распаковывает очередной элемент данных из буфера сообщений в *item*.

```
PROCEDURE DBMS_PIPE.UNPACK_MESSAGE_RAW
   (item OUT RAW);
```

Распаковывает очередной элемент данных типа RAW из буфера сообщений в *item*.

```
PROCEDURE DBMS_PIPE.UNPACK_MESSAGE_ROWID
   (item OUT ROWID);
```

Распаковывает очередной элемент данных типа ROWID из буфера сообщений в *item*.

DBMS_PROFILER

Содержит процедуры и функции, предназначенные для профилирования приложений PL/SQL с целью выявления «узких мест» для производительности. Каждая функция возвращает код завершения; 0 означает успех, а положительные целые числа представляют собой коды ошибок. Появился в Oracle8*i*.

Вызовы

```
FUNCTION DBMS_PROFILER.START_PROFILER
    (run_comment IN VARCHAR2 := SYSDATE
    [,run_comment1 IN VARCHAR := ``]#
    [,run_number OUT BINARY_INTEGER]#)
    RETURN BINARY_INTEGER;
```

Запускает профилировщик с комментариями. *run_number* – это необязательный выходной параметр, равный номеру прогона и предназначенный для организации последующего обращения к информации. Два последних параметра появились в Oracle9*i*.

```
FUNCTION DBMS_PROFILER.STOP_PROFILER
    RETURN BINARY_INTEGER;
```

Останавливает работу профилировщика.

```
FUNCTION DBMS_PROFILER.FLUSH_DATA
    RETURN BINARY_INTEGER;
```

Сохраняет данные в заранее созданные таблицы БД.

```
FUNCTION DBMS_PROFILER.PAUSE_PROFILER
    RETURN BINARY_INTEGER;
```

Приостанавливает сбор данных профилировщиком. Появилась в Oracle9*i*.

```
FUNCTION DBMS_PROFILER.RESUME_PROFILER
    RETURN BINARY_INTEGER;
```

Возобновляет сбор данных профилировщиком. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_PROFILER.GET_VERSION
    (major OUT BINARY_INTEGER,
    minor OUT BINARY_INTEGER);
```

Возвращает версию пакета.

```
FUNCTION DBMS_PROFILER.INTERNAL_VERSION_CHECK
   RETURN BINARY_INTEGER;
```

Проверяет, может ли данная версия пакета работать с имеющейся версией БД.

DBMS_PROPAGATION_ADM

Предоставляет процедуры, позволяющие администрировать тиражирование из исходной очереди в очередь назначения. Применяется для Oracle Streams. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_PROPAGATION_ADM.ALTER_PROPAGATION
    (propagation_name IN VARCHAR2,
    rule_set_name IN VARCHAR2);
```

Изменяет, добавляет или удаляет набор правил *rule_set_name* из задания *propagation_name*.

```
PROCEDURE DBMS_PROPAGATION_ADM.CREATE_PROPAGATION
    (propagation_name IN VARCHAR2,
    source_queue IN VARCHAR2,
    destination_queue IN VARCHAR2,
```

```
    destination_dblink IN VARCHAR2 DEFAULT NULL,
    rule_set_name IN VARCHAR2 DEFAULT NULL);
```

Создает задание *propagation_name* на тиражирование из исходной очереди *source_queue* в *destination_queue* в БД *destination_dblink* с набором правил *rule_set_name*.

```
PROCEDURE DBMS_PROPAGATION_ADM.DROP_PROPAGATION
   (propagation_name IN VARCHAR2);
```

Удаляет задание *propagation_name* и все его сообщения.

DBMS_RANDOM

Содержит утилиту, генерирующую случайные числа. Появился в Oracle8.

Вызовы

```
PROCEDURE DBMS_RANDOM.INITIALIZE
   (seed IN BINARY_INTEGER);
```

Инициализирует генератор случайных чисел значением *seed*, которое должно быть как минимум пятиразрядным.

```
FUNCTION DBMS_RANDOM.RANDOM
   RETURN BINARY_INTEGER;
```

Возвращает случайное целое из генератора случайных чисел.

```
PROCEDURE DBMS_RANDOM.SEED
   (seed IN BINARY_INTEGER);
```

Изменяет начальное число для генерации случайных чисел на значение *seed*, которое должно быть как минимум пятиразрядным.

```
PROCEDURE DBMS_RANDOM.TERMINATE;
```

Освобождает ресурсы, занимаемые генератором случайных чисел, когда в них больше нет необходимости.

DBMS_RECTIFIER_DIFF

Содержит процедуры, которые выявляют и разрешают противоречия между данными на двух узлах тиражирования.

Вызовы

```
PROCEDURE DBMS_RECTIFIER_DIFF.DIFFERENCES
    (sname1 IN VARCHAR2,
    oname1 IN VARCHAR2,
    reference_site IN VARCHAR2 := '',
    sname2 IN VARCHAR2,
    oname2 IN VARCHAR2,
    comparison_site IN VARCHAR2 := '',
    where_clause IN VARCHAR2 := '',
    {column_list IN VARCHAR2 := '' |
      array_columns IN DBMS_UTILITY.NAME_ARRAY},
    missing_rows_sname IN VARCHAR2,
    missing_rows_oname1 IN VARCHAR2,
    missing_rows_oname2 IN VARCHAR2,
```

```
    missing_rows_site IN VARCHAR2:= '',
    max_missing IN INTEGER,
    commit_rows IN INTEGER := 500);
```

Сравнивает список столбцов *column_list* или *array_columns* таблиц *sname1.oname1* узла *reference_site* и *sname2.oname2* узла *comparison_site*. Вставляет вплоть до *max_missing* строк в таблицу *missing_rows_sname.missing_rows_oname1* и информацию об отсутствующих строках – в *missing_rows_oname2*, осуществляя фиксацию через каждые *commit_rows* строк.

```
PROCEDURE DBMS_RECTIFIER_DIFF.RECTIFY
    (sname1 IN VARCHAR2,
    oname1 IN VARCHAR2,
    reference_site IN VARCHAR2 := '',
    sname2 IN VARCHAR2,
    oname2 IN VARCHAR2,
    comparison_site IN VARCHAR2 := '',
    where_clause IN VARCHAR2 := '',
    {column_list IN VARCHAR2 := '' |
      array_columns IN DBMS_UTILITY.NAME_ARRAY},
    missing_rows_sname IN VARCHAR2,
    missing_rows_oname1 IN VARCHAR2,
    missing_rows_oname2 IN VARCHAR2,
    missing_rows_site IN VARCHAR2:= '',
    max_missing IN INTEGER,
    commit_rows IN INTEGER := 500);
```

Исправляет отличия между двумя таблицами, обнаруженные процедурой DIFFERENCES. Параметр *max_missing* появился в Oracle8*i*.

DBMS_REDEFINITION

Содержит процедуры оперативного переопределения таблиц. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_REDEFINITION.CAN_REDEF_TABLE
    (uname IN VARCHAR2,
    tname IN VARCHAR2,
    options_flag IN BINARY_INTEGER := 1);
```

Если таблицу *uname.tname* не удается переопределить с параметром *options_flag*, равным DBMS_REDEFINITION.CONS_USE_PK (переопределение с использованием первичного ключа по умолчанию) или DBMS_REDEFINITION.CONS_USE_ROWID (переопределение на основании значения ROWID), возвращается ошибка.

```
PROCEDURE DBMS_REDEFINITION.START_REDEF_TABLE
    (uname IN VARCHAR2,
    orig_table IN VARCHAR2,
    int_table IN VARCHAR2,
    col_mapping IN VARCHAR2:= NULL,
    options_flag IN BINARY_INTEGER := 1);
```

Запускает переопределение таблицы *uname.orig_table* в промежуточную таблицу *int_table* на основе списка соответствий столбцов *col_mapping*. Если *col_mapping* содержит NULL, то участвуют все столбцы *orig_table*. Параметр *options_flag* имеет то же значение, что и для CAN_REDEF_TABLE.

```
PROCEDURE DBMS_REDEFINITION.FINISH_REDEF_TABLE
    (uname IN VARCHAR,
    orig_table IN VARCHAR2,
    int_table IN VARCHAR2);
```

Завершает переопределение *uname.orig_table*. Перед этим вызовом можно создавать новые индексы, триггеры и ограничения для таблицы *int_table*.

```
PROCEDURE DBMS_REDEFINITION.SYNCH_INTERIM_TABLE
    (uname IN VARCHAR,
    orig_table IN VARCHAR2,
    int_table IN VARCHAR2);
```

Синхронизирует таблицу *int_table* с таблицей *uname.orig_table*. Запускаемая между долго выполняющимися операциями, эта процедура помогает сократить время выполнения процедуры FINISH_REDEF_TABLE.

```
PROCEDURE DBMS_REDEFINITION.ABORT_REDEF_TABLE
    (uname IN VARCHAR,
    orig_table IN VARCHAR2,
    int_table IN VARCHAR2);
```

Удаляет ошибки, возникшие в процессе переопределения.

DBMS_REFRESH

Позволяет создавать группы материализованных представлений, которые могут совместно обновляться на момент времени, соответствующий одной транзакции.

Вызовы

```
PROCEDURE DBMS_REFRESH.ADD
    (name IN VARCHAR2,
    {list IN VARCHAR2 | tab IN BMS_UTILITY.UNCL_ARRAY},
    lax IN BOOLEAN := FALSE);
```

Добавляет материализованные представления из списка *list* или индексной таблицы *tab* в группу *name*. Если материализованное представление перемещается из одной группы в другую, то значение параметра *lax* должно быть равно TRUE.

```
PROCEDURE DBMS_REFRESH.CHANGE
    (name IN VARCHAR2,
    next_date IN DATE := NULL,
    interval IN VARCHAR2 := NULL,
    implicit_destroy IN BOOLEAN := NULL,
    rollback_seg IN VARCHAR2 := NULL,
    push_deferred_rpc IN BOOLEAN := NULL,
    refresh_after_errors IN BOOLEAN := NULL,
    purge_option IN BINARY_INTEGER := NULL
    parallelism IN BINARY_INTEGER := NULL,
    heap_size IN BINARY_INTEGER := NULL);
```

Изменяет интервал обновления группы *name*, начиная с даты *next_date*. Остальные параметры изменяют другие характеристики группы обновления, как описано в процедуре MAKE.

```
PROCEDURE DBMS_REFRESH.DESTROY
    (name IN VARCHAR);
```

Исключает все материализованные представления из группы *name* и удаляет группу.

```
PROCEDURE DBMS_REFRESH.MAKE
    (name IN VARCHAR2,
    {list IN VARCHAR2 | tab IN DBMS_UTILITY.UNCL_ARRAY},
    next_date IN DATE := NULL,
    interval IN VARCHAR2 := NULL,
    implicit_destroy IN BOOLEAN := FALSE,
    lax IN BOOLEAN := NULL,
    job IN BINARY_INTEGER := 0,
    rollback_seg IN VARCHAR2 := NULL,
    push_deferred_rpc IN BOOLEAN := TRUE,
    refresh_after_errors IN BOOLEAN := FALSE,
    purge_option IN BINARY_INTEGER := NULL
    parallelism IN BINARY_INTEGER := NULL,
    heap_size IN BINARY_INTEGER := NULL);
```

Создает группу обновления *name* из списка *list* или *tab*. Параметр *interval* – это время между обновлениями, начиная с *next_date*. Значение TRUE параметра *implicit_destroy* приводит к удалению пустой группы *name*. Параметр *lax* применяется, если какое-то материализованное представление, перемещаемое в *name*, уже входит в какую-то группу. Параметр *job* необходим утилите Import и должен быть оставлен равным 0. Параметр *rollback_seg* – это имя сегмента отката, используемого при обновлении материализованных представлений. Параметр *push_deferred_rpc* продвигает изменения из обновляемого материализованного представления в главную таблицу или представление. Параметр *refresh_after_errors* позволяет обрабатывать данное обновление даже в случае наличия конфликтов. Параметр *purge_option* управляет очисткой очередей в случае применения параллельного тиражирования. Параметр *parallelism* указывает количество параллельных процессов, используемых для тиражирования, а *heap_size* указывает максимальное количество транзакций, рассматриваемых на предмет планирования параллельного тиражирования. Последний параметр не должен устанавливаться без специальных указаний от службы технической поддержки Oracle.

```
PROCEDURE DBMS_REFRESH.REFRESH
    (name IN VARCHAR2);
```

Инициирует обновление группы *name*.

```
PROCEDURE DBMS_REFRESH.SUBTRACT
    (name IN VARCHAR2,
    {list IN VARCHAR2 | tab IN DBMS_UTILITY.UNCL_ARRAY},
    lax IN BOOLEAN := FALSE);
```

Исключает элементы списка *list* или *tab* из группы *name*. Если значение параметра *lax* равно FALSE, то в случае, если материализованное представление не является членом группы *name*, возникает ошибка.

DBMS_REPAIR

Позволяет выявить и восстановить поврежденные блоки в таблицах и индексах. С исправляемыми объектами во время восстановления можно работать. Появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_REPAIR.ADMIN_TABLES
    (table_name IN VARCHAR2,
    table_type IN BINARY_INTEGER,
```

```
    action IN BINARY_INTEGER,
    tablespace IN VARCHAR2 DEFAULT NULL);
```

Управляет административной таблицей *table_name* типа ORPHAN_TABLE или REPAIR_TABLE (параметр *table_type*). Параметр *action* равен CREATE_ACTION для создания таблицы, PURGE_ACTION для удаления из таблицы строк несуществующих объектов или DROP_ACTION для удаления таблицы. Параметр *tablespace* – это табличное пространство, используемое при создании таблицы.

```
PROCEDURE DBMS_REPAIR.CHECK_OBJECT
    (schema_name IN VARCHAR2,
    object_name IN VARCHAR2,
    partition_name IN VARCHAR2 DEFAULT NULL
    object_type IN BINARY_INTEGER DEFAULT TABLE_OBJECT,
    repair_table_name IN VARCHAR2 DEFAULT 'REPAIR_TABLE',
    flags IN BINARY_OBJECT DEFAULT NULL,
    relative_fno IN BINARY_INTEGER DEFAULT NULL,
    block_start IN BINARY_INTEGER DEFAULT NULL,
    block_end IN BINARY_INTEGER DEFAULT NULL,
    corrupt_count OUT BINARY_INTEGER);
```

Проверяет объект *schema_name.object_name* в разделе *partition_name* (при его наличии) типа *object_type* (TABLE_TYPE или INDEX_TYPE). Добавляет строки в таблицу *repair_table*, которая должна существовать в схеме SYS. Параметр *relative_fno* – это относительный номер файла; а *block_start* и *block_end* указывают диапазон блоков. Параметр *repair_table_name* – это имя «ремонтной» таблицы, которая будет заполняться данными. Параметр *flags* зарезервирован для применения в будущем. Возвращает *corrupt_count* – количество обнаруженных повреждений.

```
PROCEDURE DBMS_REPAIR.DUMP_ORPHAN_KEYS
    (schema_name IN VARCHAR2,
    object_name IN VARCHAR2,
    partition_name IN VARCHAR2 DEFAULT NULL
    object_type IN BINARY_INTEGER DEFAULT INDEX_OBJECT,
    repair_table_name IN VARCHAR2 DEFAULT 'REPAIR_TABLE',
    orphan_table_name IN VARCHAR2 DEFAULT 'ORPHAN_KEYS_TABLE',
    flags IN BINARY_OBJECT DEFAULT NULL,
    key_count OUT BINARY_INTEGER);
```

Сообщает об индексных записях объекта *schema_name.object_name* в разделе *partition_name* (при его наличии) типа *object_type* (TABLE_TYPE или INDEX_TYPE), указывающих на поврежденные блоки данных, и вставляет строку в таблицу *orphan_table_name*. Таблица *repair_table_name* содержит информацию о поврежденных блоках в базовой таблице. Параметр *flags* зарезервирован для применения в будущем. Возвращает *key_count* – количество обработанных записей индекса.

```
PROCEDURE DBMS_REPAIR.FIX_CORRUPT_BLOCKS
    (schema_name IN VARCHAR2,
    object_name IN VARCHAR2,
    partition_name IN VARCHAR2 DEFAULT NULL
    object_type IN BINARY_INTEGER DEFAULT TABLE_OBJECT,
    repair_table_name IN VARCHAR2 DEFAULT 'REPAIR_TABLE',
    flags IN BINARY_OBJECT DEFAULT NULL,
    fix_count OUT BINARY_INTEGER);
```

Восстанавливает объект *schema_name.object_name* в разделе *partition_name* (при его наличии) типа *object_type* (TABLE_TYPE или INDEX_TYPE), о котором сообщила процедура CHECK_OBJECT. Таблица *repair_table_name* содержит информа-

цию о директивах восстановления. Параметр *flags* зарезервирован для использования в будущем. Возвращает *fix_count* — количество восстановленных блоков.

```
PROCEDURE DBMS_REPAIR.REBUILD_FREELISTS
    (schema_name IN VARCHAR2,
    partition_name IN VARCHAR2 DEFAULT NULL
    object_type IN BINARY_INTEGER DEFAULT TABLE_OBJECT);
```

Перестраивает списки свободных блоков для схемы *schema_name* в разделе *partition_name* (при его наличии).

```
PROCEDURE DBMS_REPAIR.SKIP_CORRUPT_BLOCKS
    (schema_name IN VARCHAR2,
    object_name IN VARCHAR2,
    object_type IN BINARY_INTEGER DEFAULT TABLE_OBJECT,
    flags IN BINARY_INTEGER DEFAULT SKIP_FLAG);
```

Если значение параметра *flags* равно SKIP_FLAG, поврежденные блоки пропускаются при сканировании индекса или таблицы *schema_name.object_name* (противоположное значение – NOSKIP_FLAG).

```
PROCEDURE DBMS_REPAIR.SEGMENT_FIX_STATUS
    (segment_owner IN VARCHAR2,
    segment_name IN VARCHAR2,
    segment_type IN BINARY_INTEGER DEFAULT TABLE_OBJECT,
    file_number IN BINARY_INTGER DEFAULT NULL,
    block_number IN BINARY_INTGER DEFAULT NULL,
    status_value IN BINARY_INTGER DEFAULT NULL,
    partition_name IN VARCHAR2 DEFAULT NULL);
```

Восстанавливает поврежденную битовую запись для сегмента *segment_owner.segment_name* в разделе *partition_name* (при его наличии). Вы можете указать *file_number* и *block_number* для уточнения местоположения блока данных и *status_value* для установки текущего состояния статуса блока. Появилась в Oracle9*i*.

DBMS_REPCAT

Содержит процедуры и функции для управления каталогом тиражирования и его окружением.

Вызовы

```
PROCEDURE DBMS_REPCAT.ADD_GROUPED_COLUMNS
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    column_group IN VARCHAR2,
    list_of_column_names IN {VARCHAR2 | DBMS_REPCAT.VARCHAR2});
```

Добавляет список названий столбцов *list_of_column_names* в существующую группу *column_group* для таблицы *sname.oname*.

```
PROCEDURE DBMS_REPCAT.ADD_MASTER_DATABASE
    (gname IN VARCHAR2,
    master IN VARCHAR2,
    use_existing_objects IN BOOLEAN := TRUE,
    copy_rows IN BOOLEAN := TRUE,
    comment IN VARCHAR2 := '',
    propagation_mode IN VARCHAR2 := 'ASYNCHRONOUS',
    fname IN VARCHAR2 := NULL);
```

Добавляет главный узел *master* в группу тиражирования *gname*. Параметр *use_existing_objects* позволяет указать, будут ли повторно использоваться объекты одного типа на существующем узле тиражирования. Параметр *copy_rows* показывает, что требуется скопировать содержимое таблицы на новый главный узел. Содержимое параметра *comment* добавляется в столбец MASTER_COMMENT представления DBA_REPSITES. Параметр *propagation_mode* может иметь значение ASYNCHRONOUS или SYNCHRONOUS. Параметр *fname* предназначен исключительно для внутреннего использования Oracle.

```
PROCEDURE DBMS_REPCAT.ADD_NEW_MASTERS
    (export_required IN BOOLEAN,
    {available_master_list IN VARCHAR2 |
    available_master_table DBMS_UTILITY.DBLINK_ARRAY},
    masterdef_flashback_scn OUT NUMBER,
    extension_id OUT RAW,
    break_trans_to_masterdef IN BOOLEAN := FALSE,
    break_trans_to_new_masters IN BOOLEAN := FALSE,
    percentage_for_catchup_mdef IN BINARY_INTEGER := 100,
    cycle_seconds_mdef IN BINARY_INTEGER := 60,
    percentage_for_catchup_new_ BINARY_INTEGER := 100,
    cycle_seconds_new_ BINARY_INTEGER := 60);
```

Добавляет новые главные узлы тиражирования, определенные в процедуре SPECIFY_NEW_MASTERS, в главные группы. Подробная информация приведена в документации Oracle. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.ADD_PRIORITY_datatype
    (gname IN VARCHAR2,
    pgroup IN VARCHAR2,
    value IN datatype,
    priority IN NUMBER);
```

Добавляет группу *gname* в группу приоритетов *pgroup* со значением *value*, которому сопоставлен приоритет *priority*. На месте *datatype* может стоять NUMBER, VARCHAR2, CHAR, DATE, RAW, NCHAR или NCHAR2.

```
PROCEDURE DBMS_REPCAT.ADD_SITE_PRIORITY_SITE
    (gname IN VARCHAR2,
    group IN VARCHAR2,
    site IN VARCHAR2,
    priority IN NUMBER);
```

Добавляет узел *site* в группу приоритетов сайтов *name* для главной группы *gname* с приоритетом *priority*.

```
PROCEDURE DBMS_REPCAT.ADD_UPDATE_RESOLUTION
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    column_group IN VARCHAR2,
    sequence_no IN NUMBER,
    method IN VARCHAR2,
    parameter_column_name IN {VARCHAR2 | DBMS_REPCAT.VARCHAR2 |
    DBMS_UTILITY.LNAME_ARRAY},
    priority_group IN VARCHAR2 := NULL,
    function_name IN VARCHAR2 := NULL,
    comment IN VARCHAR2 := NULL);
```

Назначает метод разрешения конфликтов обновлений для группы столбцов *column_group* в таблице *sname.oname*. Параметр *sequence_no* определяет последовательность, в которой будет применяться метод разрешения конфликта. Параметр

method определяет тип применяемого метода, это может быть одна из стандартных функций *function_names* или 'USER_FUNCTION'. Параметр *parameter_column_name* содержит список столбцов, участвующих в разрешении конфликта. Параметр *priority_group* указывает существующую группу приоритетов; *comment* – это пользовательский комментарий.

```
PROCEDURE DBMS_REPCAT.ADD_DELETE_RESOLUTION
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    sequence_no IN NUMBER,
    parameter_column_name IN {VARCHAR2 | DBMS_REPCAT.VARCHAR2s},
    function_name IN VARCHAR2,
    comment IN VARCHAR2 := NULL,
    method IN VARCHAR2 := 'USER FUNCTION');
```

Назначает метод разрешения конфликтов удалений для *sname.oname*. Параметр *sequence_no* определяет последовательность, в которой будет применяться метод разрешения конфликта. Параметр *parameter_column_name* содержит список столбцов, участвующих в разрешении конфликта. Параметр *method* (появился в Oracle8*i*) определяет тип применяемого метода, это может быть одна из стандартных функций *function_names* или 'USER_FUNCTION'. Параметр *priority_group* указывает существующую группу приоритетов; *comment* – это пользовательский комментарий.

```
PROCEDURE DBMS_REPCAT.ADD_UNIQUE_RESOLUTION
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    constraint IN VARCHAR2,
    sequence_no IN NUMBER,
    parameter_column_name IN {VARCHAR2 | DBMS_REPCAT.VARCHAR2s |
       DBMS_UTLITY.LNAME_ARRAY},
    function_name IN VARCHAR2,
    comment IN VARCHAR2 := NULL);
```

Назначает метод разрешения конфликтов уникальности для *sname.oname*, которые вызваны ограничением *constraint*. Параметр *sequence_no* определяет последовательность, в которой будет применяться метод разрешения конфликта. Параметр *parameter_column_name* содержит список столбцов, участвующих в разрешении конфликта. Параметр *method* определяет тип применяемого метода, это может быть одна из стандартных функций *function_names* или 'USER_FUNCTION'. Параметр *priority_group* указывает существующую группу приоритетов; *comment* – это пользовательский комментарий.

```
PROCEDURE DBMS_REPCAT.ALTER_CATCHUP_PARAMETERS
    (extension_id IN RAW,
    percentage_for_catchup_mdef IN BINARY_INTEGER := NULL,
    cycle_seconds_mdef IN BINARY_INTEGER := NULL,
    percentage_for_catchup_new IN BINARY_INTEGER := NULL,
    cyc_seconds_new IN BINARY_INTEGER := NULL);
```

Указанным образом изменяет параметры тиражирования. Параметр *extension_id* – это идентификатор текущего отложенного запроса для добавления главной БД без приостановки. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.ALTER_MASTER_PROPAGATION
    (gname IN VARCHAR2,
    master IN VARCHAR2,
    {dblink_list IN VARCHAR2 | dblink_table IN DBMS_UTILITY.DBLINK_ARRAY},
```

```
    [copy_rows IN BOOLEAN := TRUE,]#
    propagation_mode IN VARCHAR2 := 'ASYNCHRONOUS',
    comment IN VARCHAR2 := '');
```

Изменяет метод тиражирования. Если главный узел тиражирования присутствует в *dblink_list* или *dblink_table*, связь БД игнорируется. Остальные параметры описаны в процедуре ADD_MASTER_DATABASE. Параметр *copy_rows* появился в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.ALTER_MASTER_REPOBJECT
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    type IN VARCHAR2,
    ddl_text IN VARCHAR2,
    comment IN VARCHAR2 := '',
    retry IN BOOLEAN := FALSE
    [,safe_table_change IN BOOLEAN := FALSE]#);
```

Изменяет объект *sname.oname* типа *type* при помощи *ddl_text*. Если значение параметра *retry* равно TRUE, то объект изменяется только на тех главных узлах тиражирования, где его статус отличен от VALID. Параметр *safe_table_change* позволяет выполнять изменение без «замораживания» группы, содержащей таблицу; появился в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.ALTER_MVIEW_PROPAGATION
    (gname IN VARCHAR2,
    propagation_mode IN VARCHAR2,
    comment IN VARCHAR2 := '',
    gowner IN VARCHAR2 := 'PUBLIC');
```

Изменяет метод тиражирования для группы тиражирования *gname*, принадлежащей владельцу *gowner*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.ALTER_PRIORITY
    (gname IN VARCHAR2,
    pgroup IN VARCHAR2,
    old_priority IN NUMBER,
    new_priority IN NUMBER);
```

Изменяет приоритет *old_priority* на *new_priority* в группе приоритетов *pgroup* для главной группы тиражирования *gname*.

```
PROCEDURE DBMS_REPCAT.ALTER_PRIORITY_datatype
    (gname IN VARCHAR2,
    pgroup IN VARCHAR2,
    old_value IN datatype,
    new_value IN datatype);
```

Изменяет *old_value* на *new_value* в группе приоритетов *pgroup* для главной группы тиражирования *gname*. Тип данных столбца приоритета определяет, какое значение подставляется вместо datatype: NUMBER, VARCHAR2, CHAR, DATE, RAW, NCHAR или NCHAR2.

```
PROCEDURE DBMS_REPCAT.ALTER_SITE_PRIORITY
    (gname IN VARCHAR2,
    name IN VARCHAR2,
    old_priority IN NUMBER,
    new_priority IN NUMBER);
```

Изменяет приоритет *old_priority* на *new_priority* для узла *name* в главной группе тиражирования *gname*.

```
PROCEDURE DBMS_REPCAT.ALTER_SITE_PRIORITY_SITE
    (gname IN VARCHAR2,
    name IN VARCHAR2,
    old_site IN VARCHAR2,
    new_site IN VARCHAR2);
```

Заменяет имя *old_site* на *new_site* для группы приоритетов узлов *name* в главной группе тиражирования *gname*.

```
PROCEDURE DBMS_REPCAT.ALTER_SNAPSHOT_PROPAGATION
    (gname IN VARCHAR2,
    propagation_mode IN VARCHAR2,
    comment IN VARCHAR2 := '');
```

Изменяет режим тиражирования *propagation_mode* для группы *gname* с комментарием *comment*. Только для Oracle8*i*.

```
PROCEDURE DBMS_REPCAT.CANCEL_STATISTICS
    (sname IN VARCHAR2,
    oname IN VARCHAR2);
```

Останавливает сбор статистики конфликтов для таблицы *sname.oname*.

```
PROCEDURE DBMS_REPCAT.COMMENT_ON_COLUMN_GROUP
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    column_group IN VARCHAR2,
    comment IN VARCHAR2);
```

Добавляет комментарий *comment* для группы столбцов *column_group* таблицы *sname.oname*.

```
PROCEDURE DBMS_REPCAT.COMMENT_ON_MVIEW_REPSITES
    (gowner IN VARCHAR2,
    name IN VARCHAR2,
    comment IN VARCHAR2);
```

Обновляет комментарий для группы приоритетов узлов *name*, принадлежащей *gowner*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.COMMENT_ON_PRIORITY_GROUP
    (gname IN VARCHAR2,
    pgroup IN VARCHAR2,
    comment IN VARCHAR2);
```

Обновляет комментарий для группы приоритетов *pgroup* в главной группе тиражирования *gname*.

```
PROCEDURE DBMS_REPCAT.COMMENT_ON_REPGROUP
    (gname IN VARCHAR2,
    comment IN VARCHAR2);
```

Обновляет комментарий для группы тиражирования *gname*.

```
PROCEDURE DBMS_REPCAT.COMMENT_ON_REPOBJECT
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    type IN VARCHAR2,
    comment IN VARCHAR2);
```

Обновляет комментарий для типа *type* объекта *sname.oname*.

```
PROCEDURE DBMS_REPCAT.COMMENT_ON_REPSITES
    (gname IN VARCHAR2,
```

```
    [master IN VARCHAR2,]
    comment IN VARCHAR2);
```

Обновляет комментарий для узла или группы тиражирования. Если задан параметр *master*, то обновляется столбец MASTER_COMMENT представления DBA_REPSITES; в противном случае обновляется SCHEMA_COMMENT в представлении DBA_REPGROUP. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_REPCAT.COMMENT_ON_SITE_PRIORITY
    (gname IN VARCHAR2,
    name IN VARCHAR2,
    comment IN VARCHAR2);
```

Обновляет комментарий для группы приоритетов узлов *name* в главной группе тиражирования *gname*.

```
PROCEDURE DBMS_REPCAT.COMMENT_ON_UPDATE_RESOLUTION
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    column_group IN VARCHAR2,
    sequence_no IN NUMBER,
    comment IN VARCHAR2);
```

Обновляет комментарий *comment* для процедуры разрешения конфликтов, связанной с таблицей *sname.oname* и группой столбцов *column_group*. Параметр *sequence_no* определяет номер последовательности в процессе разрешения конфликта.

```
PROCEDURE DBMS_REPCAT.COMMENT_ON_UNIQUE_RESOLUTION
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    constraint_name IN VARCHAR2,
    sequence_no IN NUMBER,
    comment IN VARCHAR2);
```

Обновляет комментарий для приводящего к конфликту ограничения *constraint_name* таблицы *sname.oname*. Параметр *sequence_no* определяет номер последовательности в процессе разрешения конфликта.

```
PROCEDURE DBMS_REPCAT.COMMENT_ON_DELETE_RESOLUTION
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    sequence_no IN NUMBER,
    comment IN VARCHAR2);
```

Обновляет комментарий для таблицы *sname.oname* для разрешения конфликта удаления. Параметр *sequence_no* определяет номер последовательности в процессе разрешения конфликта.

```
PROCEDURE DBMS_REPCAT.COMPARE_OLD_VALUES
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    {column_list IN VARCHAR2 |
    column_table IN {DBMS_UTILITY.VARCHAR2s | DBMS_UTILITY.LNAME_ARRAY}},
    operation IN VARCHAR2 := 'UPDATE',
    compare IN BOOLEAN := TRUE);
```

Сравнивает тиражируемые значения столбцов из списка *column_list*/*column_table* в таблице *sname.oname* со старыми, если параметр *compare* равен TRUE для операции *operation*, значениями которой могут быть UPDATE, DELETE или * (групповой символ подразумевает обе названные операции). Появилась в Oracle8*i*.

```
PROCEDURE DBMS_REPCAT.CREATE_MASTER_REPGROUP
    (gname IN VARCHAR2,
    group_comment IN VARCHAR2 := '',
    master_comment IN VARCHAR2 := '',
    qualifier IN VARCHAR2 := '');
```

Создает новую главную группу тиражирования *gname* со спецификатором соединения *qualifier* и комментариями.

```
PROCEDURE DBMS_REPCAT.CREATE_MASTER_REPOBJECT
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    type IN VARCHAR2,
    use_existing_object IN BOOLEAN := TRUE,
    ddl_text IN VARCHAR2,
    comment IN VARCHAR2 := '',
    retry IN BOOLEAN := FALSE,
    copy_rows IN BOOLEAN := FALSE,
    gname IN VARCHAR2 := '');
```

Добавляет объект *sname.oname* типа *type* при помощи *ddl_text* в группу тиражирования *gname*. Параметр *use_existing_object* устанавливается в TRUE, если предполагается повторно использовать объекты одного типа и структуры. Если значение параметра *retry* равно TRUE, сервер Oracle попытается создать объект, который ранее создать не удалось. Если параметр *copy_rows* установлен в TRUE, то строки объекта будут первоначально скопированы в копию объекта.

```
PROCEDURE DBMS_REPCAT.CREATE_{SNAPSHOT|MVIEW}_REPOBJECT
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    type IN VARCHAR2,
    ddl_text IN VARCHAR2,
    comment IN VARCHAR2 := '',
    gname IN VARCHAR2 := '',
    gen_object_owner IN VARCHAR2 := '',
    min_communication IN BOOLEAN := TRUE,
    generate_80_compatible  IN BOOLEAN := TRUE,
    gowner IN VARCHAR2 := 'PUBLIC');
```

Создает копию объекта *sname.oname* типа *type* при помощи *ddl_text* (для объектов типа SNAPSHOT) в группе тиражирования *gname*. Параметры gname и *gowner* указывают имя и владельца для группы тиражирования материализованных представлений, в которую добавляется объект. Параметр *min_communication* должен быть равен FALSE при работе с версией Oracle 7.3. Параметр *generate_80_compatible* содержит TRUE в случае работы с версией Oracle, более ранней, чем Oracle8*i*. До выхода версии Oracle9*i* эта процедура называлась CREATE_SNAPSHOT_REPOBJECT.

```
PROCEDURE DBMS_REPCAT.CREATE_{SNAPSHOT|MVIEW}_REPGROUP
    (gname IN VARCHAR2,
    master IN VARCHAR2,
    comment IN VARCHAR2 := '',
    propagation_mode := 'ASYNCHRONOUS',
    fname IN VARCHAR2 := NULL,
    gowner IN VARCHAR2 := 'PUBLIC');
```

Создает группу тиражирования *gname* для БД *master*. Значение параметра *propagation_mode* равно 'ASYNCHRONOUS' или 'SYNCHRONOUS'. Параметр *fname* предназначен исключительно для внутреннего применения.

```
PROCEDURE DBMS_REPCAT.DEFINE_COLUMN_GROUP
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    column_group IN VARCHAR2,
    comment IN VARCHAR2 := NULL);
```

Определяет пустую группу столбцов *column_group* для объекта *sname.oname*, при необходимости – с комментарием *comment*.

```
PROCEDURE DBMS_REPCAT.DEFINE_PRIORITY_GROUP
    (gname IN VARCHAR2,
    pgroup IN VARCHAR2,
    datatype IN VARCHAR2,
    fixed_length IN INTEGER := NULL,
    comment IN VARCHAR2 := NULL);
```

Создает группу приоритетов *pgroup* для главной группы тиражирования *gname*, при этом элементы группы приоритетов имеют тип данных *datatype* (CHAR, VARCHAR2, NUMBER, DATE, RAW, NCHAR, NCHAR2). Элементы типа CHAR имеют длину *fixed_length*. Можно указать необязательный комментарий *comment*.

```
PROCEDURE DBMS_REPCAT.DEFINE_SITE_PRIORITY
    (gname IN VARCHAR2,
    name IN VARCHAR2,
    comment IN VARCHAR2 := NULL);
```

Создает группу приоритетов узлов *name* для главной группы тиражирования *gname* с необязательным комментарием *comment*.

```
PROCEDURE DBMS_REPCAT.DO_DEFERRED_REPCAT_ADMIN
    (gname IN VARCHAR2,
    all_sites IN BOOLEAN := FALSE);
```

Выполняет локальные незавершенные отложенные административные процедуры для главной группы тиражирования *gname*. Если значение параметра *all_sites* равно TRUE, то применяется задание для выполнения локальных административных процедур на каждом главном узле тиражирования.

```
PROCEDURE DBMS_REPCAT.DROP_COLUMN_GROUP
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    column_group IN VARCHAR2);
```

Удаляет группу столбцов *column_group* объекта *sname.oname*.

```
PROCEDURE DBMS_REPCAT.DROP_GROUPED_COLUMN
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    column_group IN VARCHAR2,
    list_of_column_names IN {VARCHAR2 | DBMS_REPCAT.VARCHAR2s});
```

Удаляет столбцы *list_of_column_names* из группы столбцов *column_group* объекта *sname.oname*.

```
PROCEDURE DBMS_REPCAT.DROP_MASTER_REPGROUP
    (gname IN VARCHAR2,
    drop_contents IN BOOLEAN := FALSE,
    all_sites IN BOOLEAN := FALSE);
```

Удаляет главную группу тиражирования *gname*. Если значение параметра *drop_contents* равно TRUE, то тиражированные данные удаляются из схемы. Если па-

раметр *all_sites* установлен в TRUE, то широковещательный вызов передается всем главным узлам.

```
PROCEDURE DBMS_REPCAT.DROP_MASTER_REPOBJECT
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    type IN VARCHAR2,
    drop_objects IN BOOLEAN := FALSE);
```

Удаляет объект *sname.oname* типа *type* из главной группы тиражирования. Если значение параметра *drop_objects* равно TRUE, то объект удаляется с каждого главного узла.

```
PROCEDURE DBMS_REPCAT.DROP_MVIEW_REPGROUP
    (gname IN VARCHAR2,
    drop_contents IN BOOLEAN := FALSE,
    gowner IN VARCHAR2 := 'PUBLIC');
```

Удаляет материализованное представление *gname*, принадлежащее владельцу *gowner*. Если значение параметра *drop_contents* равно TRUE, то все тиражируемые объекты удаляются из своих схем. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.DROP_MVIEW_REPOBJECT
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    type IN VARCHAR2,
    drop_objects IN BOOLEAN := FALSE);
```

Удаляет объект *sname.oname* типа *type* из материализованного представления. Если значение параметра *drop_objects* равно TRUE, то объект удаляется с каждого главного узла. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.DROP_PRIORITY
    (gname IN VARCHAR2,
    pgroup IN VARCHAR2,
    priority_num IN NUMBER);
```

Удаляет элементы с указанным приоритетом *priority_num* из группы приоритетов *pgroup* главной группы *gname*.

```
PROCEDURE DBMS_REPCAT.DROP_PRIORITY_GROUP
    (gname IN VARCHAR2,
    pgroup IN VARCHAR2);
```

Удаляет группу приоритетов *pgroup* главной группы *gname*.

```
PROCEDURE DBMS_REPCAT.DROP_PRIORITY_datatype
    (gname IN VARCHAR2,
    pgroup IN VARCHAR2,
    value IN datatype);
```

Удаляет из группы приоритетов *pgroup* главной группы *gname* элемент со значением *value*. Значением *datatype* может быть NUMBER, VARCHAR2, CHAR, DATE, RAW, NCHAR или NCHAR2.

```
PROCEDURE DBMS_REPCAT.DROP_SITE_PRIORITY
    (gname IN VARCHAR2,
    name IN VARCHAR2);
```

Удаляет группу приоритетов узлов *name* из главной группы тиражирования *gname*.

```
PROCEDURE DBMS_REPCAT.DROP_SITE_PRIORITY_SITE
    (gname IN VARCHAR2,
```

```
    name IN VARCHAR2,
    site IN VARCHAR2);
```

Удаляет узел *site* из группы приоритетов узлов *name* главной группы тиражирования *gname*.

```
PROCEDURE DBMS_REPCAT.DROP_SNAPSHOT_REPGROUP
    (gname IN VARCHR2,
    drop_contents IN BOOLEAN := FALSE,
    gowner IN VARCHAR2 := 'PUBLIC');
```

Удаляет материализованное представление *gname*, принадлежащее владельцу *gowner*. Если значение параметра *drop_contents* равно TRUE, то все тиражируемые объекты удаляется из своих схем. Не поддерживается в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.DROP_SNAPSHOT_REPOBJECTS
    (sname IN VARCHR2,
    oname IN VARCHAR2,
    type IN VARCHAR2,
    drop_objects IN BOOLEAN := FALSE);
```

Удаляет объект *sname.oname* типа *type* с текущего узла моментальной копии. Если значение параметра *drop_contents* равно TRUE, то тиражируемые объекты также удаляются. Только в Oracle8*i*.

```
PROCEDURE DBMS_REPCAT.DROP_UPDATE_RESOLUTION
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    column_group IN VARCHAR2,
    sequence_no IN NUMBER);
```

Удаляет команду разрешения конфликта обновления для группы столбцов *column_group* объекта *sname.oname*. Параметр *sequence_no* задает идентификатор метода разрешения конфликта.

```
PROCEDURE DBMS_REPCAT.DROP_DELETE_RESOLUTION
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    sequence_no IN NUMBER);
```

Удаляет команду разрешения конфликта удаления для объекта *sname.oname*. Параметр *sequence_no* задает идентификатор метода разрешения конфликта.

```
PROCEDURE DBMS_REPCAT.DROP_UNIQUE_RESOLUTION
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    constraint_name IN VARCHAR2,
    sequence_no IN NUMBER);
```

Удаляет команду разрешения конфликта ограничения *constraint_name* для объекта *sname.oname*. Параметр *sequence_no* задает идентификатор метода разрешения конфликта.

```
PROCEDURE DBMS_REPCAT.EXECUTE_DDL
    (gname IN VARCHAR2,
    {master_list IN VARCHAR2 := NULL |
    master_table IN DBMS_UTILITY.DBLINK.ARRAY},
    ddl_text IN VARCHAR2);
```

Выполняет *ddl_text* для главной группы *gname*. Параметр *master_list*/*master_table* содержит список главных узлов, для которых должен быть выполнен *ddl_text*.

```
PROCEDURE DBMS_REPCAT.GENERATE_MVIEW_SUPPORT
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    type IN VARCHAR2,
    gen_object_owner IN VARCHAR2 := '',
    min_communication IN BOOLEAN := TRUE
    [,generate_80_compatible  IN BOOLEAN := TRUE)];
```

Формирует триггеры и пакеты в схеме *gen_object_owner* (или *sname*, если *gen_object_owner* содержит NULL), необходимые для поддержки материализованного представления объекта *sname.oname* типа *type*. Параметр *min_communication* должен быть равен FALSE при работе с версией Oracle 7.3. Параметр *generate_80_compatible* должен быть равен TRUE в случае работы в версии Oracle, более ранней, чем Oracle8*i*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.GENERATE_REPLICATION_PACKAGE
    (sname IN VARCHAR2,
    oname IN VARCHAR2);
```

Формирует пакет тиражирования для объекта *sname.oname*. Не доступна в версиях выше Oracle8.

```
PROCEDURE DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    type IN VARCHAR2,
    package_prefix IN VARCHAR2 := NULL,
    procedure_prefix IN VARCHAR2 := NULL,
    distributed IN BOOLEAN := TRUE,
    gen_object_owner IN VARCHAR2 := '',
    min_communication IN BOOLEAN := TRUE
    [, generate_80_compatible  IN BOOLEAN := TRUE]);
```

Формирует триггеры и пакеты, которые необходимы для поддержки объекта *sname.oname* типа *type*. Параметры *package_prefix* и *procedure_prefix* управляют формированием кода. Параметр *distributed* следует оставить в значении TRUE. Параметр *min_communication* должен быть равен FALSE в случае работы с версией Oracle 7.3. Параметр *generate_80_compatible* должен быть равен TRUE для версии Oracle, более ранней, чем Oracle8*i*.

```
PROCEDURE DBMS_REPCAT.GENERATE_REPLICATION_TRIGGER
    ({sname IN VARCHAR2,
    oname IN VARCHAR2 |
    gname IN VARCHAR2},
    gen_objs_owner IN VARCHAR2 := NULL,
    min_communication IN BOOLEAN := TRUE);
```

Формирует пакет тиражирования для объекта *sname.oname* или группы *gname*. Параметр *gen_objs_owner* должен быть установлен в TRUE для совместимости с версиями ниже Oracle 7.3. Параметр *min_communication* должен содержать FALSE, если хоть один из главных узлов работает с версией Oracle 7.3. Недоступна в версиях выше Oracle8.

```
PROCEDURE DBMS_REPCAT.GENERATE_SNAPSHOT_SUPPORT
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    type IN VARCHAR2,
    gen_object_owner IN VARCHAR2 := '',
    min_communication IN BOOLEAN := TRUE
    [, generate_80_compatible  IN BOOLEAN := TRUE]);
```

Формирует триггеры и пакеты, которые необходимы для поддержки моментальной копии объекта *sname.oname* типа *type*. Параметр *min_communication* должен быть равен FALSE в случае работы с версией Oracle 7.3. Параметр *generate_80_compatible* (появился в Oracle8*i*) должен быть равен TRUE для версии Oracle, более ранней, чем Oracle8*i*.

```
PROCEDURE DBMS_REPCAT.MAKE_COLUMN_GROUP
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    column_group IN VARCHAR2,
    list_of_column_names IN {VARCHAR2 | DBMS_REPCAT.VARCHAR2s});
```

Создает группу столбцов *column_group* с именами *list_of_column_names* для объекта *sname.oname*.

```
PROCEDURE DBMS_REPCAT.PREPARE_INSTANTIATED_MASTER
    (extension_id IN RAW);
```

Разрешает перенос отложенной заявки *extension_id* для добавления баз данных в главную группу узлов без приостановки. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.PURGE_MASTER_LOG
    (id IN BINARY_INTEGER,
    source IN VARCHAR2,
    gname IN VARCHAR2);
```

Удаляет идентификатор заявки *id* с узла *source* группы *gname*.

```
PROCEDURE DBMS_REPCAT.PURGE_STATISTICS
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    start_date IN DATE,
    end_date IN DATE);
```

Удаляет статистику для объекта *sname.oname* за период от *start_date* до *end_date*. Только для Oracle8*i*.

```
PROCEDURE DBMS_REPCAT.REFRESH_{SNAPSHOT | MVIEW}_REPGROUP
    (gname IN VARCHAR2,
    drop_missing_contents IN BOOLEAN := FALSE,
    refresh_mviews IN BOOLEAN := FALSE,
    refresh_other_objects  IN BOOLEAN := FALSE,
    gowner IN VARCHAR2 := 'PUBLIC');
```

Обновляет группу материализованных представлений *gname*, принадлежащую *gowner*. Параметр *drop_missing_contents* управляет удалением объектов, предварительно удаленных из группы тиражирования, из схемы материализованных представлений. Параметр *refresh_mviews* обновляет содержимое материализованных представлений, параметр *refresh_other_objects* обновляет другие объекты в *gname*. До выхода версии Oracle9*i* процедура называлась REFRESH_SNAPSHOT_REPGROUP.

```
 PROCEDURE DBMS_REPCAT.REGISTER_{SNAPSHOT | MVIEW}_REPGROUP
    (gname IN VARCHAR2,
    {snapsite|mviewsite} IN VARCHAR2,
    [comment IN VARCHAR2 := NULL,]
    rep_type IN NUMBER := REG_UNKNOWN,
    fname IN VARCHAR2 := NULL,
    gowner IN VARCHAR2 := 'PUBLIC');
```

Регистрирует группу материализованных представлений *gname* с глобальным именем *mviewsite*, принадлежащую *gowner*. Параметр *comment* является необязательным, параметр *rep_type* определяет версию группы материализованных представлений. Параметр *fname* предназначен только для внутреннего использования в версии Oracle8*i* и выше. До выхода версии Oracle9*i* процедура называлась REGISTER_SNAPSHOT_REPGROUP.

```
PROCEDURE DBMS_REPCAT.REGISTER_STATISTICS
    (sname IN VARCHAR2,
    oname IN VARCHAR2);
```

Регистрирует статистику разрешения конфликтов для объекта *sname.oname*.

```
PROCEDURE DBMS_REPCAT.RELOCATE_MASTERDEF
    (gname IN VARCHAR2,
    old_masterdef IN VARCHAR2,
    new_masterdef IN VARCHAR2,
    notify_masters IN BOOLEAN := TRUE,
    include_old_masterdef  IN BOOLEAN := TRUE,
    require_flavor_change IN BOOLEAN := FALSE);
```

Перемещает группу тиражирования *gname* с узла *old_masterdef* на узел *new_masterdef*. Если значение параметра *notify_masters* равно TRUE, то изменения передаются всем главным узлам тиражирования. Если и параметр *include_old_masterdef* установлен в TRUE, то узел *old_masterdef* также оповещается об изменениях. Параметр *require_flavor_change* предназначен для внутреннего использования сервером Oracle в версиях Oracle8*i* выше.

```
PROCEDURE DBMS_REPCAT.REMOVE_MASTER_DATABASES
    (gname IN VARCHAR2,
    {master_list IN VARCHAR2 | master_table IN DBMS_UTILITY.DBLINK_ARRAY});
```

Удаляет главные базы данных, заданные во втором параметре, из среды тиражирования, соответствующей группе тиражирования *gname*.

```
PROCEDURE DBMS_REPCAT.RENAME_SHADOW_COLUMN_GROUP
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    new_col_group_name IN VARCHAR2);
```

Помещает столбцы теневой группы объекта *sname.oname* в группу *new_col_group_name*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.IMPORT_CHECK
    (gname IN VARCHAR2,
    master IN BOOLEAN,
    gowner IN VARCHR2 := 'PUBLIC');
```

Проверяет, имеют ли объекты главной группы соответствующие идентификаторы и значения статусов после импорта/экспорта тиражированного объекта. Если нет значения для главной группы *gname* и владельца *gowner*, то проверяются все главные группы текущего узла. Параметр *master* означает, что проверка будет выполняться для главного узла.

```
PROCEDURE DBMS_REPCAT.RESUME_MASTER_ACTIVITY
    (gname IN VARCHAR2,
    override IN BOOLEAN := FALSE);
```

Восстанавливает тиражирование для группы *gname*. Если значение параметра *override* равно TRUE, то «подвешенные» административные заявки игнорируются, активность восстанавливается как можно скорее.

```
PROCEDURE DBMS_REPCAT.RESUME_PROPAGATION_TO_MDEF
    (extension_id IN RAW);
```

Завершает добавление нового главного узла в главную группу без приостановки. Параметр *extension_id* идентифицирует «подвешенную» заявку на добавление. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.SEND_[AND_COMPARE_]OLD_VALUES
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    {column_list IN VARCHAR2 |
    column_table IN {DBMS_UTILITY.VARCHAR2s | DBMS_UTILITY.LNAME_ARRAY}},
    operation IN VARCHAR2 := 'UPDATE',
    send IN BOOLEAN := 'TRUE');
```

Если параметр *send* установлен в TRUE, то процедура отсылает старые значения столбцов в процессе переноса отложенных транзакций для столбцов, не являющихся ключевыми, из *column_list*/*column_table* для объекта *sname.oname* для операции *operation*, значениями которой могут быть UPDATE, DELETE или * (групповой символ подразумевает обе названные операции). В версии Oracle8 эта процедура называлась SEND_AND_COMPARE_OLD_VALUES.

```
PROCEDURE DBMS_REPCAT.SET_COLUMNS
    (sname IN VARCHAR2,
    oname IN VARCHAR2,
    {column_list IN VARCHAR2 |
    column_table IN {DBMS_UTILITY.VARCHAR2s | DBMS_UTILITY.LNAME_ARRAY}});
```

Позволяет использовать столбцы из параметра *column_list*/*column_table* для сравнения вместо первичного ключа при тиражировании объекта *sname.oname*.

```
PROCEDURE DBMS_REPCAT.SPECIFY_NEW_MASTERS
    (gname IN VARCHAR2,
    {master_list IN VARCHAR2, master_table IN DBMS_UTILITY.DBLINK_ARRAY});
```

Добавляет новые главные узлы, перечисленные во втором параметре, в главную группу *gname*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY
    (gname IN VARCHAR2);
```

Приостанавливает активность главной группы *gname*.

```
PROCEDURE DBMS_REPCAT.SWITCH_{SNAPSHOT | MVIEW}_MASTER
    (gname N VARCHAR2,
    master IN VARCHAR2,
    gowner IN VARCHAR2 := 'PUBLIC']);
```

Перенаправляет группу материализованных представлений *gname*, принадлежащую *gowner*, на главный узел *master*. До версии Oracle9*i* процедура называлась SWITCH_SNAPSHOT_MASTER.

```
PROCEDURE DBMS_REPCAT.UNDO_ADD_NEW_MASTERS_REQUEST
    (extension_id IN RAW,
    drop_contents IN BOOLEAN := TRUE);
```

Отменяет все изменения, выполненные процедурами SPECIFY_NEW_MASTERS и ADD_NEW_MASTERS для заявки *extension_id*. Если параметр *drop_contents* установлен в TRUE, то содержимое объектов в новых группах тиражирования также удаляется. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT.UNREGISTER_{SNAPSHOT | MVIEW}_REPGROUP
    (gname IN VARCHAR2,
```

```
    {snapsite|mviewsite} IN VARCHAR2,
    gowner IN VARCHAR := 'PUBLIC']);
```

Удаляет регистрацию группы *gname*, принадлежащей владельцу *gowner*, с узла *mviewsite*. До версии Oracle9*i* процедура называлась UNREGISTER_SNAPSHOT_REPGROUP.

```
FUNCTION DBMS_REPCAT.VALIDATE
    (gname IN VARCHAR2,
    check_genflags IN BOOLEAN := FALSE,
    check_valid_objs IN BOOLEAN := FALSE,
    check_links_sched IN BOOLEAN := FALSE,
    check_links IN BOOLEAN := FALSE,
    {error_table OUT DBMS_REPCAT.VALIDATE_ERR_TABLE |
    error_msg_table OUT DBMS_UTILITY.UNCL_ARRAY,
    error_num_table OUT DBMS_UTILITY.NUMBER_ARRAY})
    RETURN BINARY_INTEGER;
```

Проверяет корректность условий, установленных в различных параметрах вида *check_** для группы *gname*, и возвращает ошибки в последние параметры, а количество ошибок возвращается как значение функции.

```
PROCEDURE DBMS_REPCAT.WAIT_MASTER_LOG
    (gname IN VARCHAR,
    record_count IN NATURAL,
    timeout IN NATURAL,
    true_count OUT NATURAL);
```

Определяет количество незавершенных действий *true_count* для главной группы *gname* по истечении *timeout* секунд. Параметр *record_count* указывает количество незавершенных действий, которое приводит к возврату из процедуры.

DBMS_REPCAT_ADMIN

Создает пользователей с привилегиями, необходимыми для симметричного тиражирования. Появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_REPCAT_ADMIN.GRANT_ADMIN_ANY_SCHEMA
    (username IN VARCHAR2),
```

Предоставляет пользователю *username* привилегии на администрирование любых групп тиражирования текущего узла.

```
PROCEDURE DBMS_REPCAT_ADMIN.GRANT_ADMIN_SCHEMA
    (username IN VARCHAR2),
```

Предоставляет пользователю *username* привилегии на администрирование групп тиражирования в рамках данной схемы текущего узла.

```
PROCEDURE DBMS_REPCAT_ADMIN.REGISTER_USER_REPGROUP
    (username IN VARCHAR2,
    privilege_type IN VARCHAR2,
    {list_of_gnames IN VARCHAR2 |
    table_of_gnames IN DBMS_UTILITY.NAME_ARRAY});
```

Присваивает тип привилегии *privilege_type* (RECEIVER для получения привилегий или PROXY_SNAPADMIN для привилегий на доверенное администрирование материализованных представлений) пользователю *username* для списка групп из последнего параметра. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_REPCAT_ADMIN.REVOKE_ADMIN_ANY_SCHEMA
    (username IN VARCHAR2);
```

Отзывает все привилегии, выданные пользователю *username* процедурой GRANT_ADMIN_ANY_SCHEMA.

```
PROCEDURE DBMS_REPCAT_ADMIN.REVOKE_ADMIN_SCHEMA
    (username IN VARCHAR2);
```

Отзывает все привилегии, выданные пользователю *username* процедурой GRANT_ADMIN_SCHEMA.

```
PROCEDURE DBMS_REPCAT_ADMIN.UNREGISTER_USER_REPGROUP
    (username IN VARCHAR2,
    privilege_type IN VARCHAR2,
    {list_of_gnames IN VARCHAR2 |
    table_of_gnames IN DBMS_UTILITY.NAME_ARRAY});
```

Отзывает привилегии *privilege_type* (RECEIVER для получения привилегий или PROXY_SNAPADMIN для привилегий на доверенное администрирование материализованных представлений) пользователю *username* для списка групп из последнего параметра. Появилась в Oracle8*i*.

DBMS_REPCAT_AUTH

Предоставляет и отзывает псевдопривилегии для тиражирования. Не реализован в версии Oracle8*i* и выше.

Вызовы

```
PROCEDURE DBMS_REPACT_AUTH.GRANT_SURROGATE_REPCAT
    (userid IN VARCHAR2);
```

Предоставляет пользователю *userid* привилегии, необходимые для расширенных возможностей тиражирования.

```
PROCEDURE DBMS_REPACT_AUTH.REVOKE_SURROGATE_REPCAT
    (userid IN VARCHAR2);
```

Отзывает у пользователя *userid* привилегии, необходимые для расширенных возможностей тиражирования.

DBMS_REPCAT_INSTANTIATE

Управляет шаблонами развертывания. Появилась в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_REPCAT_INSTANTIATE.DROP_SITE_INSTANTIATION
    (refresh_template_name IN VARCHAR2,
    site_name IN VARCHAR2
    );
```

Удаляет шаблон *refresh_template_name* с узла *site_name*.

```
FUNCTION DBMS_REPCAT_INSTANTIATE.OFFLINE
    (refresh_template_name IN VARCHAR2,
    site_name IN VARCHAR2,
    runtime_parm_id IN NUMBER := -1e-130,
    next_date IN DATE := SYSDATE,
```

```
    interval IN VARCHAR := 'SYSDATE+1'
    [,user_default_gowner IN BOOLEAN := TRUE]#);
    RETURN NUMBER;
```

Формирует файл, используемый для создания среды материализованного представления на основе *refresh_template_name* на узле *site_name* при автономной работе. Параметр *runtime_parm_id* – это идентификатор, возвращаемый процедурой INSERT_RUNTIME_PARMS. Параметр *next_date* определяет следующую дату для создания группы обновления, а *interval* задает соответствующий интервал. Если значение параметра *use_default_gowner* равно TRUE, то все группы материализованных представлений принадлежат пользователю PUBLIC; в противном случае – пользователю, выполняющему тиражирование. Возвращает идентификатор сформированного сценария.

```
FUNCTION DBMS_REPCAT_INSTANTIATE.INSTANTIATE_ONLINE
    (refresh_template_name IN VARCHAR2,
    site_name IN VARCHAR2,
    runtime_parm_id IN NUMBER := -1e-130,
    next_date IN DATE := SYSDATE,
    interval IN VARCHAR := 'SYSDATE+1'
    [, user_default_gowner IN BOOLEAN := TRUE]#);
    RETURN NUMBER;
```

Формирует файл, используемый для создания среды материализованного представления на основе *refresh_template_name* на узле *site_name*, при работе в оперативном режиме. Параметр *runtime_parm_id*– это идентификатор, возвращаемый процедурой INSERT_RUNTIME_PARMS. Параметр *next_date* определяет следующую дату для создания группы обновления, а *interval* задает соответствующий интервал. Если значение параметра *user_default_gowner* равно TRUE, то все группы материализованных представлений принадлежат пользователю PUBLIC; в противном случае – пользователю, выполняющему тиражирование. Возвращает идентификатор сформированного сценария.

DBMS_REPCAT_RGT

Создает и поддерживает шаблоны развертывания. Появилась в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_REPCAT_RGT.ALTER_REFRESH_TEMPLATE
    (refresh_template_name IN VARCHAR2,
    new_owner IN VARCHAR2 := '-',
    new_refresh_group_name IN VARCHAR2 := '-',
    new_refresh_template_name IN VARCHAR2 := '-',
    new_template_comment IN VARCHAR2 := '-',
    new_public_template IN VARCHAR2 := '-',
    new_last_modified IN DATE := to_date('1', 'J'),
    new_modified_by IN NUMBER := -1e-130);
```

Изменяет шаблон *refresh_template_name*. Для того чтобы сохранить владельца шаблона, не вводите значения для *new_owner*.

```
PROCEDURE DBMS_REPCAT_RGT.ALTER_TEMPLATE_OBJECT
    (refresh_template_name IN VARCHAR2,
    object_name IN VARCHAR2,
    object_type IN VARCHAR2,
    new_refresh_template_name IN VARCHAR2 := '-',
```

```
    new_object_name IN VARCHAR2 := '-',
    new_object_type IN VARCHAR2 := '-',
    new_ddl_text IN CLOB := '-',
    new_master_rollback_seg IN VARCHAR2 := '-',
    [new_last_modified IN DATE := to_date('1', 'J'),]#
    new_flavor_id IN NUMBER := -1e-130);
```

Изменяет объект *object_name* типа *object_type* в шаблоне *refresh_template_name*. Если хотите сохранить тот же шаблон развертывания, не вводите значение для *refresh_template_name*. Параметр *new_last_modified* появился в Oracle9*i*.

```
PROCEDURE DBMS_REPCAT_RGT.ALTER_TEMPLATE_PARM
    (refresh_template_name IN VARCHAR2,
    parameter_name IN VARCHAR2,
    new_refresh_template_name IN VARCHAR2 := '-',
    new_parameter_name IN VARCHAR2 := '-',
    new_default_parm_value IN CLOB := NULL,
    new_prompt_string IN VARCHAR2 := '-',
    new_user_override IN VARCHAR2 := '-');
```

Изменяет параметр *parameter_name* в шаблоне *refresh_template_name*. Для того чтобы сохранить то же имя шаблона развертывания, не вводите значение для *refresh_template_name*.

```
PROCEDURE DBMS_REPCAT_RGT.ALTER_USER_AUTHORIZATION
    (user_name IN VARCHAR2,
    refresh_template_name IN VARCHAR2,
    new_user_name IN VARCHAR2:= '-',
    new_refresh_template_name IN VARCHAR2 := '-');
```

Заменяет имя пользователя *user_name* на новое *new_user_name* в шаблоне *refresh_template_name*. Для того чтобы сохранить то же имя шаблона развертывания, не вводите значение для *refresh_template_name*.

```
PROCEDURE DBMS_REPCAT_RGT.ALTER_USER_PARM_VALUE
    (refresh_template_name IN VARCHAR2,
    parameter_name IN VARCHAR2,
    user_name IN VARCHAR2,
    new_refresh_template_name IN VARCHAR2 := '-',
    new_parameter_name IN VARCHAR2 := '-',
    new_user_name IN VARCHAR2 := '-',
    new_parm_value IN CLOB := NULL);
```

Изменяет значение параметра *parameter_name* для пользователя *user_name* в шаблоне *refresh_template_name*. Для того чтобы сохранить прежние свойства параметра, не указывайте значения параметров *new_*.

```
FUNCTION DBMS_REPCAT_RGT.COMPARE_TEMPLATES
    (source_template_name IN VARCHAR2,
    compare_template_name IN VARCHAR2)
    RETURN NUMBER;
```

Сравнивает содержимое двух шаблонов развертывания. Возвращает указатель для просмотра отличий в представлении USER_REPCAT_TEMP_OUTPUT.

```
FUNCTION DBMS_REPCAT_RGT.COPY_TEMPLATE
    (old_refresh_template_name IN VARCHAR2,
    new_refresh_template_name IN VARCHAR2,
    copy_user_authorizations IN VARCHAR2,
    dblink IN VARCHAR2 := NULL)
    RETURN NUMBER;
```

Копирует шаблон *old_refresh_template* (возможно, с удаленного узла *dblink*) в шаблон *refresh_template_name*. Параметр *copy_user_authorizations* определяет, должны ли быть скопированы вместе с исходным шаблоном и разрешения для пользователей (Y или N). Возвращает число, используемое сервером Oracle во внутренних целях.

```
FUNCTION DBMS_REPCAT_RGT.CREATE_OBJECT_FROM_EXISTING
    (refresh_template_name IN VARCHAR2,
    object_name IN VARCHAR2,
    sname IN VARCHAR2,
    oname IN VARCHAR2,
    otype IN VARCHAR2)
    RETURN NUMBER;
```

Создает объект *object_name* типа *otype* на основе *sname.oname* и добавляет его в шаблон *refresh_template_name*. Возвращает число, используемое сервером Oracle во внутренних целях.

```
FUNCTION DBMS_REPCAT_RGT.CREATE_REFRESH_TEMPLATE
    (owner IN VARCHAR2,
    refresh_group_name IN VARCHAR2,
    refresh_template_name IN VARCHAR2,
    template_comment IN VARCHAR2 := NULL,
    public_template IN VARCHAR2 := NULL,
    last_modified IN DATE := SYSDATE,
    modified_by IN VARCHAR2 := USER,
    creation_date IN DATE := SYSDATE,
    created_by IN VARCHAR2 := USER)
    RETURN NUMBER;
```

Создает шаблон *refresh_template_name* в группе *refresh_template_group*, принадлежащий владельцу *owner*. Параметр *public_template* может иметь значение Y или N. Возвращает число, используемое сервером Oracle во внутренних целях.

```
FUNCTION DBMS_REPCAT_RGT.CREATE_TEMPLATE_OBJECT
    (refresh_template_name IN VARCHAR2,
    object_name IN VARCHAR2,
    object_type IN VARCHAR2,
    ddl_text IN CLOB,
    master_rollback_seg IN VARCHAR2 := NULL,
    flavor_id IN NUMBER := -1e-130)
    RETURN NUMBER;
```

Создает объект *object_name* типа *object_type* при помощи *ddl_text* в шаблоне *refresh_template_name*. Для создаваемого объекта будет использоваться сегмент отката *master_rollback_seg*. Параметр *flavor_id* и возвращаемое число используются сервером Oracle во внутренних целях.

```
FUNCTION DBMS_REPCAT_RGT.CREATE_TEMPLATE_PARM
    (refresh_template_name IN VARCHAR2,
    parameter_name IN VARCHAR2,
    default_parm_value IN CLOB := NULL,
    new_prompt_string IN VARCHAR2 := NULL,
    user_override IN VARCHAR2 := NULL)
    RETURN NUMBER;
```

Создает параметр *parameter_name* для шаблона *refresh_template_name*. Если параметр *user_override* содержит Y, то пользователь может подменить значение по умолчанию *default_parm_value*. Возвращает число, используемое сервером Oracle во внутренних целях.

```
FUNCTION DBMS_REPCAT_RGT.CREATE_USER_AUTHORIZATION
    (user_name IN VARCHAR2,
    refresh_template_name IN VARCHAR2)
    RETURN NUMBER;
```

Создает пользователя *user_name* для шаблона *refresh_template_name*. Возвращает число, используемое сервером Oracle во внутренних целях.

```
FUNCTION DBMS_REPCAT_RGT.CREATE_USER_PARM_VALUE
    (refresh_template_name IN VARCHAR2,
    parameter_name IN VARCHAR2,
    user_name IN VARCHAR2,
    parm_value IN CLOB := NULL)
    RETURN NUMBER;
```

Изменяет параметр *parameter_name* для пользователя *user_name* в шаблоне *refresh_template_name*. Возвращает число, используемое сервером Oracle во внутренних целях.

```
PROCEDURE DBMS_REPCAT_RGT.DELETE_RUNTIME_PARMS
    (runtime_parm_id IN NUMBER,
    parameter_name IN VARCHAR2);
```

Удаляет параметр *parameter_name* с идентификатором *runtime_parm_id*, созданный процедурой INSERT_RUNTIME_PARMS перед созданием шаблона развертывания.

```
PROCEDURE DBMS_REPCAT_RGT.DROP_ALL_OBJECTS
    (refresh_template_name IN VARCHAR2,
    object_type IN VARCHAR2 := NULL)
```

Удаляет все объекты типа *object_type* из шаблона *refresh_template_name*. Если *object_type* содержит NULL, то удаляются все объекты.

```
PROCEDURE DBMS_REPCAT_RGT.DROP_ALL_TEMPLATE_PARMS
    (refresh_template_name IN VARCHAR2,
    drop_objects IN VARCHAR2 := N);
```

Удаляет все параметры из шаблона *refresh_template_name*, если на них не ссылается объект шаблона. Если значение параметра *drop_objects* равно Y, то объекты, ссылающиеся на какие-либо параметры шаблона, и сами параметры удаляются из шаблона.

```
PROCEDURE DBMS_REPCAT_RGT.DROP_ALL_TEMPLATE_SITES
    (refresh_template_name IN VARCHAR2);
```

Удаляет все узлы из шаблона *refresh_template_name*.

```
PROCEDURE DBMS_REPCAT_RGT.DROP_ALL_TEMPLATES;
```

Удаляет все шаблоны с узла, на котором вызывается процедура.

```
PROCEDURE DBMS_REPCAT_RGT.DROP_ALL_USER_AUTHORIZATIONS
    (refresh_template_name IN VARCHAR2);
```

Удаляет все разрешения для пользователей на шаблон *refresh_template_name*.

```
PROCEDURE DBMS_REPCAT_RGT.DROP_ALL_USER_PARM_VALUES
    (refresh_template_name IN VARCHAR2,
    user_name IN VARCHAR2,
    parameter_name IN VARCHAR2):
```

Принимает любые комбинации двух последних параметров, позволяя удалять все параметры для пользователя, все параметры шаблона, определенный параметр для пользователя или же все параметры для всех пользователей из шаблона.

```
PROCEDURE DBMS_REPCAT_RGT.DROP_REFRESH_TEMPLATE
    (refresh_template_name IN VARCHAR2);
```

Удаляет шаблон *refresh_template_name*.

```
PROCEDURE DBMS_REPCAT_RGT.DROP_SITE_INSTANTIATION
    (refresh_template_name IN VARCHAR2,
    user_name IN VARCHAR2,
    site_name IN VARCHAR2,
    repapi_site_id IN NUMBER :=-1e-130);
```

Удаляет с узла *site_name* шаблон *refresh_template_name*, созданный пользователем *user_name*. Параметр *repapi_site_id* применяется только для Oracle8*i*.

```
PROCEDURE DBMS_REPCAT_RGT.DROP_TEMPLATE_OBJECT
    (refresh_template_name IN VARCHAR2,
    object_name IN VARCHAR2,
    object_type IN VARCHAR2);
```

Удаляет объект *object_name* типа *object_type* из шаблона *refresh_template_name*.

```
PROCEDURE DBMS_REPCAT_RGT.DROP_TEMPLATE_PARM
    (refresh_template_name IN VARCHAR2,
    parameter_name IN VARCHAR2);
```

Удаляет параметр *parameter_name* из шаблона *refresh_template_name*.

```
PROCEDURE DBMS_REPCAT_RGT.DROP_USER_AUTHORIZATION
    (refresh_template_name IN VARCHAR2
    user_name IN VARCHAR2);
```

Удаляет пользователя *user_name* из шаблона *refresh_template_name*.

```
PROCEDURE DBMS_REPCAT_RGT.DROP_USER_PARM_VALUE
    (refresh_template_name IN VARCHAR2,
    parameter_name IN VARCHAR2,
    user_name IN VARCHAR2);
```

Удаляет параметр *parameter_name* для пользователя *user_name* из шаблона *refresh_template_name*.

```
FUNCTION DBMS_REPCAT_RGT.GET_RUNTIME_PARM_ID
   RETURN NUMBER;
```

Возвращает идентификатор, необходимый при определении значения параметра времени выполнения.

```
PROCEDURE DBMS_REPCAT_RGT.INSERT_RUNTIME_PARMS
    (runtime_parm_id IN NUMBER,
    parameter_name IN VARCHAR2,
    parameter_value IN CLOB);
```

Назначает параметру шаблона *parameter_name* значение *parameter_value* параметра времени выполнения, идентификатором которого служит *runtime_parm_id*.

```
FUNCTION DBMS_REPCAT_RGT.INSTANTIATE_OFFLINE
    (refresh_template_name IN VARCHAR2,
    site_name IN VARCHAR2 := NULL,
    user_name IN VARCHAR := NULL,
    runtime_parm_id IN NUMBER := -1e-130,
    next_date IN DATE := SYSDATE,
```

```
    interval IN VARCHAR2 := 'SYSDATE+1'
    [,use_default_gowner IN BOOLEAN := TRUE]#)
    RETURN NUMBER;
```

Формирует сценарий на главном узле, используемом для создания узла материализованного представления на основе шаблона *refresh_template_name* для узла *site_name* от имени пользователя *user_name*, если удаленный узел не соединен с главным. Параметр *runtime_parm_id* – это идентификатор, возвращаемый процедурой INSERT_RUNTIME_PARMS. Параметр *next_date* определяет следующую дату для создания группы обновления, а *interval* задает соответствующий интервал. Если параметр *user_default_gowner* (появился в Oracle9*i*) установлен в TRUE, то все группы материализованных представлений принадлежат пользователю PUBLIC. Возвращает идентификатор сформированного сценария.

```
FUNCTION DBMS_REPCAT_RGT.INSTANTIATE_ONLINE
    (refresh_template_name IN VARCHAR2,
    site_name IN VARCHAR2 := NULL,
    user_name IN VARCHAR := NULL,
    runtime_parm_id IN NUMBER := -1e-130,
    next_date IN DATE := SYSDATE,
    interval IN VARCHAR2 := 'SYSDATE+1'
    [, use_default_gowner IN BOOLEAN := TRUE]#)
    RETURN NUMBER;
```

Формирует сценарий на главном узле, используемом для создания узла материализованного представления на основе шаблона *refresh_template_name* для узла *site_name* от имени пользователя *user_name*, если удаленный узел соединен с главным. Параметр *runtime_parm_id* – это идентификатор, возвращаемый процедурой INSERT_RUNTIME_PARMS. Параметр *next_date* определяет следующую дату для создания группы обновления, а *interval* задает соответствующий интервал. Если параметр *user_default_gowner* (появился в Oracle9*i*) установлен в TRUE, то все группы материализованных представлений принадлежат пользователю PUBLIC. Возвращает идентификатор сформированного сценария.

```
PROCEDURE DBMS_REPCAT_RGT.LOCK_TEMPLATE_EXCLUSIVE;
```

Блокирует шаблон на время его обновления или изменения.

```
PROCEDURE DBMS_REPCAT_RGT.LOCK_TEMPLATE_SHARED;
```

Блокирует шаблон, делая его доступным только для чтения.

DBMS_REPUTIL

Содержит процедуры и функции для формирования теневых таблиц, триггеров и пакетов для тиражирования таблиц.

Вызовы

```
PROCEDURE DBMS_REPUTIL.REPLICATION_OFF;
```

Позволяет изменять таблицы, не распространяя изменения на другие узлы.

```
PROCEDURE DBMS_REPUTIL.REPLICATION_ON;
```

Возобновляет тиражирование.

```
FUNCTION DBMS_REPUTIL.REPLICATION_IS_ON
    RETURN BOOLEAN;
```

Позволяет определить, выполняется ли тиражирование. Появилась в Oracle8*i*.

```
FUNCTION DBMS_REPUTIL.FROM_REMOTE
   RETURN BOOLEAN;
```

Возвращает TRUE в начале внутренних процедур тиражирования и FALSE – в конце. Появилась в Oracle8*i*.

```
FUNCTION DBMS_REPUTIL.GLOBAL_NAME
    RETURN VARCHAR2;
```

Возвращает глобальное имя локальной базы данных. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_REPUTIL.MAKE_INTERNAL_PKG
    (canon_sname IN VARCHAR2,
    canon_oname IN VARCHAR2);
```

Синхронизирует внутренние пакеты с таблицей или материализованным представлением *canon_sname.canon_oname* в каталоге тиражирования. Эту процедуру следует применять только при соответствующей рекомендации Oracle Support. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_REPUTIL.SYNC_UP_REP
    (canon_sname IN VARCHAR2,
    canon_oname IN VARCHAR2);
```

Синхронизирует внутренние триггеры с таблицей или материализованным представлением *canon_sname.canon_oname* в каталоге тиражирования. Эту процедуру следует применять только при соответствующей рекомендации Oracle Support. Появилась в Oracle8*i*.

DBMS_RESOURCE_MANAGER

Предоставляет процедуры для управления планами, группами потребителей и директивами плана, которые используются диспетчером ресурсов БД (Database Resource Manager – DRM). Появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_RESOURCE_MANAGER.CREATE_PLAN
    (plan IN VARCHAR2,
    comment IN VARCHAR2,
    cpu_mth IN VARCHAR DEFAULT 'EMPHASIS',
    active_sess_pool_mth | max_active_sess_target_mth
     IN VARCHAR2 DEFAULT 'ACTIVE_SESSION_POOL_ABSOLUTE',
    parallel_degree_limit_mth IN VARCHAR2 DEFAULT
     'PARALLEL_DEGREE_LIMIT_ABSOLUTE',
    queueing_mth IN VARCHAR2 DEFAULT 'FIFO_TIMEOUT');
```

Создает план ресурсов *plan* с методами, указанными в параметрах. В версии Oracle 8.2 параметр *max_active_sess_target_mth* был переименован в *active_sess_pool_mth*, добавился параметр *queueing_mth*.

```
PROCEDURE DBMS_RESOURCE_MANAGER.CREATE_SIMPLE_PLAN
   (simple_plan IN VARCHAR2,
    consumer_group1 IN VARCHAR2,
    group1_cpu IN VARCHAR2,
    consumer_group2 IN VARCHAR2,
    group2_cpu IN VARCHAR2,
    consumer_group3 IN VARCHAR2,
    group3_cpu IN VARCHAR2,
    consumer_group4 IN VARCHAR2,
```

```
    group4_cpu IN VARCHAR2,
    consumer_group5 IN VARCHAR2,
    group5_cpu IN VARCHAR2,
    consumer_group6 IN VARCHAR2,
    group6_cpu IN VARCHAR2,
    consumer_group7 IN VARCHAR2,
    group7_cpu IN VARCHAR2,
    consumer_group8 IN VARCHAR2,
    group8_cpu IN VARCHAR2);
```

Создает план ресурсов *simple_plan*, в котором один вызов может содержать до восьми групп потребителей. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_RESOURCE_MANAGER.UPDATE_PLAN
    (plan IN VARCHAR2,
    new_comment IN VARCHAR2,
    new_cpu_mth IN VARCHAR DEFAULT NULL,
    new_active_sess_pool_mth | new_max_active_sess_target_mth
     IN VARCHAR2 DEFAULT NULL,
    new_parallel_degree_limit_mth IN VARCHAR2 DEFAULT NULL,
    new_queueing_mth IN VARCHAR2 DEFAULT NULL
    [,new_group_switch_mth IN VARCHAR2 DEFAULT NULL)]#;
```

Обновляет план ресурсов *plan* с методами, указанными в параметрах. В версии Oracle 8.2 параметр *max_active_sess_target_mth* был переименован в *active_sess_pool_mth*, добавился параметр *new_queueing_mth*. Параметр *new_group_switch_mth* был добавлен в Oracle9*i*.

```
PROCEDURE DBMS_RESOURCE_MANAGER.DELETE_PLAN
    (plan IN VARCHAR2);
```

Удаляет план и директивы, на которые он ссылается.

```
PROCEDURE DBMS_RESOURCE_MANAGER.DELETE_PLAN_CASCADE
    (plan IN VARCHAR2);
```

Удаляет план и директивы, на которые он ссылается, а также всех потомков.

```
PROCEDURE DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP
    (consumer_group IN VARCHAR2,
    comment IN VARCHAR2,
    cpu_mth IN VARCHAR2 DEFAULT 'ROUND-ROBIN');
```

Создает группу потребителей *consumer_group* с комментарием *comment*, использующую для выделения ресурсов процессора метод *cpu_mth*.

```
PROCEDURE DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP
    (consumer_group IN VARCHAR2,
    new_comment IN VARCHAR2 DEFAULT NULL,
    new_cpu_mth IN VARCHAR2 DEFAULT NULL);
```

Обновляет группу потребителей *consumer_group* с комментарием *new_comment*, использующую для выделения ресурсов процессора метод *new_cpu_mth*.

```
PROCEDURE DBMS_RESOURCE_MANAGER.DELETE_CONSUMER_GROUP
    (consumer_group IN VARCHAR2);
```

Удаляет группу потребителей.

```
PROCEDURE DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE
    (plan IN VARCHAR2,
    group_or_subplan IN VARCHAR2,
    comment IN VARCHAR2,
```

```
    cpu_p1 IN NUMBER DEFAULT NULL,
    cpu_p2 IN NUMBER DEFAULT NULL,
    cpu_p3 IN NUMBER DEFAULT NULL,
    cpu_p4 IN NUMBER DEFAULT NULL,
    cpu_p5 IN NUMBER DEFAULT NULL,
    cpu_p6 IN NUMBER DEFAULT NULL,
    cpu_p7 IN NUMBER DEFAULT NULL,
    cpu_p8 IN NUMBER DEFAULT NULL,
    {active_sess_pool_p1 | max_active_sess_target_p1}
     IN NUMBER DEFAULT UNLIMITED,
    [,queueing_pl IN NUMBER DEFAULT UNLIMITED]#
    [,switch_group IN VARCHAR2 DEFAULT NULL]#
    [,switch_time IN NUMBER DEFAULT UNLIMITED]#
    [,switch_estimate IN BOOLEAN DEFAULT FALSE]#
    [,max_est_exec_time IN NUMBER DEFAULT UNLIMITED]#
    [,undo_pool IN NUMBER DEFAULT UNLIMITED,]#
    parallel_degree_limit_pl IN NUMBER DEFAULT UNLIMITED);
```

Создает план *plan*, являющийся частью *group_or_subplan*, с указанием до восьми методов выделения ресурсов процессора *cpu_p**. Параметр *switch_group* учитывает параметр *switch_time*, который оценивается, если значение параметра *switch_estimate* равно TRUE. В версии Oracle 8.2 параметр *max_active_sess_target_p1* был переименован в *active_sess_pool_p1*. Остальные параметры регулируют соответствующие значения. Параметры с *queueing_p1* по *undo_pool* появились в Oracle9*i*.

```
PROCEDURE DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE
    (plan IN VARCHAR2,
    group_or_subplan IN VARCHAR2,
    comment IN VARCHAR2,
    new_cpu_p1 IN NUMBER DEFAULT NULL,
    new_cpu_p2 IN NUMBER DEFAULT NULL,
    new_cpu_p3 IN NUMBER DEFAULT NULL,
    new_cpu_p4 IN NUMBER DEFAULT NULL,
    new_cpu_p5 IN NUMBER DEFAULT NULL,
    new_cpu_p6 IN NUMBER DEFAULT NULL,
    new_cpu_p7 IN NUMBER DEFAULT NULL,
    new_cpu_p8 IN NUMBER DEFAULT NULL,
    {new_active_sess_pool_p1 | new_max_active_sess_target_p1}
     IN NUMBER DEFAULT NULL
    [,new_queueing_p1 IN NUMBER DEFAULT NULL]#
    [,new_parallel_degree_limit_p1 IN NUMBER DEFAULT NULL]#
    [,new_switch_group IN VARCHAR2 DEFAULT NULL]#
    [,new_switch_time IN NUMBER DEFAULT NULL]#
    [,new_switch_estimate IN BOOLEAN DEFAULT FALSE]#
    [,new_max_est_exec_time IN NUMBER DEFAULT NULL]#
    [,new_undo_pool IN NUMBER DEFAULT NULL]#);
```

Обновляет план *plan*, являющийся частью *group_or_subplan*, с указанием до восьми методов выделения ресурсов процессора *cpu_p**. Параметр *new_switch_group* учитывает параметр *new_switch_time*, который оценивается, если параметр *new_switch_estimate* установлен в TRUE. В версии Oracle 8.2 параметр *new_max_active_sess_target_p1* был переименован в *new_active_sess_pool_p1*. Остальные параметры регулируют соответствующие значения. Параметры *new_queueing_p1* и с *new_switch_group* по *new_undo_pool* появились в Oracle9*i*.

```
PROCEDURE DBMS_RESOURCE_MANAGER.DELETE_PLAN_DIRECTIVE
    (plan IN VARCHAR2,
```

```
    group_or_subplan IN VARCHAR2);
```

Удаляет *plan*, являющийся частью *group_or_subplan*.

```
PROCEDURE DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA;
```

Создает временную рабочую область, в которой могут осуществляться изменения планов для объектов диспетчера ресурсов.

```
PROCEDURE DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA;
```

Проверяет корректность изменений в рабочей области.

```
PROCEDURE DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA;
```

Отменяет изменения, выполненные в рабочей области для диспетчера ресурсов.

```
PROCEDURE DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA;
```

Передает в рабочую область изменения для диспетчера ресурсов. Очищает рабочую область после проверки корректности всех изменений и их фиксации.

```
PROCEDURE DBMS_RESOURCE_MANAGER.SET_INITIAL_CONSUMER_GROUP
    (user IN VARCHAR2,
    consumer_group IN VARCHAR2);
```

Устанавливает первоначальную группу потребителей *consumer_group* для пользователя *user*.

```
PROCEDURE DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_SESSION
    (session_id IN NUMBER,
    session_serial IN NUMBER,
    consumer_group IN VARCHAR2);
```

Переводит сеанс, определяемый параметрами *session_id* (SID) и *session_serial* (столбец SERIAL# из V$SESSION), в группу *consumer_group* вместе со всеми подчиненными сеансами (если таковые имеются).

```
PROCEDURE DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_USER
    (user IN VARCHAR2,
    consumer_group IN VARCHAR2);
```

Переводит все сеансы пользователя *user* в группу *consumer_group* вместе со всеми подчиненными сеансами (если таковые имеются).

DBMS_RESOURCE_MANAGER_PRIVS

Применяется для управления привилегиями, связанными с диспетчером ресурсов. Появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SYSTEM_PRIVILEGE
    (grantee_name IN VARCHAR2,
    privilege_name IN VARCHAR2 DEFAULT 'ADMINISTER_RESOURCE_MANAGER',
    admin_option IN BOOLEAN);
```

Выдает привилегию *privilege_name* для *grantee_name*. Если значение параметра *admin_option* равно TRUE, то получивший привилегию имеет право выдавать ее.

```
PROCEDURE DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SYSTEM_PRIVILEGE
    (grantee_name IN VARCHAR2,
    privilege_name IN VARCHAR2 DEFAULT 'ADMINISTER_RESOURCE_MANAGER');
```

Отзывает привилегию *privilege_name* у *grantee_name*.

```
PROCEDURE DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP
    (grantee_name IN VARCHAR2,
    consumer_group IN VARCHAR2,
    grant_option IN BOOLEAN);
```

Предоставляет *grantee_name* право смены группы *consumer_group*. Если значение параметра *admin_option* равно TRUE, то получивший такое право может передавать его.

```
PROCEDURE DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SWITCH_CONSUMER_GROUP
    (grantee_name IN VARCHAR2,
    consumer_group IN VARCHAR2);
```

Отзывает у *grantee_name* право смены группы *consumer_group*.

DBMS_RESUMABLE

Содержит процедуры и функции для управления возобновляемыми операциями, испытывающими недостаток памяти или достигшими границы выделенной области. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_RESUMABLE.ABORT
    (session_id IN NUMBER);
```

Прерывает приостановленную операцию выделения памяти в сеансе *session_id*.

```
FUNCTION DBMS_RESUMABLE.GET_SESSION_TIMEOUT
    (session_id IN NUMBER);
```

Возвращает таймаут для возобновляемого выделения памяти для сеанса *session_id*.

```
FUNCTION DBMS_RESUMABLE.GET_SESSION_TIMEOUT
    (session_id IN NUMBER,
    timeout IN NUMBER);
```

Устанавливает таймаут для возобновляемого выделения памяти для сеанса *session_id*.

```
FUNCTION DBMS_RESUMABLE.GET_TIMEOUT;
```

Возвращает значение таймаута для текущего сеанса.

```
PROCEDURE DBMS_RESUMABLE.SET_TIMEOUT
    (timeout IN NUMBER);
```

Устанавливает таймаут для текущего сеанса.

```
FUNCTION DBMS_RESUMABLE.SPACE_ERROR_INFO
    (error_type OUT VARCHAR2,
    object_type OUT VARCHAR2,
    object_owner OUT VARCHAR2,
    table_space_name OUT VARCHAR2,
    object_name OUT VARCHAR2,
    sub_object_name OUT VARCHAR2)
    RETURN BOOLEAN;
```

В случае ошибки, связанной с нехваткой памяти, возвращает информацию в параметры и TRUE как значение функции; в противном случае возвращает FALSE.

DBMS_RLS

Обеспечивает доступ к административному интерфейсу детального контроля доступа Oracle. Этот пакет появился в Oracle8*i* и доступен только в рамках Enterprise Edition.

Вызовы

```
PROCEDURE DBMS_RLS.ADD_POLICY
    (object_schema IN VARCHAR2 DEFAULT NULL,
    object_name IN VARCHAR2,
    policy_name IN VARCHAR2,
    function_schema IN VARCHAR2 DEFAULT NULL,
    policy_function IN DEFAULT VARCHAR2,
    statement_types IN VARCHAR2 DEFAULT NULL,
    update_check IN BOOLEAN DEFAULT FALSE,
    enable IN BOOLEAN DEFAULT TRUE
    [,static_policy IN BOOLEAN DEFAULT FALSE]#);
```

Добавляет политику безопасности *policy_name* для объекта *object_schema.object_name*. Политика использует *function_schema.policy_function* и применяется к типам команд *statement_types*, которыми могут быть любые комбинации SELECT, INSERT, UPDATE и DELETE. Если параметр *update_check* установлен в TRUE, то сервер проверяет значение политики после выполнения команды INSERT или UPDATE. Если параметр *static_policy* (появился в Oracle9*i*) установлен в TRUE, то сервер считает, что политика формирует единый предикат для всех пользователей, за исключением SYS или пользователей с привилегией EXEMPT ACCESS POLICY.

```
PROCEDURE DBMS_RLS.DROP_POLICY
    (object_schema IN VARCHAR2 DEFAULT NULL,
    object_name IN VARCHAR2,
    policy_name IN VARCHAR2);
```

Удаляет политику *policy_name* для объекта *object_schema.object_name*.

```
PROCEDURE DBMS_RLS.REFRESH_POLICY
    (object_schema IN VARCHAR2 DEFAULT NULL,
    object_name IN VARCHAR2,
    policy_name IN VARCHAR2);
```

Инициирует повторный синтаксический анализ всех кэшированных команд, сопоставленных объекту *object_schema.object_name*.

```
PROCEDURE DBMS_RLS.ENABLE_POLICY
    (object_schema IN VARCHAR2 := NULL,
    object_name IN VARCHAR2,
    policy_name IN VARCHAR2,
    enable IN BOOLEAN);
```

Включает (если параметр enable установлен в TRUE) или отключает (если параметр enable установлен в FALSE) политику *policy_name* для объекта *object_schema.object_name*.

```
PROCEDURE DBMS_RLS.CREATE_POLICY_GROUP
    (object_schema IN VARCHAR2 DEFAULT NULL,
    object_name IN VARCHAR2,
    policy_group IN VARCHAR2);
```

Создает группу политик безопасности *policy_group* для объекта *object_schema.object_name*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_RLS.ADD_GROUPED_POLICY
    (object_schema IN VARCHAR2,
    object_name IN VARCHAR2,
    policy_group IN VARCHAR2,
    policy_name IN VARCHAR2,
    function_schema IN VARCHAR2,
    policy_function IN DEFAULT VARCHAR2,
    statement_types IN VARCHAR2,
    update_check IN BOOLEAN,
    enable IN BOOLEAN,
    static_policy IN BOOLEAN DEFAULT FALSE);
```

Добавляет политику *policy_name* в группу *policy_group* для объекта *object_schema.object_name*. Параметры совпадают с параметрами процедуры ADD_POLICY. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_RLS.ADD_POLICY_CONTEXT
    (object_schema IN VARCHAR2,
    object_name IN VARCHAR2,
    namespace IN VARCHAR2,
    attribute IN VARCHAR2);
```

Добавляет контекст (атрибут *attribute* и пространство имен *namespace*) для объекта *object_schema.object_name*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_RLS.DELETE_POLICY_GROUP
    (object_schema IN VARCHAR2 DEFAULT NULL,
    object_name IN VARCHAR2,
    policy_group IN VARCHAR2);
```

Удаляет группу политик *policy_group* для объекта *object_schema.object_name*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_RLS.DROP_GROUPED_POLICY
    (object_schema IN VARCHAR2,
    object_name IN VARCHAR2,
    policy_group IN VARCHAR2,
    policy_name IN VARCHAR2);
```

Удаляет политику *policy_name* для объекта *object_schema.object_name* из группы *policy_group*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_RLS.DROP_POLICY_CONTEXT
    (object_schema IN VARCHAR2,
    object_name IN VARCHAR2,
    namespace IN VARCHAR2,
    attribute IN VARCHAR2);
```

Удаляет атрибут и пространство имен контекста для объекта *object_schema.object_name*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_RLS.ENABLE_GROUPED_POLICY
    (object_schema IN VARCHAR2,
    object_name IN VARCHAR2,
    group_name IN VARCHAR2,
    policy_name IN VARCHAR2,
    enable IN BOOLEAN);
```

Включает (если параметр *enable* установлен в TRUE) или отключает (если параметр *enable* установлен в FALSE) политику *policy_name* группы *group_name* для объекта *object_schema.object_name*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_RLS.REFRESH_GROUPED_POLICY
    (object_schema IN VARCHAR2,
    object_name IN VARCHAR2,
    group_name IN VARCHAR2,
    policy_name IN VARCHAR2);
```

Инициирует повторный синтаксический анализ всех кэшированных команд, сопоставленных политике *policy_name* группы *group_name* для объекта *object_schema.object_name*. Появилась в Oracle9*i*.

DBMS_ROWID

Содержит программы для работы с типом ROWID. Структура ROWID изменилась в Oracle8, и пакет работает как со старым, так и с новым типами ROWID для версий выше Oracle8.

В Oracle7 номер ROWID (ограниченный) состоит из трех частей в шестнадцатеричной системе счисления:

```
BBBBBBBB.RRRR.FFFF
```

В Oracle8 номер ROWID (расширенный) состоит из четырех частей в кодировке base64:

```
OOOOOOFFFBBBBBBRRR
```

где

OOOOOO – это номер объекта.

FFFF (FFF) – это абсолютный (Oracle7) или относительный (Oracle8) номер файла.

BBBBBBBB (BBBBBB) – это номер блока в файле.

RRRR (RRR) – это номер строки в блоке.

Вызовы

```
FUNCTION DBMS_ROWID.ROWID_BLOCK_NUMBER
     (row_id IN ROWID)
   RETURN NUMBER;
```

Возвращает составляющую номера блока для *row_id*.

```
FUNCTION DBMS_ROWID.ROWID_CREATE
    (rowid_type IN NUMBER,
    object_number IN NUMBER,
    relative_fno IN NUMBER,
    block_number IN NUMBER,
    row_number IN NUMBER)
    RETURN ROWID;
```

Создает ROWID типа *rowid_type*, состоящий из номера объекта *object_number*, относительного номера файла *relative_fno*, номера блока *block_number* и номера строки *row_number*. Параметр *rowid_type* может иметь значение ROWID_TYPE_EXTENDED или ROWID_TYPE_RESTRICTED. Параметр *object_number* может содержать ROWID_OBJECT_UNDEFINED или номер объекта (OID).

```
PROCEDURE DBMS_ROWID.ROWID_INFO
    (rowid_in IN ROWID,
    rowid_type OUT NUMBER,
    object_number OUT NUMBER,
    relative_fno OUT NUMBER,
```

```
    block_number OUT NUMBER,
    row_number OUT NUMBER);
```

Разбирает *rowid_in* на составляющие. Параметр *rowid_type* может иметь значение ROWID_TYPE_EXTENDED или ROWID_TYPE_RESTRICTED. Параметр *object_number* может содержать ROWID_OBJECT_UNDEFINED или номер объекта (OID).

```
FUNCTION DBMS_ROWID.ROWID_OBJECT
    (row_id IN ROWID)
    RETURN NUMBER;
```

Возвращает составляющую номера объекта для *row_id*.

```
FUNCTION DBMS_ROWID.ROWID_RELATIVE_FNO
    (row_id IN ROWID)
    RETURN NUMBER;
```

Возвращает составляющую относительного номера файла для *row_id*.

```
FUNCTION DBMS_ROWID.ROWID_ROW_NUMBER
    (row_id IN ROWID)
    RETURN NUMBER;
```

Возвращает составляющую номера строки для *row_id*.

```
FUNCTION DBMS_ROWID.ROWID_TO_ABSOLUTE_FNO
    (row_id IN ROWID,
    schema_name IN VARCHAR2,
    object_name IN VARCHAR2)
    RETURN NUMBER;
```

Возвращает абсолютный номер файла для *row_id*, схемы *schema_name* и объекта *object_name*.

```
FUNCTION DBMS_ROWID.ROWID_TO_EXTENDED
    (old_rowid IN ROWID,
    schema_name IN VARCHAR2,
    object_name IN VARCHAR2,
    conversion_type IN INTEGER)
   RETURN ROWID;
```

Возвращает расширенный номер ROWID для ограниченного значения *old_rowid*, схемы *schema_name* и объекта *object_name* при помощи преобразования *conversion_type*. Параметр *conversion_type* (появился в Oracle8*i*) может иметь значение ROWID_CONVERT_INTERNAL (которое показывает, что номер ROWID хранился в столбце типа ROWID) или ROWID_CONVERT_EXTERNAL (которое показывает, что номер ROWID хранился в столбце типа CHAR/VARCHAR/VARCHAR2).

```
FUNCTION DBMS_ROWID.ROWID_TO_RESTRICTED
    (old_rowid IN ROWID,
    conversion_type IN INTEGER)
    RETURN ROWID;
```

Возвращает ограниченный номер ROWID для расширенного номера *old_rowid*, получаемый при помощи преобразования *conversion_type*. Параметр *conversion_type* может иметь значение ROWID_CONVERT_INTERNAL (которое показывает, что номер ROWID будет храниться в столбце типа ROWID) или ROWID_CONVERT_EXTERNAL (которое показывает, что номер ROWID будет храниться в столбце типа CHAR/VARCHAR/VARCHAR2).

```
FUNCTION DBMS_ROWID.ROWID_TYPE
    (row_id IN ROWID)
    RETURN NUMBER;
```

Возвращает ROWID_TYPE_EXTENDED или ROWID_TYPE_RESTRICTED для *row_id*.

```
FUNCTION DBMS_ROWID.ROWID_VERIFY
    (rowid_in IN ROWID,
    schema_name IN VARCHAR2,
    object_name IN VARCHAR2,
    conversion_type IN INTEGER)
    RETURN NUMBER;
```

Возвращает ROWID_VALID или ROWID_INVALID для *rowid_in*, схемы *schema_name* и объекта *object_name* при помощи преобразования *conversion_type*. Параметр *conversion_type* (появился в Oracle8*i*) может иметь значение ROWID_CONVERT_INTERNAL (которое показывает, что номер ROWID хранится в столбце типа ROWID) или ROWID_CONVERT_EXTERNAL (которое показывает, что номер ROWID хранится в столбце типа CHAR/VARCHAR/VARCHAR2).

DBMS_RULE

Оценивает правила из набора правил в определенном контексте. Появилась в Oracle9*i*.

Вызов

```
PROCEDURE DBMS_RULE.EVALUATE
    (rule_set_name IN VARCHAR2,
    evaluation_context IN VARCHAR2,
    event_context IN SYS.RE$NVLIST DEFAULT NULL,
    table_values IN SYS.RE$TABLE_VALUE_LIST DEFAULT NULL,
    [column_values IN SYS.RE$COLUMN_VALUE_LIST,]
    variable_values IN SYS.RE$VARIABLE_VALUE_LIST DEFAULT NULL,
    [attribute_values IN SYS.RE$ATTRIBUTE_VALUE_LIST,]
    stop_on_first_hit IN BOOLEAN DEFAULT FALSE,
    simple_rules_only IN BOOLEAN DEFAULT FALSE,
    true_rules OUT SYS.RE$RULE_HIT_LIST,
    maybe_rules OUT SYS.RE$RULE_HIT_LIST);
```

Оценивает набор правил *rule_set_name* на основе контекста оценки *evaluation_context* и контекста события *event_context* (список пар имя/значение, которые определяют события, вызывающие оценку). Значения берутся из указанных таблиц. Если значение параметра *stop_on_first_hit* равно TRUE, то обработка останавливается, как только обнаружено первое правило, оцененное как TRUE, или (в отсутствие правила TRUE) когда обнаружено правило, которое могло бы быть оценено как TRUE при наличии большего объекта данных. Если параметр *simple_rules_only* установлен в TRUE, то процедура оценивает только правила, которые достаточно просты для того, чтобы для их оценки не требовалось выполнение команд SQL. Параметр *true_rules* возвращает правила, которые оцениваются как TRUE для заданных данных. Параметр *maybe_rules* возвращает правила, которые не удалось оценить из-за нехватки данных или из-за сложности правила, если задан параметр *simple_rules_only*. Параметры *column_values* и *attribute_values* или оба присутствуют, или оба отсутствуют в вызове процедуры.

DBMS_RULE_ADMIN

Содержит процедуры для администрирования правил, наборов правил и контекстов оценки правил. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_RULE_ADMIN.ADD_RULE
    (rule_name IN VARCHAR2,
    rule_set_name IN VARCHAR2,
    evaluation_context IN VARCHAR2 DEFAULT NULL,
    rule_comment IN VARCHAR2 DEFAULT NULL);
```

Добавляет правило *rule_name* в набор правил *rule_set_name* с необязательным указанием контекста оценки *evaluation_context* и комментария к правилу *rule_comment*.

```
PROCEDURE DBMS_RULE_ADMIN.ALTER_RULE
    (rule_name IN VARCHAR2,
    condition IN VARCHAR2 DEFAULT NULL,
    evaluation_context IN VARCHAR2 DEFAULT NULL,
    remove_evaluation_context IN BOOLEAN DEFAULT FALSE,
    action_context IN SYS.RE$NV_LIST DEFAULT NULL,
    remove_action_context IN BOOLEAN DEFAULT FALSE,
    rule_comment IN VARCHAR2 DEFAULT NULL,
    remove_rule_comment IN BOOLEAN DEFAULT FALSE);
```

Изменяет правило *rule_name*, изменяя условие *condition* и изменяя или удаляя контекст оценки, контекст действия или правило.

```
PROCEDURE DBMS_RULE_ADMIN.CREATE_EVALUATION_CONTEXT
    (evaluation_context_name IN VARCHAR2,
    table_aliases IN SYS.RE$TABLE_ALIAS_LIST DEFAULT NULL,
    variable_types IN SYS.RE$VARIABLE_TYPE_LIST DEFAULT NULL,
    evaluation_function IN VARCHAR2 DEFAULT NULL,
    evaluation_context_comment IN VARCHAR2 DEFAULT NULL);
```

Создает контекст оценки *evaluation_context_name* с псевдонимами таблиц и переменными, указанными в таблицах. Параметр *evaluation_function* – это необязательная функция, которая будет применяться для оценки правил в данном контексте; она имеет ту же форму, что и процедура DBMS_RULE.EVALUATE.

```
PROCEDURE DBMS_RULE_ADMIN.CREATE_RULE
    (rule_name IN VARCHAR2,
    condition IN VARCHAR2 DEFAULT NULL,
    evaluation_context IN VARCHAR2 DEFAULT NULL,
    action_context IN SYS.RE$NV_LIST DEFAULT NULL,
    rule_comment IN VARCHAR2 DEFAULT NULL);
```

Создает правило *rule_name* с условием *condition*, которое может оцениваться как логическое с необязательным указанием контекста оценки и комментария к правилу: *evaluation_context* и *rule_set_comment*.

```
PROCEDURE DBMS_RULE_ADMIN.CREATE_RULE_SET
    (rule_set_name IN VARCHAR2,
    evaluation_context IN VARCHAR2 DEFAULT NULL,
    rule_set_comment IN VARCHAR2 DEFAULT NULL);
```

Создает набор правил *rule_set_name* с необязательным указанием контекста оценки и комментария:

```
PROCEDURE DBMS_RULE_ADMIN.DROP_EVALUATION_CONTEXT
    (evaluation_context IN VARCHAR2,
    force IN BOOLEAN DEFAULT FALSE);
```

Удаляет контекст *evaluation_context*. Если значение параметра force равно TRUE, то процедура удаляет *evaluation_context* и все правила и наборы правил, которые

его использовали; в противном случае процедура или удаляет контекст оценки, если его не используют никакие правила и наборы правил, или же возвращает исключение.

```
PROCEDURE DBMS_RULE_ADMIN.DROP_RULE
    (rule_name IN VARCHAR2,
    force IN BOOLEAN DEFAULT FALSE);
```

Удаляет правило *rule_name*. Если значение параметра *force* равно TRUE, то процедура удаляет правило и все наборы правил, которые его использовали; в противном случае процедура удаляет или правило, если его не использует никакой набор правил, или же возвращает исключение.

```
PROCEDURE DBMS_RULE_ADMIN.DROP_RULE_SET
    (rule_set_name IN VARCHAR2,
    delete_rules IN BOOLEAN DEFAULT FALSE);
```

Удаляет набор правил *rule_set_name*. Если значение параметра *delete_rules* равно TRUE, то процедура удаляет все правила из набора; в противном случае правила не удаляются.

```
PROCEDURE DBMS_RULE_ADMIN.GRANT_OBJECT_PRIVILEGE
    (privilege IN BINARY_INTEGER,
    object_name IN VARCHAR2,
    grantee IN VARCHAR2,
    grant_option IN BOOLEAN DEFAULT FALSE);
```

Предоставляет объектную привилегию на правило для объекта *object_name* пользователю *grantee*. Если значение параметра *grant_option* равно TRUE, то получивший такую привилегию может, в свою очередь, выдавать ее.

```
PROCEDURE DBMS_RULE_ADMIN.GRANT_SYSTEM_PRIVILEGE
    (privilege IN BINARY_INTEGER,
    grantee IN VARCHAR2,
    grant_option IN BOOLEAN DEFAULT FALSE);
```

Предоставляет системную привилегию на правило пользователю *grantee*. Если значение параметра *grant_option* равно TRUE, то получивший такую привилегию может, в свою очередь, выдавать ее.

```
PROCEDURE DBMS_RULE_ADMIN.REMOVE_RULE
    (rule_name IN VARCHAR2,
    rule_set_name IN VARCHAR2,
    evaluation_context IN VARCHAR2 DEFAULT NULL,
    all_evaluation_contexts IN BOOLEAN DEFAULT FALSE);
```

Удаляет правило *rule_name* из набора *rule_set_name*. Если контекст оценки *evaluation_context* был задан при помощи процедуры ADD_RULE, он должен быть указан в этой процедуре. Если параметр *all_evaluation_contexts* содержит TRUE, то правило удаляется из всех контекстов оценки; в противном случае оно удаляется только из наборов правил с указанным контекстом оценки.

```
PROCEDURE DBMS_RULE_ADMIN.REVOKE_OBJECT_PRIVILEGE
    (privilege IN BINARY_INTEGER,
    object_name IN VARCHAR2,
    revokee IN VARCHAR2);
```

Отзывает объектную привилегию на правило для объекта *object_name* у пользователя *revokee*.

```
PROCEDURE DBMS_RULE_ADMIN.REVOKE_SYSTEM_PRIVILEGE
    (privilege IN BINARY_INTEGER,
```

```
    revokee IN VARCHAR2);
```

Отзывает системную привилегию на правило у пользователя *revokee*.

DBMS_SESSION

Позволяет устанавливать и изменять настройки сеанса, включать и отключать роли, а также управлять ресурсами сеанса.

Вызовы

```
PROCEDURE DBMS_SESSION.CLEAR_IDENTIFIER;
```

Удаляет значение клиентского идентификатора сеанса. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_SESSION.CLOSE_DATABASE_LINK
    (dblink IN VARCHAR2);
```

Закрывает канал связи БД *dblink* или порождает исключение, если *dblink* не открыт или применяется.

```
PROCEDURE DBMS_SESSION.FREE_UNUSED_USER_MEMORY;
```

Освобождает память сеанса и возвращает ее операционной системе (для выделенного соединения) или разделяемому пулу Oracle (для соединения Shared Server).

```
FUNCTION DBMS_SESSION.IS_ROLE_ENABLED
      (rolename IN VARCHAR2)
    RETURN BOOLEAN;
```

Возвращает TRUE, если роль *rolename* разрешена для сеанса на текущий момент.

```
FUNCTION DBMS_SESSION.IS_SESSION_ALIVE
    (uniqueid IN VARCHAR2)
    RETURN BOOLEAN;
```

Возвращает TRUE, если сеанс, идентифицируемый *uniqueid*, еще не завершен. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_SESSION.LIST_CONTEXT
    (list OUT AppCtxTabTyp,
    size OUT NUMBER);
```

Возвращает информацию о текущем контексте в таблицу типа AppCtxTabTyp. Параметр *size* определяет количество записей, возвращаемых в буфер. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_SESSION.MODIFY_PACKAGE_STATE
    (action_flags IN PLS_INTEGER);
```

Изменяет состояние пакета в соответствии со значением флага FREE_ALL_RESOURCES, когда освобождается вся память, сопоставленная ранее запущенным программам PL/SQL, очищаются значения глобальных переменных пакета и закрываются кэшированные курсоры, или значением флага REINITIALIZE, когда повторно инициализируются пакеты и заново применяется выделенная им память. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_SESSION.LIST_CONTEXT
    (list OUT AppCtxTabTyp,
    size OUT NUMBER);
```

Возвращает список пространств имен, атрибутов и значений для текущего сеанса. Параметр *size* определяет количество возвращаемых записей. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_SESSION.RESET_PACKAGE;
```

Возвращает все пакеты сеанса в исходное состояние, уничтожая значения всех глобальных переменных пакета.

```
PROCEDURE DBMS_SESSION.SET_CLOSE_CACHED_OPEN_CURSORS
   (close_cursors IN BOOLEAN);
```

Заменяет значение параметра БД CLOSE_CACHED_OPEN_CURSORS для сеанса значением *close_cursors*.

```
PROCEDURE DBMS_SESSION.SET_CONTEXT
    (namespace IN VARCHAR2,
    attribute IN VARCHAR2,
    [,value IN VARCHAR2]#
    [,user_name IN VARCHAR2]#
    client_id IN VARCHAR2);
```

Устанавливает атрибут *attribute* контекста приложения в значение *value* для пространства имен *namespace*. Можно одновременно указать оба параметра: *user_name* и *client_id*. Появилась в Oracle8*i*. Параметры *user_name* и *value* появились в Oracle9*i*.

```
PROCEDURE DBMS_SESSION.SET_IDENTIFIER
    (client_id IN VARCHAR);
```

Задает *client_id* для сеанса. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_SESSION.SET_LABEL
    (lbl IN VARCHAR2);
```

Устанавливает метку сеанса Trusted Oracle в значение *lbl*. Не поддерживается начиная с версии Oracle8*i*.

```
PROCEDURE DBMS_SESSION.SET_MLS_LABEL_FORMAT
    (fmt IN VARCHAR2);
```

Устанавливает формат метки по умолчанию для сеанса Trusted Oracle в *fmt*. Не поддерживается начиная с версии Oracle8*i*.

```
PROCEDURE DBMS_SESSION.SET_NLS
    (param IN VARCHAR2,
    value IN VARCHAR2);
```

Устанавливает параметр поддержки национальных языков *param* в значение *value*. Если *value* – это маска формата, то необходимо помещать строки в тройные кавычки.

```
PROCEDURE DBMS_SESSION.SET_ROLE
    (role_cmd IN VARCHAR2);
```

Включает и выключает роль (роли), добавляя *role_cmd* в команду SET ROLE и выполняя ее. Можно отменить все роли, установив *role_cmd* в NONE.

```
PROCEDURE DBMS_SESSION.SET_SQL_TRACE
    (sql_trace IN BOOLEAN);
```

Включает и выключает трассировку SQL для сеанса в зависимости от значения параметра *sql_trace* (TRUE или FALSE соответственно). Данная процедура действует аналогично процедуре DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION.

```
PROCEDURE DBMS_SESSION.SWITCH_CURRENT_CONSUMER_GROUP
    (new_consumer_group IN VARCHAR2,
    old_consumer_group OUT VARCHAR2,
    initial_group_on_error IN BOOLEAN);
```

Переключает текущую группу потребителей в группу *new_consumer_group* и возвращает имя старой группы *old_consumer_group*. Если значение параметра *initial_group_on_error* равно TRUE, то устанавливает текущую группу потребителей в исходную группу при возникновении ошибки.

```
FUNCTION DBMS_SESSION.UNIQUE_SESSION_ID
    RETURN VARCHAR2;
```

Возвращает уникальный для сеанса строковый идентификатор длиной до 24 байт.

DBMS_SHARED_POOL

Содержит программы, применяемые для управления разделяемым пулом системной глобальной области (SGA).

Вызовы

```
PROCEDURE DBMS_SHARED_POOL.ABORTED_REQUEST_THRESHOLD
    (threshold_size IN NUMBER);
```

Устанавливает максимальный размер объекта, для которого разделяемый пул будет удалять другие объекты для выделения участка памяти. При попытке загрузки объектов, размер которых превышает *threshold_size* байт, процедура возвращает ошибку ORA-04031 при отсутствии необходимого пространства. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_SHARED_POOL.KEEP
    (name IN VARCHAR2,
    flag IN CHAR DEFAULT 'P');
```

Закрепляет объект, идентифицируемый по имени *name*, в разделяемом пуле. Тип объекта определяется значением параметра *flag*: 'P' или 'p' для пакета, процедуры или функции; 'C' или 'c' для курсора (в Oracle8); 'Q' или 'q' для последовательностей; 'R' или 'r' для триггеров. Любое другое символьное значение *flag* служит для прикрепления курсора, определяемого адресом и значением хеш-функции (из V$SQLAREA).

```
PROCEDURE DBMS_SHARED_POOL.SIZES
    (minsize IN NUMBER);
```

Отображает те объекты и курсоры разделяемого пула, размер которых превышает *minsize* килобайт.

```
PROCEDURE DBMS_SHARED_POOL.UNKEEP
    (name IN VARCHAR2,
    flag IN CHAR DEFAULT 'P');
```

Отменяет закрепление объекта типа *flag* с именем *name* из разделяемого пула. Для параметра *flag* допустимы те же значения, что и в процедуре DBMS_SHARED_POOL.KEEP.

DBMS_SNAPSHOT

Этот пакет заменен пакетом DBMS_MVIEW в Oracle9*i*.

DBMS_SPACE

Содержит процедуры, предоставляющие информацию о расходовании внутреннего пространства и списках свободных блоков для сегментов таблиц, индексов и кластеров.

Вызовы

```
PROCEDURE DBMS_SPACE.FREE_BLOCKS
    (segment_owner IN VARCHAR2,
    segment_name IN VARCHAR2,
    segment_type IN VARCHAR2,
    freelist_group_id IN NUMBER,
    free_blks OUT NUMBER
    [,scan_limit IN NUMBER DEFAULT NULL]
    [,partition_name IN VARCHAR2 DEFAULT NULL]);
```

Возвращает в параметр *free_blks* количество свободных блоков экземпляра *freelist_group_id* для сегмента *segment_name* типа *segment_type* (TABLE, INDEX или CLUSTER), принадлежащего владельцу *segment_owner*, в разделе *partition_name* (необязательный параметр, появился в Oracle8). Параметр *scan_limit* (необязательный) ограничивает количество просматриваемых свободных блоков.

```
PROCEDURE DBMS_SPACE.SPACE_USAGE
    (segment_owner IN VARCHAR2,
    segment_name IN VARCHAR2,
    segment_type IN VARCHAR2,
    unformatted_blocks OUT NUMBER,
    unformatted_bytes OUT NUMBER,
    fs1_blocks OUT NUMBER,
    fs1_bytes OUT NUMBER,
    fs2_blocks OUT NUMBER,
    fs2_bytes OUT NUMBER,
    fs3_blocks OUT NUMBER,
    fs3_bytes OUT NUMBER,
    fs4_blocks OUT NUMBER,
    fs4_bytes OUT NUMBER,
    full_blocks OUT NUMBER,
    full_bytes OUT NUMBER,
    partition_name IN VARCHAR2 DEFAULT NULL);
```

Возвращает объем свободного пространства в сегменте *segment_name* типа *segment_type* (TABLE, INDEX или CLUSTER), принадлежащем владельцу *segment_owner*, из раздела *partition_name*. Параметры отображают количество неформатированных блоков и байт, содержащих менее 25% свободного пространства (*fs1*), от 25 до 50% свободного пространства (*fs2*), от 50 до 75% свободного пространства (*fs3*) и как минимум 75% свободного пространства (*fs4*), а также количество заполненных блоков и байт (*full*). Появилась в Oracle9*i*.

```
PROCEDURE DBMS_SPACE.UNUSED_SPACE
    (segment_owner IN VARCHAR2,
    segment_name IN VARCHAR2,
    segment_type IN VARCHAR2,
    total_blocks OUT NUMBER,
    total_bytes OUT NUMBER,
    unused_blocks OUT NUMBER,
    unused_bytes OUT NUMBER,
    last_used_extent_file_id OUT NUMBER,
    last_used_extent_block_id OUT NUMBER,
    last_used_block OUT NUMBER
    [,partition_name IN VARCHAR2 DEFAULT NULL]);
```

Возвращает идентификаторы файла и блока последнего занятого экстента, последнего занятого блока в экстенте (*last_used_extent_file_id*, *last_used_extent_block_id*

и *last_used_block*), размер сегмента (*total_blocks*, *total_bytes*) и объем свободного пространства (*unused_blocks*, *unused_bytes*) в сегменте *segment_name* типа *segment_type* (TABLE, INDEX или CLUSTER), принадлежащем владельцу *segment_owner*, из раздела *partition_name*. Необязательный параметр *partition_name* появился в Oracle8.

DBMS_SPACE_ADMIN

Предоставляет процедуры для работы с локально управляемыми табличными пространствами (*locally managed tablespaces*). Локально управляемое табличное пространство должно управляться пользователем, поэтому процедуры пакета занимаются работой, которая обычно автоматически выполняется сервером Oracle. Появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_SPACE_ADMIN.SEGMENT_VERIFY
    (tablespace_name IN VARCHAR2,
    header_relative_file IN POSITIVE,
    header_block IN POSITIVE
    verify_option IN POSITIVE DEFAULT SEGMENT_VERIFY_EXTENTS);
```

Проверяет соответствие карты экстентов сегмента с относительным номером файла заголовка сегмента *header_relative_file* и номером блока заголовка *header_block* битовой карте для табличного пространства *tablespace_name* с параметром *verify_option*, принимающим значение SEGMENT_VERIFY_EXTENTS или SEGMENT_VERIFY_EXTENTS_GLOBAL.

```
PROCEDURE DBMS_SPACE_ADMIN.SEGMENT_CORRUPT
    (tablespace_name IN VARCHAR2,
    header_relative_file IN POSITIVE,
    header_block IN POSITIVE
    corrupt_option IN POSITIVE DEFAULT SEGMENT_MARK_CORRUPT);
```

Помечает как поврежденный или неповрежденный сегмент табличного пространства *tablespace_name* с относительным номером файла заголовка сегмента *header_relative_file* и номером блока заголовка *header_block*, основываясь на параметре *corrupt_option*, который может иметь значение SEGMENT_MARK_CORRUPT или SEGMENT_MARK_VALID.

```
PROCEDURE DBMS_SPACE_ADMIN.SEGMENT_DROP_CORRUPT
    (tablespace_name IN VARCHAR2,
    header_relative_file IN POSITIVE,
    header_block IN POSITIVE);
```

Удаляет сегмент, помеченный как поврежденный в табличном пространстве *tablespace_name*, с относительным номером файла заголовка сегмента *header_relative_file* и номером блока заголовка *header_block*.

```
PROCEDURE DBMS_SPACE_ADMIN.SEGMENT_DUMP
    (tablespace_name IN VARCHAR2,
    header_relative_file IN POSITIVE,
    header_block IN POSITIVE
    dump_option IN POSITIVE DEFAULT SEGMENT_DUMP_EXTENT_MAP);
```

Создает для сегмента, расположенного в табличном пространстве *tablespace_name*, с относительным номером файла заголовка сегмента *header_relative_file* и номером блока заголовка *header_block*, дамп заголовка и карт блоков экстентов. Для

параметра *dump_option* допустимым является всего одно значение: SEGMENT_DUMP_EXTENT_MAP.

```
PROCEDURE DBMS_SPACE_ADMIN.TABLESPACE_VERIFY
   (tablespace_name IN VARCHAR2,
    verify_option IN POSITIVE := TABLESPACE_VERIFY_BITMAP);
```

Проверяет соответствие карт экстентов сегментов битовым картам в табличном пространстве *tablespace_name*. Значениями параметра *verify_option* могут быть TABLESPACE_EXTENT_MAKE_FREE и TABLESPACE_EXTENT_MAKE_USED.

```
PROCEDURE DBMS_SPACE_ADMIN.TABLESPACE_FIX_BITMAPS
   (tablespace_name IN VARCHAR2,
    dbarange_relative_file IN POSITIVE,
    dbarange_begin_block IN POSITIVE,
    dbarange_end_block IN POSITIVE,
    fix_option IN POSITIVE);
```

Помечает экстент с относительным номером файла (в диапазоне DBA) *dbarange_relative_file*, начиная с блока с номером *dbarange_begin_block* и заканчивая блоком *dbarange_end_block* в табличном пространстве *tablespace_name*, как свободный или используемый в битовой карте в зависимости от значения параметра *fix_option*: TABLESPACE_EXTENT_MAKE_FREE или TABLESPACE_EXTENT_MAKE_USED.

```
PROCEDURE DBMS_SPACE_ADMIN.TABLESPACE_REBUILD_BITMAPS
   (tablespace_name IN VARCHAR2,
    bitmap_relative_file IN POSITIVE DEFAULT NULL,
    bitmap_block IN POSITIVE DEFAULT NULL);
```

Перестраивает блок битовой карты *bitmap_block* для файла с относительным номером *bitmap_relative_file* в табличном пространстве *tablespace_name*.

```
PROCEDURE DBMS_SPACE_ADMIN.REBUILD_QUOTAS
   (tablespace_name IN VARCHAR2);
```

Перестраивает квоты для табличного пространства *tablespace_name*.

```
PROCEDURE DBMS_SPACE_ADMIN.MIGRATE_FROM_LOCAL
   (tablespace_name IN VARCHAR2);
```

Преобразует локально управляемое табличное пространство *tablespace_name* в управляемое словарем данных табличное пространство.

```
PROCEDURE DBMS_SPACE_ADMIN.MIGRATE_TO_LOCAL
   (tablespace_name IN VARCHAR,
    allocation_unit IN INTEGER,
    relative_fno IN INTEGER);
```

Преобразует табличное пространство *tablespace_name*, управляемое словарем данных, в локально управляемое с указанием *allocation_unit* и необязательного относительного номера файла *relative_fno*, в который будут помещены битовые карты. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_SPACE_ADMIN.RELOCATE_BITMAPS
   (tablespace_name IN VARCHAR,
    relative_fno IN INTEGER,
    block_number IN NUMBER);
```

Перемещает битовые карты табличного пространства *tablespace_name* в файл с относительным номером *relative_fno* в блок *block_number*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_SPACE_ADMIN.TABLESPACE_FIX_SEGMENT_STATES
    (tablespace_name IN VARCHAR);
```

Восстанавливает состояние сегментов табличного пространства *tablespace_name*, на которых произошел сбой преобразования. Появилась в Oracle9*i*.

DBMS_SQL

Предоставляет процедуры и функции для использования динамического SQL из PL/SQL. Для версий Oracle8 и выше эти процедуры обеспечивают поддержку операций с массивами в PL/SQL. В версии Oracle8*i* был введен собственный динамический SQL, который можно применять вместо описанных ниже процедур.

Вызовы

```
PROCEDURE DBMS_SQL.BIND_ARRAY
    (c IN INTEGER,
    name IN VARCHAR2,
    <table_variable IN datatype>
    [,index1 IN INTEGER
    ,index2 IN INTEGER]);
```

Связывает массив *table_variable* с заполнителем *name* в синтаксически проанализированной (но не выполненной) команде SQL в курсоре *c* (возвращаемом вызовом OPEN CURSOR). В версии Oracle8 и выше применяется для работы с массивами. В качестве *<table_variable IN datatype>* может использоваться одна из приведенных ниже конструкций:

```
n_tab IN DBMS_SQL.NUMBER_TABLE
c_tab IN DBMS_SQL.VARCHAR2_TABLE
d_tab IN DBMS_SQL.DATE_TABLE
bl_tab IN DBMS_SQL.BLOB_TABLE
cl_tab IN DBMS_SQL.CLOB_TABLE
bf_tab IN DBMS_SQL.BFILE_TABLE
```

Необязательный аргумент *index1* определяет нижнюю границу (первую строку) таблицы, а *index2* – верхнюю границу (последнюю строку). Доступны в Oracle8 и далее.

```
PROCEDURE DBMS_SQL.BIND_VARIABLE
    (c IN INTEGER,
    name IN VARCHAR2,
    value IN {NUMBER | VARACHAR2 | DATE | BLOB | CLOB
       CHARACTER SET ANY_CS | BFILE}
    [,out_value_size IN INTEGER]);
```

Связывает скалярное значение с заполнителем *name* в синтаксически проанализированной команде SQL в курсоре *c*, при этом можно дополнительно указать максимальный ожидаемый размер значения *out_value_size*. Для переменных типов CHAR, RAW и ROWID может применяться следующий синтаксис:

```
PROCEDURE DBMS_SQL.BIND_VARIABLE_CHAR
    (c IN INTEGER,
    name IN VARCHAR2,
    value IN CHAR CHARACTER SET ANY_CS
    [,out_value_size IN INTEGER]);

PROCEDURE DBMS_SQL.BIND_VARIABLE_RAW
    (c IN INTEGER,
```

```
    name IN VARCHAR2,
    value IN RAW
    [,out_value_size IN INTEGER]);

PROCEDURE DBMS_SQL.BIND_VARIABLE_ROWID
    (c IN INTEGER,
    name IN VARCHAR2,
    value IN ROWID);
```

См. выше пояснения для процедуры BIND_VARIABLE.

```
PROCEDURE DBMS_SQL.CLOSE_CURSOR
    (c IN OUT INTEGER);
```

Закрывает курсор *c*.

```
PROCEDURE DBMS_SQL.COLUMN_VALUE
    (c IN INTEGER,
    position IN INTEGER,
    value OUT {NUMBER | VARCHAR | DATE | BLOB | CLOB
      CHARACTER SET ANY_CS | BFILE | MLSLABEL }
    [,column_error OUT NUMBER
    [,actual_length OUT INTEGER]]);
```

Передает содержимое столбца с номером *position* в списке выборки извлекаемого курсора *c* в переменную *value*, при этом можно (необязательно) указать в параметре *actual_length* длину в байтах до усечения, а в *column_error* – код ошибки для указанного значения. Усечение возможно в случае, если существует отличие в размерах извлекаемого значения курсора и длины переменной. Значение MLSLABEL предназначалось для функциональности Trusted Oracle, не поддерживаемой начиная с Oracle9*i*. Процедура COLUMN_VALUE может иметь и такой синтаксис:

```
PROCEDURE DBMS_SQL.COLUMN_VALUE
    (c IN INTEGER,
    position IN INTEGER,
    <table_parameter> IN <table_type>);
```

Параметры *table_parameter* и *table_type* могут иметь следующие значения:

```
    n_tab IN DBMS_SQL.NUMBER_TABLE
    c_tab IN DBMS_SQL.VARCHAR2_TABLE
    d_tab IN DBMS_SQL.DATE_TABLE
    bl_tab IN DBMS_SQL.BLOB_TABLE
    cl_tab IN DBMS_SQL.CLOB_TABLE
    bf_tab IN DBMS_SQL.BFILE_TABLE
```

Для типов данных CHAR, LONG, RAW и ROWID процедура может иметь следующий формат:

```
PROCEDURE DBMS_SQL.COLUMN_VALUE_CHAR
    (c IN INTEGER,
    position IN INTEGER,
    value OUT CHAR CHARACTER SET ANY_CS
    [,column_error OUT NUMBER
    [,actual_length OUT INTEGER]]);.

PROCEDURE DBMS_SQL.COLUMN_VALUE_LONG
    (c IN INTEGER,
    position IN INTEGER,
    length IN INTEGER,
```

```
    offset IN INTEGER,
    value OUT VARCHAR2,
    value_length OUT INTEGER);

PROCEDURE DBMS_SQL.COLUMN_VALUE_RAW
    (c IN INTEGER,
    position IN INTEGER,
    value OUT RAW
    [,column_error OUT NUMBER
    [,actual_length OUT INTEGER]]);

PROCEDURE DBMS_SQL.COLUMN_VALUE_ROWID
    (c IN INTEGER,
    position IN INTEGER,
    value OUT ROWID);
    [,column_error OUT NUMBER
    [,actual_length OUT INTEGER]]);
```

Сведения об этих процедурах приведены в описании процедуры COLUMN_VALUE.

```
PROCEDURE DBMS_SQL.DEFINE_ARRAY
    (c IN INTEGER,
    position IN INTEGER,
    <table_parameter IN table_type>,
    cnt IN INTEGER,
    lower_bound IN INTEGER);
```

Определяет тип данных и размер элементов массива выборки для столбца с номером *position* в списке SELECT курсора *c* как совпадающий с типом данных и размером вложенной таблицы *table_parameter*, начиная со строки *lower_bound* для не более чем *cnt* строк. Для версий Oracle8 и выше параметры *table_parameter* и *table_type* могут иметь следующие значения:

```
    n_tab IN DBMS_SQL.NUMBER_TABLE
    c_tab IN DBMS_SQL.VARCHAR2_TABLE
    d_tab IN DBMS_SQL.DATE_TABLE
    bl_tab IN DBMS_SQL.BLOB_TABLE
    cl_tab IN DBMS_SQL.CLOB_TABLE
    bf_tab IN DBMS_SQL.BFILE_TABLE
```

```
PROCEDURE DBMS_SQL.DEFINE_COLUMN
    (c IN INTEGER,
    position IN INTEGER,
    column IN NUMBER | DATE | BLOB | CLOB CHARACTER SET
      ANY_CS | BFILE | MLSLABEL);
```

Переменная или выражение *column* определяет тип данных столбца с номером *position* в списке выборки курсора *c*. Значение MLSLABEL предназначалось для функциональности Trusted Oracle, не поддерживаемой начиная с Oracle9*i*. Процедура COLUMN_VALUE может иметь и такой синтаксис:

```
PROCEDURE DBMS_SQL.DEFINE_COLUMN
    (c IN INTEGER,
    position IN INTEGER,
    column IN VARCHAR2 CHARACTER SET ANY_CS,
    column_size IN INTEGER);
```

Для типов данных CHAR, LONG, RAW и ROWID процедура может иметь следующий формат:

```
PROCEDURE DBMS_SQL.DEFINE_COLUMN_CHAR
    (c IN INTEGER,
    position IN INTEGER,
    column IN CHAR CHARACTER SET ANY_CS,
    column_size IN INTEGER);

PROCEDURE DBMS_SQL.DEFINE_COLUMN_LONG
    (c IN INTEGER,
    position IN INTEGER);

PROCEDURE DBMS_SQL.DEFINE_COLUMN_RAW
    (c IN INTEGER,
    position IN INTEGER,
    column IN RAW,
    column_size IN INTEGER);

PROCEDURE DBMS_SQL.DEFINE_COLUMN_ROWID
    (c IN INTEGER,
    position IN INTEGER,
    column IN ROWID);
```

Сведения об этих процедурах приведены в описании процедуры DEFINE_COLUMN.

```
PROCEDURE DBMS_SQL.DESCRIBE_COLUMNS
    (c IN INTEGER,
    col_cnt OUT INTEGER,
    desc_t OUT DESC_TAB);
```

Наполняет PL/SQL-таблицу *desc_t* типа DBMS_SQL.DESC_REC описанием столбцов курсора *c*. Параметр *col_cnt* определяет количество столбцов *c* и количество строк в *desc_t*.

```
FUNCTION DBMS_SQL.EXECUTE
    (c IN INTEGER)
    RETURN INTEGER;
```

Для команд INSERT, UPDATE или DELETE возвращает количество строк, обработанных при выполнении курсора *c*. Для всех остальных команд SQL выполняет курсор *c* и возвращает неопределенное значение.

```
FUNCTION DBMS_SQL.EXECUTE_AND_FETCH
    (c IN INTEGER,
    exact IN BOOLEAN DEFAULT FALSE)
    RETURN INTEGER;
```

Возвращает количество строк, полученных при выполнении и извлечении курсора *c*. Если значение параметра *exact* равно TRUE, процедура порождает исключение, если извлекается более одной строки. Для нескольких строк требуется версия Oracle8 или выше и поддержка работы с массивами.

```
FUNCTION DBMS_SQL.FETCH_ROWS
    (c IN INTEGER)
    RETURN INTEGER;
```

Возвращает количество строк, извлеченных из курсора *c* (или 0, когда больше не остается строк для извлечения).

```
FUNCTION DBMS_SQL.IS_OPEN
    (c IN INTEGER)
    RETURN BOOLEAN;
```

Возвращает TRUE, если курсор *c* открыт, и FALSE – в противном случае.

```
FUNCTION DBMS_SQL.LAST_ERROR_POSITION
    RETURN INTEGER;
```

Возвращает смещение в байтах для команды SQL, определяющее байт, в котором произошла последняя ошибка. Должна вызываться сразу же после EXECUTE и EXECUTE_AND_FETCH.

```
FUNCTION DBMS_SQL.LAST_ROW_COUNT
    RETURN INTEGER;
```

Возвращает общее количество строк, извлеченных к данному моменту (подобно атрибуту %ROWCOUNT для статических курсоров).

```
FUNCTION DBMS_SQL.LAST_ROW_ID
    RETURN ROWID;
```

Возвращает идентификатор ROWID для последней извлеченной строки. Должна вызываться сразу же после FETCH_ROWS и EXECUTE_AND_FETCH.

```
FUNCTION DBMS_SQL.LAST_SQL_FUNCTION_CODE
    RETURN INTEGER;
```

Возвращает код SQL-функции для команды SQL. Полный список кодов функций приведен в справочном руководстве по работе с сервером корпорации Oracle (Oracle Corporation's Server Reference Manual) в разделе, описывающем столбец V$SESSION.COMMAND.

```
FUNCTION DBMS_SQL.OPEN_CURSOR
    RETURN INTEGER;
```

Возвращает указатель (типа INTEGER) на область памяти, выделенную для динамического курсора.

```
PROCEDURE DBMS_SQL.PARSE
    (c IN INTEGER,
    statement IN VARCHAR2,
    language_flag IN INTEGER);
```

Выполняет синтаксический анализ команды SQL, длина которой не превышает 32 Кбайт, и сопоставляет ей курсор *c*, следуя правилам, определяемым параметром *language_flag* (значениями которого могут быть DBMS_SQL.NATIVE, DBMS_SQL.V7 или DBMS_SQL.V6). Завершать строку SQL точкой с запятой следует лишь в том случае, если речь идет о блоке PL/SQL. Для команд DDL (например, TRUNCATE TABLE) также происходит выполнение команды. Для больших команд SQL формат вызова процедуры может быть таким:

```
PROCEDURE DBMS_SQL.PARSE
    (c IN INTEGER,
    statement IN VARCHAR2S,
    lb IN INTEGER,
    ub IN INTEGER,
    lfflg IN BOOLEAN,
    language_flag IN INTEGER);
```

Выполняет синтаксический анализ команды SQL, которая содержится в строках с *lb* по *ub* таблицы PL/SQL, и сопоставляет ей курсор *c*, следуя правилам, определяемым параметром *language_flag* (значениями которого могут быть DBMS_SQL.NATIVE, DBMS_SQL.V7 или DBMS_SQL.V6). Если значение параметра *lfflg* равно TRUE, то после каждой строки команды добавляется символ перевода строки.

```
PROCEDURE DBMS_SQL.VARIABLE_VALUE
    (c IN INTEGER,
```

```
    name IN VARCHAR2,
    value OUT {NUMBER | VARCHAR2 | DATE | BLOB | CLOB
      CHARACTER SET ANY_CS| BFILE | MLSLABEL});
```

Извлекает значение переменной *name* курсора *c* в переменную PL/SQL *value*. Значение MLSLABEL предназначалось для функциональности Trusted Oracle, не поддерживаемой начиная с Oracle9*i*. Процедура VARIABLE_VALUE может иметь и такой синтаксис:

```
PROCEDURE DBMS_SQL.VARIABLE_VALUE
    (c IN INTEGER,
    name IN VARCHAR2,
    value IN <table_type>);
```

Извлекает значения переменной *name* курсора *c* в PL/SQL-таблицу *value*. Для версий Oracle8 и выше параметр *table_type* может иметь одно из следующих значений:

```
DBMS_SQL.NUMBER_TABLE
DBMS_SQL.VARCHAR2_TABLE
DBMS_SQL.DATE_TABLE
DBMS_SQL.BLOB_TABLE
DBMS_SQL.CLOB_TABLE
DBMS_SQL.BFILE_TABLE
```

Для типов данных CHAR, RAW и ROWID процедура может иметь следующий формат:

```
PROCEDURE DBMS_SQL.VARIABLE_VALUE_CHAR
    (c IN INTEGER,
    name IN VARCHAR2,
    value OUT CHAR CHARACTER SET ANY_CS);

PROCEDURE DBMS_SQL.VARIABLE_VALUE_RAW
    (c IN INTEGER,
    name IN VARCHAR2,
    value OUT RAW);

PROCEDURE DBMS_SQL.VARIABLE_VALUE_ROWID
    (c IN INTEGER,name IN VARCHAR2,
    value OUT ROWID);
```

Сведения об этих процедурах приведены в описании процедуры VARIABLE_VALUE.

DBMS_STATS

Содержит процедуры для сбора статистики, необходимой оптимизатору по стоимости. Появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_STATS.PREPARE_COLUMN_VALUES[_NVARCHAR | _ROWID]
    (srec IN OUT STATREC,
    values IN values_type
    [,rwmin IN ROWID,
    rwmax IN ROWID]);
    [,rwmin IN ROWID,
    rwmax IN ROWID]);
```

STATREC – это запись, имеющая такую структуру:

```
    (epc NUMBER,
    minval RAW(2000),
    maxval RAW(2000),
    bkvals NUMARRAY,
    novals NUMARRAY)
```

Параметр *srec* хранит количество значений *values* в значениях столбца *epc* записи и количество вхождений каждого из значений *values* в *bkvals*. Для параметра *values* поддерживаются следующие типы *values_types*: CHARARRAY, DATEARRAY, NUMARRAY или RAWARRAY. Для PREPARE_COLUMNS_NVARCHAR следует задавать первый набор параметров *rw*. Для PREPARE_COLUMNS_ROWID – второй набор параметров *rw*.

```
PROCEDURE DBMS_STATS.SET_COLUMN_STATS
    (ownname IN VARCHAR2,
    tabname IN VARCHAR2,
    colname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    distcnt IN NUMBER DEFAULT NULL,
    density IN NUMBER DEFAULT NULL,
    nullcnt IN NUMBER DEFAULT NULL,
    srec IN STATREC DEFAULT NULL,
    avgclen IN NUMBER DEFAULT NULL,
    [flags IN NUMBER DEFAULT NULL,]
    statown IN VARCHAR2 DEFAULT NULL
    [, no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Эта версия применяется для стандартной статистики. Задает статистику с необязательным идентификатором *statid* для столбца *ownname.tabname.colname* в разделе *partname*, хранящуюся в таблице *statown.stattab*. (Если параметр *stattab* содержит NULL, то статистика хранится в словаре данных). Задает плотность *density*, количество неопределенных значений (*nullcnt*) и среднюю длину столбца (*avgclen*). Если значение параметра *no_invalidate* равно TRUE, то зависимые курсоры не становятся недействительными. Параметр *srec* – это запись STATREC, заполненная при помощи PREPARE_COLUMN_VALUES или GET_COLUMN_STATS. Параметр *flags* использовался для внутренних целей и не поддерживается в Oracle9*i*. Параметр *no_invalidate* появился в Oracle9*i*.

Для пользовательской статистики, появившейся в версии Oracle9*i*, вызов принимает такую форму:

```
PROCEDURE DBMS_STATS.SET_COLUMN_STATS
    (ownname IN VARCHAR2,
    tabname IN VARCHAR2,
    colname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    ext_stats IN RAW,
    stattypown IN VARCHAR2 DEFAULT NULL,
    stattypname IN VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL,
    no_invalidate IN BOOLEAN DEFAULT FALSE);
```

где *ext_stats* – это пользовательская статистика для статистики *stattypname* из схемы *stattypown*.

```
PROCEDURE DBMS_STATS.SET_INDEX_STATS
    (ownname IN VARCHAR2,
    indname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    numrows IN NUMBER DEFAULT NULL,
    numlblks IN NUMBER DEFAULT NULL,
    numdist IN NUMBER DEFAULT NULL,
    avglblk IN NUMBER DEFAULT NULL,
    avgdblk IN NUMBER DEFAULT NULL,
    clstfct IN NUMBER DEFAULT NULL,
    indlevel IN NUMBER DEFAULT NULL,
    flags IN NUMBER DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Эта версия применяется для стандартной статистики для индексов. Задает статистику с необязательным идентификатором *statid* для индекса *ownname.indname* в разделе *partname*, хранящуюся в таблице *statown.stattab*. (Если параметр *stattab* содержит NULL, то статистика хранится в словаре данных). Указывает количество строк индекса (*numrows*), количество листовых блоков (leaf blocks) в индексе (*numlblks*), количество различных ключей индекса (*numdist*), среднее количество листовых блоков, в которых появляется каждый неповторяющийся ключ (*avglblk*), среднее количество блоков данных в таблице, на которую указывает один из неповторяющихся ключей индекса (*indlevel*). Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

Для пользовательской статистики, которая появилась в Oracle9*i*, вызов принимает такую форму.

```
PROCEDURE DBMS_STATS.SET_INDEX_STATS
    (ownname IN VARCHAR2,
    indname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    ext_stats IN RAW,
    stattypown VARCHAR2 DEFAULT NULL,
    stattypname VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL,
    no_invalidate IN BOOLEAN DEFAULT FALSE);
```

где *ext_stats* – это пользовательская статистика для статистики *stattypname* из схемы *stattypown*.

```
PROCEDURE DBMS_STATS.SET_SYSTEM_STATS
    (pname IN VARCHAR2,
    pvalue IN NUMBER,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL);
```

Устанавливает хранящуюся в таблице *statown.stattab* системную статистику *pname* с необязательным идентификатором *statid* в значение *pvalue*. (Если параметр *stattab* содержит NULL, то статистика хранится в словаре данных). Значениями параметра *pname* могут быть среднее время чтения одного блока в милли-

секундах (*sreadtim*), среднее время чтения целого *mbrc*-блока в миллисекундах (*mreadtim*), средняя скорость процессора в миллионах циклов в секунду (*cpuspeed*), среднее количество считанных блоков при многоблочном считывании (*mbrc*), максимальная пропускная способность системы ввода/вывода в байтах в секунду (*maxthr*) и средняя пропускная способность подчиненной системы ввода/вывода в байтах в секунду (*slavethr*). Появилась в Oracle9*i*.

```
PROCEDURE DBMS_STATS.SET_TABLE_STATS
    (ownname IN VARCHAR2,
    indname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    numrows IN NUMBER DEFAULT NULL,
    numlblks IN NUMBER DEFAULT NULL,
    avgrlen IN NUMBER DEFAULT NULL,
    flags IN NUMBER DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Задает статистику с необязательным идентификатором *statid* для индекса *ownname.indname* в разделе *partname*, хранящуюся в таблице *statown.stattab*. (Если параметр *stattab* содержит NULL, то статистика хранится в словаре данных). Задает количество строк в таблице (*numrows*), количество блоков в таблице (*numlblks*) и среднюю длину строки таблицы (*avgrlen*). Параметр *flags* предназначен для внутреннего использования сервером Oracle. Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.CONVERT_RAW_VALUE[_NVARCHAR | _ROWID]
    (rawval IN RAW,
    resval OUT {VARCHAR2 | DATE | NUMBER | NVARCHAR2 | ROWID});
```

Преобразует минимальное или максимальное значение *rawval* в тип данных *resval*. Для NVARCHAR2 и ROWID имя типа данных добавляется в конец имени процедуры.

```
PROCEDURE DBMS_STATS.GET_COLUMN_STATS
    (ownname IN VARCHAR2,
    tabname IN VARCHAR2,
    colname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    distcnt OUT NUMBER DEFAULT NULL,
    density OUT NUMBER DEFAULT NULL,
    nullcnt OUT NUMBER DEFAULT NULL,
    srec OUT STATREC DEFAULT NULL,
    avgclen OUT NUMBER DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL);
```

Эта версия применяется для получения стандартной статистики для индексов. Получает статистику с необязательным идентификатором *statid* для столбца *ownname.tabname.colname* в разделе *partname* из таблицы *statown.stattab*. (Если параметр *stattab* содержит NULL, то статистика берется из словаря данных). Возвращает плотность *density*, количество неопределенных значений (*nullcnt*) и среднюю длину столбца (*avgclen*).

Для пользовательской статистики, которая появилась в Oracle9*i*, вызов принимает такую форму.

```
PROCEDURE DBMS_STATS.GET_COLUMN_STATS
    (ownname IN VARCHAR2,
    tabname IN VARCHAR2,
    colname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    ext_stats OUT RAW,
    stattypown OUT VARCHAR2 DEFAULT NULL,
    stattypname OUT VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL);
```

где *ext_stats* – это пользовательская статистика для статистики *stattypname* из схемы *stattypown*.

```
PROCEDURE DBMS_STATS.GET_INDEX_STATS
    (ownname IN VARCHAR2,
    indname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    numrows OUT NUMBER,
    numlblks OUT NUMBER,
    numdist OUT NUMBER,
    avglblk OUT NUMBER,
    avgdblk OUT NUMBER,
    clstfct OUT NUMBER,
    indlevel OUT NUMBER,
    [flags IN NUMBER DEFAULT NULL,]
    statown IN VARCHAR2 DEFAULT NULL
    [guessq OUT NUMBER]#);
```

Эта версия применяется для получения стандартной статистики для индексов. Получает статистику с необязательным идентификатором *statid* для индекса *ownname.indname* в разделе *partname* из таблицы *statown.stattab*. (Если параметр *stattab* содержит NULL, то статистика берется из словаря данных). Возвращает количество строк индекса (*numrows*), количество листовых блоков в индексе (*numlblks*), количество различных ключей индекса (*numdist*), среднее количество листовых блоков, в которых появляется каждый неповторяющийся ключ (*avglblk*), среднее количество блоков данных в таблице, на которую указывает один из неповторяющихся ключей индекса (*indlevel*), степень кластеризации (clustering factor) (*clstfct*), высоту индекса (*indlevel*) и предполагаемое качество (guess quality) для индекса (*guessq*). Появилась в Oracle9*i*.

Для пользовательской статистики, которая появилась в Oracle9*i*, вызов принимает такую форму:

```
PROCEDURE DBMS_STATS.GET_INDEX_STATS
    (ownname IN VARCHAR2,
    indname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    ext_stats OUT RAW,
    stattypown OUT VARCHAR2 DEFAULT NULL,
```

```
    stattypname OUT VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL);
```

где *ext_stats* – это пользовательская статистика для статистики *stattypname* из схемы *stattypown*.

```
PROCEDURE DBMS_STATS.GET_SYSTEM_STATS
    (status OUT VARCHAR2,
    dstart OUT DATE,
    dstop OUT DATE,
    pname IN VARCHAR2,
    pvalue OUT NUMBER,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL);
```

Получает системную статистику *pname* с необязательным идентификатором *statid* в переменную *pvalue* из таблицы *statown.stattab*. Если параметр *stattab* содержит NULL, то статистика хранится в словаре данных. Параметр *status* может принимать значения COMPLETED, AUTOGATHERING, BADSTATS или MANUALGATHERING. Если значение *status* равно MANUALGATHERING, то *dstart* и *dstop* – это даты начала и окончания периода ручного сбора статистики. Значениями параметра *pname* могут быть среднее время чтения одного блока в миллисекундах (*sreadtim*), среднее время чтения целого *mbrc*-блока в миллисекундах (*mreadtim*), средняя скорость процессора в миллионах циклов в секунду (*cpuspeed*), среднее количество считанных блоков при многоблочном считывании (*mbrc*), максимальная пропускная способность системы ввода/вывода в байтах в секунду (*maxthr*) и средняя пропускная способность подчиненной системы ввода-вывода в байтах в секунду (*slavethr*). Появилась в Oracle9*i*.

```
PROCEDURE DBMS_STATS.GET_TABLE_STATS
    (ownname IN VARCHAR2,
    indname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    numrows OUT NUMBER DEFAULT NULL,
    numlblks OUT NUMBER DEFAULT NULL,
    avgrlen OUT NUMBER DEFAULT NULL,
    statown VARCHAR2 DEFAULT NULL);
```

Получает статистику с необязательным идентификатором *statid* для индекса *ownname.indname* в разделе *partname* из таблицы *statown.stattab*. (Если параметр *stattab* содержит NULL, то статистика берется из словаря данных). Возвращает количество строк в таблице *(numrows*), количество блоков в таблице (*numlblks*) и среднюю длину строки таблицы (*avgrlen*).

```
PROCEDURE DBMS_STATS.DELETE_COLUMN_STATS
    (ownname IN VARCHAR2,
    tabname IN VARCHAR2,
    colname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    cascade_parts IN NUMBER DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Удаляет статистику с необязательным идентификатором *statid* для столбца *ownname.tabname.colname* в разделе *partname* из таблицы *statown.stattab*. Если параметр *stattab* содержит NULL, то статистика удаляется из словаря данных. Если таблица секционирована и параметр *partname* содержит NULL, то установка параметра *cascade_parts* приводит к удалению базовых разделов. Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.DELETE_INDEX_STATS
    (ownname IN VARCHAR2,
    indname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    cascade_parts IN NUMBER DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Удаляет статистику с необязательным идентификатором *statid* для индекса *ownname.indname* в разделе *partname* из таблицы *statown.stattab*. Если параметр *stattab* содержит NULL, то статистика удаляется из словаря данных. Если таблица секционирована и параметр *partname* содержит NULL, то установка параметра *cascade_parts* приводит к удалению базовых разделов. Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.DELETE_SYSTEM_STATS
    (stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL);
```

Удаляет системную статистику с необязательным идентификатором *statid* из таблицы *statown.stattab*. (Если параметр *stattab* содержит NULL, то статистика удаляется из словаря данных). Появилась в Oracle9*i*.

```
PROCEDURE DBMS_STATS.DELETE_TABLE_STATS
    (ownname IN VARCHAR2,
    tabname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    cascade_parts IN BOOLEAN DEFAULT TRUE,
    cascade_columns IN BOOLEAN DEFAULT TRUE,
    cascade_indexes IN BOOLEAN DEFAULT TRUE,
    statown VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Удаляет статистику с необязательным идентификатором *statid* для таблицы *ownname.tabname* в разделе *partname* из таблицы *statown.stattab*. Если параметр *stattab* содержит NULL, то статистика хранится в словаре данных. Если таблица секционирована и параметр *partname* содержит NULL, то установка параметра *cascade_parts* приводит к удалению базовых разделов. Если значение параметра *cascade_columns* или *cascade_indexes* равно TRUE, то вызывается процедура DELETE_COLUMN_STATS или DELETE_INDEX_STATS соответственно с параметром *cascade_parts*. Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.DELETE_SCHEMA_STATS
    (ownname IN VARCHAR2,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    statown VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Удаляет статистику с необязательным идентификатором *statid* для схемы *ownname* из таблицы *statown.stattab*. (Если параметр *stattab* содержит NULL, то статистика удаляется из словаря данных). Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.DELETE_DATABASE_STATS
    (stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    statown VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Удаляет статистику для базы данных с необязательным идентификатором *statid* из таблицы *statown.stattab*. (Если параметр *stattab* содержит NULL, то статистика хранится в словаре данных). Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.CREATE_STAT_TABLE
    (ownname IN VARCHAR2,
    stattab IN VARCHAR2,
    tblspace IN VARCHAR2 DEFAULT NULL);
```

Создает в схеме *ownname* таблицу *stattab* для статистики. Если табличное пространство *tblspace* не указано, то применяется схема *ownname*.

```
PROCEDURE DBMS_STATS.DROP_STAT_TABLE
    (ownname IN VARCHAR2,
    stattab IN VARCHAR2);
```

Удаляет таблицу статистики *stattab* из схемы *ownname*.

```
PROCEDURE DBMS_STATS.EXPORT_COLUMN_STATS
    (ownname IN VARCHAR2,
    tabname IN VARCHAR2,
    colname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2,
    statid IN VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL);
```

Экспортирует статистику с необязательным идентификатором *statid* для столбца *ownname.tabname.colname* в разделе *partname* в таблицу *statown.stattab*.

```
PROCEDURE DBMS_STATS.EXPORT_INDEX_STATS
    (ownname IN VARCHAR2,
    indname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2,
    statid IN VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL);
```

Экспортирует статистику с необязательным идентификатором *statid* для индекса *ownname.indname* в разделе *partname* в таблицу *statown.stattab*. Если параметр *statown* содержит NULL, то статистика экспортируется в схему пользователя.

```
PROCEDURE DBMS_STATS.EXPORTS_SYSTEM_STATS
    (stattab IN VARCHAR2,
    statid IN VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL);
```

Экспортирует системную статистику с необязательным идентификатором *statid* и сохраняет ее в таблице *statown.stattab*. Если параметр *statown* содержит NULL, то статистика экспортируется в схему пользователя. Эта процедура в версии Oracle8*i* называлась EXPORT_DATABASE_STATS.

```
PROCEDURE DBMS_STATS.EXPORT_TABLE_STATS
    (ownname IN VARCHAR2,
    tabname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2,
    statid IN VARCHAR2 DEFAULT NULL,
    cascade IN BOOLEAN DEFAULT TRUE,
    statown VARCHAR2 DEFAULT NULL);
```

Экспортирует статистику с необязательным идентификатором *statid* для таблицы *ownname.tabname* в разделе *partname* в таблицу *statown.stattab*. Если значение параметра *cascade* равно TRUE, то экспортируются и статистики для столбцов и индексов.

```
PROCEDURE DBMS_STATS.EXPORT_SCHEMA_STATS
    (ownname IN VARCHAR2,
    stattab IN VARCHAR2,
    statid IN VARCHAR2 DEFAULT NULL,
    statown VARCHAR2 DEFAULT NULL);
```

Экспортирует статистику с необязательным идентификатором *statid* для схемы *ownname* в таблицу *statown.stattab*.

```
PROCEDURE DBMS_STATS.EXPORT_DATABASE_STATS
    (stattab IN VARCHAR2,
    statid IN VARCHAR2 DEFAULT NULL,
    statown VARCHAR2 DEFAULT NULL);
```

Экспортирует статистику с необязательным идентификатором *statid* базы данных в таблицу *statown.stattab*. Если параметр *statown* содержит NULL, то статистика экспортируется в схему пользователя.

```
PROCEDURE DBMS_STATS.IMPORT_COLUMN_STATS
    (ownname IN VARCHAR2,
    tabname IN VARCHAR2,
    colname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2,
    statid IN VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Импортирует статистику с необязательным идентификатором *statid* для столбца *ownname.tabname.colname* в разделе *partname* из таблицы *statown.stattab*. Если параметр *statown* содержит NULL, то статистика импортируется из схемы *ownname*. Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.IMPORT_INDEX_STATS
    (ownname IN VARCHAR2,
```

```
    indname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2,
    statid IN VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Импортирует статистику с необязательным идентификатором *statid* для индекса *ownname.indname* в разделе *partname* из таблицы *statown.stattab*. Если параметр *statown* содержит NULL, то статистика хранится в схеме *ownname*. Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.IMPORT_SYSTEM_STATS
    (stattab IN VARCHAR2,
    statid IN VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL);
```

Импортирует системную статистику с необязательным идентификатором *statid* из таблицы *statown.stattab*.

```
PROCEDURE DBMS_STATS.IMPORT_TABLE_STATS
    (ownname IN VARCHAR2,
    tabname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    stattab IN VARCHAR2,
    statid IN VARCHAR2 DEFAULT NULL,
    cascade IN BOOLEAN DEFAULT TRUE,
    statown VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Импортирует статистику с необязательным идентификатором *statid* для таблицы *ownname.tabname* в разделе *partname* из таблицы *statown.stattab*. Если значение параметра *cascade* равно TRUE, то импортируются статистики для столбцов и индексов. Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.IMPORT_SCHEMA_STATS
    (ownname IN VARCHAR2,
    stattab IN VARCHAR2,
    statid IN VARCHAR2 DEFAULT NULL,
    statown VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Импортирует статистику с необязательным идентификатором *statid* для схемы *ownname* из таблицы *statown.stattab*. Если параметр *statown* содержит NULL, то статистика хранится в схеме *ownname*. Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.IMPORT_DATABASE_STATS
    (stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    statown VARCHAR2 DEFAULT NULL,
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Импортирует статистику с необязательным идентификатором *statid* для базы данных из таблицы *statown.stattab*. Если параметр *statown* содержит NULL, то статистика хранится в схеме пользователя. Если значение параметра *no_invali-*

date (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.GATHER_INDEX_STATS
    (ownname IN VARCHAR2,
    indname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    estimate_percent IN NUMBER DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    statown IN VARCHAR2 DEFAULT NULL
    [,degree IN NUMBER DEFAULT NULL]#
    [,granularity IN VARCHAR2 DEFAULT 'DEFAULT']#
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Собирает статистику с необязательным идентификатором *statid* для индекса *ownname.indname* в разделе *partname* в таблицу *statown.stattab*. Если параметр *statown* содержит NULL, то статистика хранится в *ownname*. Параметр *estimate_percent* определяет количество строк (в процентном отношении), которые будут участвовать в выборке. Если параметр *estimate_percent* содержит NULL, то статистики вычисляются. Параметр *degree* (появился в Oracle9*i*) определяет степень параллелизма. Сервер Oracle может порекомендовать, какие значения лучше всего использовать для *estimate_percent* (DBMS_STATS.AUTO_SAMPLE_SIZE) и *degree* (DBMS_STATS.DEFAULT_DEGREE). Параметр *granularity* (появился в Oracle9*i*) может иметь значение DEFAULT, SUBPARTITION, PARTITION, GLOBAL или ALL. Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.GATHER_TABLE_STATS
    (ownname IN VARCHAR2,
    tabname IN VARCHAR2,
    partname IN VARCHAR2 DEFAULT NULL,
    estimate_percent IN NUMBER BOOLEAN,
    block_sample IN BOOLEAN DEFAULT FALSE,
    method_opt IN VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',
    degree IN NUMBER DEFAULT NULL,
    granularity IN VARCHAR2 DEFAULT 'DEFAULT',
    cascade IN BOOLEAN DEFAULT FALSE,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    statown VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#);
```

Собирает статистику с необязательным идентификатором *statid* для таблицы *ownname.tabname* в разделе *partname* в таблицу *statown.stattab*. Если параметр *statown* содержит NULL, то статистика хранится в схеме *ownname*. Параметр *estimate_percent* определяет количество строк (в процентном отношении), которые будут участвовать в выборке. Если параметр *estimate_percent* содержит NULL, то статистики вычисляются. Параметр *degree* определяет степень параллелизма. Сервер Oracle может порекомендовать, какие значения лучше всего задать для *estimate_percent* (DBMS_STATS.AUTO_SAMPLE_SIZE) и *degree* (DBMS_STATS.DEFAULT_DEGREE). Если параметр *block_sample* установлен в TRUE, то процедура вместо случайной выборки строк выполняет случайную выборку блоков. Параметр *method_opt* определяет метод оптимизации (подробная информация об этом параметре приведена в документации Oracle). Параметр *granularity* может иметь значение DEFAULT, SUBPARTITION, PARTITION, GLOBAL или ALL. Если зна-

чение параметра *cascade* равно TRUE, то процедура также собирает статистику индексов для таблицы. Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными.

```
PROCEDURE DBMS_STATS.GATHER_SCHEMA_STATS
    (ownname IN VARCHAR2,
    estimate_percent IN NUMBER BOOLEAN,
    block_sample IN BOOLEAN DEFAULT FALSE,
    method_opt IN VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',
    degree IN NUMBER DEFAULT NULL,
    granularity IN VARCHAR2 DEFAULT 'DEFAULT',
    cascade IN BOOLEAN DEFAULT FALSE,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    options IN VARCHAR2 DEFAULT 'GATHER',
    [objList OUT OBJECTTAB,]#
    statown VARCHAR2 DEFAULT NULL
    [,noinvalidate IN BOOLEAN DEFAULT FALSE]
    [,gather_temp IN BOOLEAN DEFAULT FALSE]#);
```

Собирает статистику с необязательным идентификатором *statid* для схемы *ownname* в таблицу *statown.stattab*. Если параметр *statown* содержит NULL, то статистика хранится в схеме *ownname*. Параметр *estimate_percent* определяет количество строк (в процентном отношении), которое будет использоваться для выборки. Если параметр *estimate_percent* содержит NULL, то статистики вычисляются. Параметр *degree* определяет используемую степень параллелизма. Сервер Oracle может порекомендовать, какие значения лучше всего задавать для *estimate_percent* (DBMS_STATS.AUTO_SAMPLE_SIZE) и *degree* (DBMS_STATS.DEFAULT_DEGREE). Если значение параметра *block_sample* равно TRUE, то процедура вместо случайной выборки строк выполняет случайную выборку блоков. Параметр *method_opt* определяет метод оптимизации (подробная информация об этом параметре приведена в документации Oracle). Параметр *granularity* может иметь значение DEFAULT, SUBPARTITION, PARTITION, GLOBAL или ALL. Если значение параметра *cascade* равно TRUE, то процедура также собирает статистику индексов для таблицы. Параметр *options* указывает, для каких объектов следует собирать статистику, и может иметь значение GATHER, GATHER | LIST AUTO, GATHER | LIST STALE, GATHER | LIST EMPTY. Параметр *objList* (появился в Oracle 9*i*) содержит список объектов, которые оказались пустыми или устаревшими. Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными. Если параметр *gather_temp* (появился в Oracle9*i*) установлен в TRUE, то процедура собирает статистику для глобальных временных таблиц.

```
PROCEDURE DBMS_STATS.GATHER_DATABASE_STATS
    (ownname IN VARCHAR2,
    estimate_percent IN NUMBER BOOLEAN,
    block_sample IN BOOLEAN DEFAULT FALSE,
    method_opt IN VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',
    degree IN NUMBER DEFAULT NULL,
    granularity IN VARCHAR2 DEFAULT 'DEFAULT',
    cascade IN BOOLEAN DEFAULT FALSE,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    options IN VARCHAR2 DEFAULT 'GATHER',
    [objList OUT OBJECTTAB,]#
```

```
    statown VARCHAR2 DEFAULT NULL
    [,no_invalidate IN BOOLEAN DEFAULT FALSE]#
    [,gather_temp IN BOOLEAN DEFAULT FALSE]#);
```

Собирает статистику с необязательным идентификатором *statid* для базы данных в *statown.stattab*. Если параметр *statown* содержит NULL, то статистика хранится в схеме *ownname*. Параметр *estimate_percent* определяет количество строк (в процентном отношении), которые будут участвовать в выборке. Если параметр *estimate_percent* содержит NULL, то статистики вычисляются. Параметр *degree* определяет степень параллелизма. Сервер Oracle может порекомендовать, какие значения лучше всего задать для *estimate_percent* (DBMS_STATS.AUTO_SAMPLE_SIZE) и *degree* (DBMS_STATS.DEFAULT_DEGREE). Если значение параметра *block_sample* равно TRUE, то процедура вместо случайной выборки строк выполнит случайную выборку блоков. Параметр *method_opt* определяет метод оптимизации (подробная информация об этом параметре приведена в документации Oracle). Параметр *granularity* может иметь значение DEFAULT, SUBPARTITION, PARTITION, GLOBAL или ALL. Если параметр *cascade* установлен в TRUE, то процедура также собирает статистику индексов для таблицы. Параметр *options* указывает, для каких объектов следует собирать статистику, и может иметь значение GATHER, GATHER | LIST AUTO, GATHER | LIST STALE, GATHER | LIST EMPTY. Параметр *objList* (появился в Oracle 9*i*) содержит список объектов, которые оказались пустыми или устаревшими. Если значение параметра *no_invalidate* (появившегося в Oracle9*i*) равно TRUE, то зависимые курсоры не становятся недействительными. Если параметр *gather_temp* (появился в Oracle9*i*) установлен в TRUE, то процедура собирает статистику для глобальных временных таблиц.

```
PROCEDURE DBMS_STATS.GATHER_SYSTEM_STATS
    (gathering_mode IN VARCHAR2 DEFAULT 'NOWORKLOAD',
    interval IN INTEGER DEFAULT NULL,
    stattab IN VARCHAR2 DEFAULT NULL,
    statid IN VARCHAR2 DEFAULT NULL,
    statown VARCHAR2 DEFAULT NULL);
```

Собирает статистику для системы. Параметр *gathering_mode* определяет режим сбора и может иметь значение NOWORKLOAD, INTERVAL, START или STOP. Параметр *interval* задает период времени для сбора статистики и задается только в режиме INTERVAL. Статистика сохраняется в таблице *statown.stattab* с необязательным идентификатором *statid*. Если параметр *statown* содержит NULL, то статистика хранится в схеме пользователя. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_STATS.GENERATE_STATS
    (ownname IN VARCHAR2,
    objname IN VARCHAR2,
    organized IN NUMBER DEFAULT 7);
```

Собирает статистику для объекта *ownname.objname* из ранее собранной статистики для связанных с ним объектов. Параметр *organized* определяет степень упорядоченности индекса по отношению к соответствующей ему таблице (от 0 до 10). Чем меньше это число, тем выше организованность, вплоть до наличия последовательных ключей, ссылающихся на последовательные строки на диске.

```
PROCEDURE DBMS_STATS.FLUSH_SCHEMA_MONITORING_INFO
    (ownname VARCHAR2 DEFAULT NULL);
```

Подавляет мониторинг во внутренней памяти для таблиц схемы *ownname* или текущей схемы, если *ownname* содержит NULL. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_STATS.FLUSH_DATABASE_MONITORING_INFO;
```

Подавляет мониторинг во внутренней памяти для всех таблиц словаря данных. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_STATS.ALTER_SCHEMA_TABLE_MONITORING
    (ownname IN VARCHAR2 DEFAULT NULL,
    monitoring IN BOOLEAN DEFAULT TRUE);
```

В зависимости от значения параметра *monitoring* включает или выключает возможность мониторинга DML для всех таблиц схемы *ownname*. Появилась в Oracle9*i*.

```
PROCEDURE DBMS_STATS.ALTER_DATABASE_TABLE_MONITORING
    (monitoring IN BOOLEAN DEFAULT TRUE,
    sysobjs IN BOOLEAN DEFAULT FALSE);
```

В зависимости от значения параметра *monitoring* включает или выключает возможность мониторинга DML для всех таблиц базы данных. Если значение параметра *sysobjs* равно TRUE, то изменение состояния мониторинга отражается и на объектах словаря данных. Появилась в Oracle9*i*.

DBMS_STORAGE_MAP

Применялся для взаимодействия с FMON (фоновый процесс, отвечающий за обработку отображений файлов) для вызова *операций отображений* (*mapping operations*) и заполнения представлений отображения. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_STORAGE_MAP.MAP_ELEMENT
    (elemname IN VARCHAR2,
    cascade IN BOOLEAN,
    dictionary_update IN BOOLEAN DEFAULT TRUE);
```

Создает информацию для отображения элемента *elemname*. Если значение параметра *cascade* равно TRUE, то отображаются и все элементы графа DAG[1] стека ввода/вывода для *elemname*. Если параметр *dictionary_update* равен TRUE, то обновляется и информация для отображения в словаре данных.

```
PROCEDURE DBMS_STORAGE_MAP.MAP_FILE
    (filename IN VARCHAR2,
    filetype IN VARCHAR2,
    cascade IN BOOLEAN,
    max_num_fileextent IN NUMBER DEFAULT 100,
    dictionary_update IN BOOLEAN DEFAULT TRUE);
```

Создает информацию для отображения файла *filename* типа *filetype*, который может иметь значение DATAFILE, SPFILE, TEMPFILE, CONTROLFILE, LOGFILE или ARCHIVEFILE. Если значение параметра *cascade* равно TRUE, то отображаются и все элементы графа DAG стека ввода/вывода для *filename*. Параметр *max_num_fileextent* определяет максимальное количество экстентов файла, которое может быть отображено. Если параметр *dictionary_update* установлен в TRUE, то обновляется и информация для отображения в словаре данных.

---

[1] DAG (Directed Acyclic Graph) – ориентированный ациклический граф, используемый для отображения произвольных отношений между объектами или классами. – *Примеч. перев.*

```
PROCEDURE DBMS_STORAGE_MAP.MAP_OBJECTS
    (objname IN VARCHAR2,
    owner IN VARCHAR2,
    objtype IN VARCHAR2);
```

Создает информацию для отображения объекта *owner.objname* типа *objtype*.

```
PROCEDURE DBMS_STORAGE_MAP.MAP_ALL
    (max_num_fileext IN NUMBER DEFAULT 100,
    dictionary_update IN BOOLEAN DEFAULT TRUE);
```

Создает информацию для отображения всех типов файлов Oracle, за исключением журнальных. Параметр *max_num_fileextent* определяет максимальное количество экстентов файла, которое может быть отображено. Если параметр *dictionary_update* равен TRUE, то обновляется и информация для отображения в словаре данных.

```
PROCEDURE DBMS_STORAGE_MAP.DROP_ELEMENT
    (elemname IN VARCHAR2,
    cascade IN BOOLEAN,
    dictionary_update IN BOOLEAN DEFAULT TRUE);
```

Удаляет информацию для отображения элемента *elemname*. Если значение параметра *cascade* равно TRUE, то удаляются и все элементы графа DAG стека ввода/вывода для *elemname*. Если параметр *dictionary_update* равен TRUE, то обновляется и информация для отображения в словаре данных.

```
PROCEDURE DBMS_STORAGE_MAP.DROP_FILE
    (filename IN VARCHAR2,
    cascade IN BOOLEAN,
    dictionary_update IN BOOLEAN DEFAULT TRUE);
```

Удаляет информацию для отображения файла *filename*. Если значение параметра *cascade* равно TRUE, то удаляются и все элементы графа DAG стека ввода/вывода для *filename*. Если параметр *dictionary_update* равен TRUE, то обновляется и информация для отображения в словаре данных.

```
PROCEDURE DBMS_STORGE_MAP.DROP_ALL
    (dictionary_update IN BOOLEAN DEFAULT TRUE);
```

Удаляет всю информацию для отображения экземпляра. Если параметр *dictionary_update* установлен в TRUE, то обновляется информация для отображения в словаре данных.

```
PROCEDURE DBMS_STORAGE_MAP.SAVE;
```

Сохраняет в словаре данных информацию, необходимую для повторного формирования информации для отображения.

```
PROCEDURE DBMS_STORAGE_MAP.RESTORE;
```

Восстанавливает информацию для отображения из ранее сохраненного словаря данных.

```
PROCEDURE DBMS_STORAGE_MAP.LOCK_MAP;
```

Блокирует информацию для отображения в разделяемой памяти экземпляра БД.

```
PROCEDURE DBMS_STORAGE_MAP.UNLOCK_MAP;
```

Разблокирует информацию для отображения в разделяемой памяти экземпляра БД.

DBMS_STREAM_ADM

Предоставляет административный интерфейс, обеспечивающий добавление и удаление простых правил для захвата, распространения и применения к таблицам, схемам и базам данных, используемым в рамках компонента Oracle Streams. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_STREAM_ADM.ADD_GLOBAL_PROPAGATION_RULES
    (streams_name IN VARCHAR2,
    source_queue_name IN VARCHAR2,
    destination_queue_name IN VARCHAR2,
    include_dml IN BOOLEAN DEFAULT TRUE,
    include_ddl IN BOOLEAN DEFAULT FALSE,
    include_tagged_lcr IN BOOLEAN DEFAULT FALSE,
    source_database IN VARCHAR2 DEFAULT NULL,
    dml_rule_name OUT VARCHAR2,
    ddl_rule_name OUT VARCHAR2);
```

Добавляет правило в задание распространения (propagation job) *streams_name*, которое копирует все логические записи изменения (Logical Change Records – LCR) из исходной очереди *source_queue_name* БД *source_database* в очередь назначения *destination_queue_name*. Если значение параметра *include_dml* равно TRUE, то возвращается *dml_rule_name*. Если параметр *include_ddl* установлен в TRUE, то возвращается *ddl_rule_name*. Если параметр *include_tagged_lcr* установлен в TRUE, то при распространении учитываются LCR, у которых значение тега отлично от NULL.

```
PROCEDURE DBMS_STREAM_ADM.ADD_GLOBAL_RULES
    (streams_name IN VARCHAR2,
    streams_type IN VARCHAR2 DEFAULT NULL,
    queue_name IN VARCHAR2 DEFAULT 'STREAMS_QUEUE',
    include_dml IN BOOLEAN DEFAULT TRUE,
    include_ddl IN BOOLEAN DEFAULT FALSE,
    include_tagged_lcr IN BOOLEAN DEFAULT FALSE,
    source_database IN VARCHAR2 DEFAULT NULL,
    dml_rule_name OUT VARCHAR2,
    ddl_rule_name OUT VARCHAR2);
```

Добавляет правило в поток *streams_name* типа *streams_type* (CAPTURE или APPLY) для всех LCR в очереди *queue_name* из исходной БД *source_database*. (Для правил захвата изменения ставятся в очередь *queue_name*; для правил применения изменения выводятся из очереди *queue_name*.) Если параметр *include_dml* установлен в TRUE, то возвращается *dml_rule_name*. Если параметр *include_ddl* установлен в TRUE, то возвращается *ddl_rule_name*. Если параметр *include_tagged_lcr* установлен в TRUE, то при распространении учитываются LCR, у которых значение тега отлично от NULL.

```
PROCEDURE DBMS_STREAM_ADM.ADD_SCHEMA_PROPAGATION_RULES
    (schema_name IN VARCHAR2,
    streams_name IN VARCHAR2,
    source_queue_name IN VARCHAR2,
    destination_queue_name IN VARCHAR2,
    include_dml IN BOOLEAN DEFAULT TRUE,
    include_ddl IN BOOLEAN DEFAULT FALSE,
```

```
    include_tagged_lcr IN BOOLEAN DEFAULT FALSE,
    source_database IN VARCHAR2 DEFAULT NULL,
    dml_rule_name OUT VARCHAR2,
    ddl_rule_name OUT VARCHAR2);
```

Добавляет в поток *streams_name* правила распространения для всех LCR из схемы *schema_name* в исходной базе данных *source_database* в очередь назначения *destination_queue_name*. Если параметр *include_dml* установлен в TRUE, то возвращается *dml_rule_name*. Если параметр *include_ddl* установлен в TRUE, то возвращается *ddl_rule_name*. Если параметр *include_tagged_lcr* установлен в TRUE, то при распространении учитываются LCR, у которых значение тега отлично от NULL.

```
PROCEDURE DBMS_STREAM_ADM.ADD_SCHEMA_RULES
    (schema_name IN VARCHAR2,
    streams_name IN VARCHAR2,
    streams_type IN VARCHAR2 DEFAULT NULL,
    queue_name IN VARCHAR2 DEFAULT 'STREAMS_QUEUE',
    include_dml IN BOOLEAN DEFAULT TRUE,
    include_ddl IN BOOLEAN DEFAULT FALSE,
    include_tagged_lcr IN BOOLEAN DEFAULT FALSE,
    source_database IN VARCHAR2 DEFAULT NULL,
    dml_rule_name OUT VARCHAR2,
    ddl_rule_name OUT VARCHAR2);
```

Добавляет правило захвата или применения (в зависимости от значения параметра *streams_type*) в поток *streams_name* для всех LCR в очереди *queue_name* из схемы *schema_name* исходной БД *source_database*. (Для правил захвата изменения ставятся в очередь *queue_name*; для правил применения изменения выводятся из очереди *queue_name*.) Если параметр *include_dml* установлен в TRUE, то возвращается *dml_rule_name*. Если параметр *include_ddl* установлен в TRUE, то возвращается *ddl_rule_name*. Если параметр *include_tagged_lcr* установлен в TRUE, при распространении учитываются LCR, у которых значение тега отлично от NULL.

```
PROCEDURE DBMS_STREAM_ADM.ADD_SUBSET_RULES
    (table_name IN VARCHAR2,
    dml_condition IN VARCHAR2,
    streams_type IN VARCHAR2 DEFAULT 'APPLY',
    streams_name IN VARCHAR2,
    queue_name IN VARCHAR2 DEFAULT 'STREAMS_QUEUE',
    include_tagged_lcr IN BOOLEAN DEFAULT FALSE,
    source_database IN VARCHAR2 DEFAULT NULL,
    insert_rule_name OUT VARCHAR2,
    update_rule_name OUT VARCHAR2,
    delete_rule_name OUT VARCHAR2);
```

Добавляет правила применения для подмножества строк таблицы *table_name*, ограниченного условием *dml_condition*, для потока *streams_name* типа *streams_type* (CAPTURE или APPLY) в очереди *queue_name* исходной БД *source_database*. Если параметр *include_tagged_lcr* установлен в TRUE, при распространении учитываются LCR, у которых значение тега отлично от NULL. Возвращает правила для команд INSERT, UPDATE и DELETE.

```
PROCEDURE DBMS_STREAM_ADM.ADD_TABLE_PROPAGATION_RULES
    (table_name IN VARCHAR2,
    streams_name IN VARCHAR2,
    source_queue_name IN VARCHAR2,
    destination_queue_name IN VARCHAR2,
```

```
    include_dml IN BOOLEAN DEFAULT TRUE,
    include_ddl IN BOOLEAN DEFAULT FALSE,
    include_tagged_lcr IN BOOLEAN DEFAULT FALSE,
    source_database IN VARCHAR2 DEFAULT NULL,
    dml_rule_name OUT VARCHAR2,
    ddl_rule_name OUT VARCHAR2);
```

Добавляет правила распространения для строк таблицы *table_name* для потока *streams_name* из исходной очереди *source_queue_name* в очередь назначения *destination_queue_name* в БД *source_database*. Если параметр *include_dml* установлен в TRUE, то возвращается *dml_rule_name*. Если параметр *include_ddl* установлен в TRUE, то возвращается *ddl_rule_name*. Если параметр *include_tagged_lcr* установлен в TRUE, при распространении учитываются LCR, у которых отлично значение тега от NULL.

```
PROCEDURE DBMS_STREAM_ADM.ADD_TABLE_RULES
    (table_name IN VARCHAR2,
    streams_type IN VARCHAR2,
    streams_name IN VARCHAR2 DEFAULT NULL,
    queue_name IN VARCHAR2 DEFAULT 'streams_queue',
    include_dml IN BOOLEAN DEFAULT TRUE,
    include_ddl IN BOOLEAN DEFAULT FALSE,
    include_tagged_lcr IN BOOLEAN DEFAULT FALSE,
    source_database IN VARCHAR2 DEFAULT NULL,
    dml_rule_name OUT VARCHAR2,
    ddl_rule_name OUT VARCHAR2);
```

Добавляет правило захвата или применения (в зависимости от значения параметра *streams_type*) для строк таблицы *table_name* для потока *streams_name* в очереди *queue_name* исходной БД *source_database*. Если параметр *include_dml* установлен в TRUE, то возвращается *dml_rule_name*. Если параметр *include_ddl* установлен в TRUE, то возвращается *ddl_rule_name*. Если параметр *include_tagged_lcr* установлен в TRUE, при распространении учитываются LCR, у которых значение тега отлично от NULL.

```
PROCEDURE DBMS_STREAM_ADM.PURGE_SOURCE_CATALOG
    (source_database IN VARCHAR2,
    source_object_name IN VARCHAR2,
    source_object_type IN VARCHAR2);
```

Удаляет всю информацию о потоках из базы данных *source_database* для объекта *source_object_name* типа *source_object_type*.

```
PROCEDURE DBMS_STREAM_ADM.REMOVE_RULE
    (rule_name IN VARCHAR2,
    streams_type IN VARCHAR2,
    streams_name IN VARCHAR2,
    drop_unused_rule IN BOOLEAN DEFAULT TRUE);
```

Удаляет правило *rule_name* из потока *streams_name* типа *streams_type* (параметр может иметь значение CAPTURE, APPLY или PROPAGATE). Если значение параметра *drop_unused_rule* равно TRUE и правило больше не входит ни в какой набор правил, то оно удаляется из БД.

```
PROCEDURE DBMS_STREAM_ADM.SET_UP_QUEUE
    (queue_table IN VARCHAR2 DEFAULT 'streams_queue_table',
    storage_clause IN VARCHAR2 DEFAULT NULL,
    queue_name IN VARCHAR2 DEFAULT 'streams_queue',
```

```
    queue_user IN VARCHAR2 DEFAULT NULL,
    comment IN VARCHAR2 DEFAULT NULL);
```

Создает очередь *queue_name* в таблице *queue_table* при помощи любой корректной инструкции *storage_clause*, которая входит в генерируемую команду CREATE TABLE. Пользователь *queue_user* получает привилегии ENQUEUE и DEQUEUE для очереди.

DBMS_STREAMS

Предоставляет процедуры и функции, которые позволяют работать с данными из потоков Oracle Streams. Появился в Oracle9*i*.

Вызовы

```
FUNCTION DBMS_STREAMS.CONVERT_ANYDATA_TO_LCR
    (source IN SYS.ANYDATA);
    RETURN SYS.LCR$_DDL_RECORD;
```

Преобразует объект *source* типа SYS.ANYDATA в объект типа SYS.LCR$_DDL_RECORD, логическую запись изменений (LCR).

```
FUNCTION DBMS_STREAMS.CONVERT_ANYDATA_TO_LCR_ROW
    (source IN SYS.ANYDATA);
    RETURN SYS.LCR$_ROW_RECORD;
```

Преобразует объект *source* типа SYS.ANYDATA в объект типа SYS.LCR$_ROW_RECORD, логическую запись изменений.

```
FUNCTION DBMS_STREAMS.GET_INFORMATION
    (name IN VARCHAR2);
    RETURN SYS.ANYDATA;
```

Получает информацию о параметре *name*, значениями которого могут быть SENDER или CONSTRAINT_NAME.

```
FUNCTION DBMS_STREAMS.GET_TAG()
    RETURN RAW;
```

Возвращает двоичный тег для всех журнальных записей, сформированных текущим сеансом.

```
PROCEDURE DBMS_STREAMS.SET_TAG
    (tag IN RAW DEFAULT NULL);
```

Устанавливает двоичный тег для журнальных записей, впоследствии формируемых текущим сеансом.

DBMS_SYSTEM

Содержит процедуры и функции для внутренних событий трассировки (включая трассировку SQL) на уровне сеанса. Этот пакет не поддерживается начиная с версии Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_SYSTEM.READ_EV
    (iev BINARY_INTEGER,
    oev OUT BINARY_INTEGER);
```

Возвращает уровень вывода трассировочной информации для текущего сеанса для события с номером *iev* в переменную *oev*. Теперь данная функциональность включена в пакет DBMS_TRACE.

```
PROCEDURE DBMS_SYSTEM.SET_EV
   (si BINARY_INTEGER,
    se BINARY_INTEGER,
    ev BINARY_INTEGER,
    le BINARY_INTEGER,
    nm IN VARCHAR2);
```

Устанавливает уровень вывода трассировочной информации события с номером *ev* для сеанса, идентифицируемого параметрами *si* (идентификатор сеанса) и *se* (номер сеанса), в значение *le*. Переменная *nm* применяется для указания имени события. Теперь данная функциональность включена в пакет DBMS_TRACE.

```
PROCEDURE DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION
    (sid IN NUMBER,
    serial# IN NUMBER,
    sql_trace IN BOOLEAN);
```

В зависимости от значения параметра *sql_trace* (TRUE или FALSE соответственно) включает или выключает трассировку SQL для сеанса, идентифицируемого параметрами *sid* и *serial#*.

DBMS_TRACE

Содержит процедуры для управления трассировкой выполнения программ PL/SQL. Появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_TRACE.SET_PLSQL_TRACE
    (trace_level IN INTEGER);
```

Устанавливает уровень вывода трассировочной информации *trace_level* в 1 (для трассировки всех строк – TRACE_ALL_LINES) или 2 (для трассировки только строк в тех программных единицах, где это разрешено – TRACE_ENABLE_LINES).

```
PROCEDURE DBMS_TRACE.CLEAR_PLSQL_TRACE;
```

Отключает существующую трассировку PL/SQL.

```
PROCEDURE DBMS_TRACE.PLSQL_TRACE_VERSION
    (major OUT BINARY_INTEGER,
    minor OUT BINARY_INTEGER);
```

Возвращает версию текущей утилиты трассировки PL/SQL.

DBMS_TRANSACTION

Содержит процедуры и функции для управления локальными и распределенными транзакциями.

Вызовы

```
PROCEDURE DBMS_TRANSACTION.ADVISE_COMMIT;
```

Советует удаленной базе данных при возможности зафиксировать сомнительные распределенные транзакции.

```
PROCEDURE DBMS_TRANSACTION.ADVISE_NOTHING;
```

Убирает из удаленной БД советы относительно обработки сомнительных распределенных транзакций.

```
PROCEDURE DBMS_TRANSACTION.ADVISE_ROLLBACK;
```

Советует удаленной базе данных при возможности откатить сомнительные распределенные транзакции.

```
PROCEDURE DBMS_TRANSACTION.BEGIN_DISCRETE_TRANSACTION;
```

Указывает, что для текущей транзакции будет применяться дискретная обработка.

```
PROCEDURE DBMS_TRANSACTION.COMMIT;
```

Фиксирует текущую транзакцию.

```
PROCEDURE DBMS_TRANSACTION.COMMIT_COMMENT
    (cmnt IN VARCHAR2);
```

Фиксирует текущую транзакцию и отправляет *cmnt* удаленной БД как комментарий для сомнительной транзакции при использовании распределенных транзакций.

```
PROCEDURE DBMS_TRANSACTION.COMMIT_FORCE
    (xid IN VARCHAR2
    [,scn IN VARCHAR2 DEFAULT NULL]);
```

Выполняет фиксацию локальной части сомнительной распределенной транзакции, определяемой идентификатором транзакции *xid* и (необязательно) системным номером изменения *scn*.

```
FUNCTION DBMS_TRANSACTION.LOCAL_TRANSACTION_ID
    (create_transaction IN BOOLEAN := FALSE)
    RETURN VARCHAR2;
```

Возвращает уникальный идентификатор Oracle для текущей транзакции. При этом, если значение параметра *create_transaction* равно TRUE, то начинается новая транзакция.

```
PROCEDURE DBMS_TRANSACTION.PURGE_LOST_DB_ENTRY
    (xid IN VARCHAR2);
```

Заставляет сервер Oracle очистить все локальные записи для распределенной транзакции, определяемой идентификатором *xid* в случае, если участвующий в транзакции узел был утерян навсегда. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_TRANSACTION.PURGE_MIXED
    (xid IN VARCHAR2);
```

Заставляет сервер Oracle очистить локальные записи для распределенной транзакции, завершенной различными способами на разных узлах, локально определяемой идентификатором *xid*.

```
PROCEDURE DBMS_TRANSACTION.READ_ONLY;
```

Устанавливает согласованность по чтению на уровне транзакции, когда все запросы возвращают согласованные по чтению образы данных в состояние на момент начала транзакции.

```
PROCEDURE DBMS_TRANSACTION.READ_WRITE;
```

Устанавливает согласованность по чтению на уровне команды (поведение по умолчанию).

```
PROCEDURE DBMS_TRANSACTION.ROLLBACK;
```

Откатывает текущую транзакцию.

```
PROCEDURE DBMS_TRANSACTION.ROLLBACK_FORCE
    (xid IN VARCHAR2);
```

Откатывает локальную часть сомнительной распределенной транзакции, определяемой идентификатором транзакции *xid*.

```
PROCEDURE DBMS_TRANSACTION.ROLLBACK_SAVEPOINT
    (savept IN VARCHAR2);
```

Откатывает текущую транзакцию до точки сохранения *savept*.

```
PROCEDURE DBMS_TRANSACTION.SAVEPOINT
    (savept IN VARCHAR2);
```

Устанавливает точку сохранения с именем *savept* в текущей транзакции.

```
FUNCTION DBMS_TRANSACTION.STEP_ID
    RETURN NUMBER;
```

Возвращает уникальное целое положительное число, которое упорядочивает DML-операции текущей транзакции.

```
PROCEDURE DBMS_TRANSACTION.USE_ROLLBACK_SEGMENT
    (rb_name IN VARCHAR2);
```

Сопоставляет текущей транзакции сегмент отката *rb_name*.

DBMS_TRANSFORM

Предоставляет интерфейс для функций преобразования компонента Oracle Advanced Queueing. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_TRANSFORM.CREATE_TRANSFORMATION
    (schema IN VARCHAR2(30),
    name IN VARCHAR2(30),
    from_schema IN VARCHAR2(30),
    from_type IN VARCHAR2(30),
    to_schema IN VARCHAR2(30),
    to_type IN VARCHAR2(30),
    transformation IN VARCHAR(4000));
```

Создает преобразование *name* в схеме *schema*, трансформирующее объект типа *from_type* схемы *from_schema* в объект типа *to_type* схемы *to_schema* посредством выражения *transformation*.

```
PROCEDURE DBMS_TRANSFORM.MODIFY_TRANSFORMATION
    (schema IN VARCHAR2(30),
    name IN VARCHAR2(30),
    attribute_number IN INTEGER,
    transformation IN VARCHAR(4000));
```

Изменяет отображение для атрибута с номером *attribute_number* результирующего типа преобразования *name* в схеме *schema* посредством выражения *transformation*.

```
PROCEDURE DBMS_TRANSFORM.DROP_TRANSFORMATION
    (schema IN VARCHAR2(30),
    name IN VARCHAR2(30));
```

Удаляет преобразование *name* в схеме *schema*.

DBMS_TTS

Проверяет, является ли замкнутым набор табличных пространств, предназначенных для переноса. Появился в Oracle8*i*.

Вызовы

```
PROCEDURE DBMS_TTS.TRANSPORT_SET_CHECK
    (ts_list IN VARCHAR2,
    incl_constraints IN BOOLEAN DEFAULT
    [,full_closure IN BOOLEAN DEFAULT FALSE]#);
```

Проверяет содержащийся в параметре *ts_list* список имен табличных пространств. Если значение параметра *incl_constraints* равно TRUE, то при проверке замкнутости учитываются ограничения ссылочной целостности. Если параметр *full_closure* (появился в Oracle9*i*) установлен в TRUE, то при проверке учитываются все указатели IN и OUT (зависимости) и, если они не являются замкнутыми, то воспринимаются как нарушения.

```
PROCEDURE DBMS_TTS.DOWNGRADE;
```

Удаляет данные, относящиеся к переносимому табличному пространству.

DBMS_TYPES

Содержит константы, которые представляют собой встроенные и пользовательские типы данных, применяемые при работе с interMedia. Появился в Oracle9*i*.

DBMS_UTILITY

Предоставляет процедуры и функции, предназначенные для выполнения ряда полезных задач, таких как синтаксический разбор и токенизация, получение сведений о конфигурации БД, анализ объектов, получение информации об ошибках и вызовах, планирование выполнения кода.

Вызовы

```
FUNCTION DBMS_UTILITY.CURRENT_INSTANCE
    RETURN NUMBER;
```

Возвращает номер текущего подключенного экземпляра БД или NULL, если подключенный экземпляр не активен. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_UTILITY.CURRENT_INSTANCES
    (instance_table OUT INSTANCE_TABLE,
    instance_count OUT NUMBER);
```

Возвращает список активных экземпляров БД и их имен в таблицу *instance_table*, а количество активных экземпляров БД – в параметр *instance_count*. Появилась в Oracle8*i*. В версии Oracle8*i* эта процедура называлась ACTIVE_INSTANCES.

```
PROCEDURE DBMS_UTILITY.ANALYZE_DATABASE
    (method IN VARCHAR2,
    estimate_rows IN NUMBER DEFAULT NULL,
    estimate_percent IN NUMBER DEFAULT NULL,
    method_opt IN VARCHAR2 DEFAULT NULL);
```

Анализирует все таблицы, кластеры и индексы базы данных, используя параметр *method* (который может иметь значение ESTIMATE, COMPUTE или DELETE). Если *method* установлен в ESTIMATE, то для определения размера выборки необходимо задать параметр *estimate_rows* или *estimate_percent*. Дополнительные условия анализа можно указать при помощи параметра *method_opt*, значениями которого могут быть FOR TABLE, FOR ALL COLUMNS [SIZE N], FOR ALL INDEXED COLUMNS [SIZE N] и FOR ALL INDEXES. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_UTILITY.ANALYZE_PART_OBJECT
    (schema IN VARCHAR2 DEFAULT NULL,
    object_name IN VARCHAR2 DEFAULT NULL,
    object_type IN CHAR DEFAULT 'T',
    command_type IN CHAR DEFAULT 'E',
    command_opt IN VARCHAR2 DEFAULT NULL,
    sample_clause IN VARCHAR2 DEFAULT 'SAMPLE 5 PERCENT');
```

Выполняет параллельный (по отношению к разделам) анализ объекта (секционированной таблицы или индекса) *object_name* типа *object_type* ('T' для таблицы, 'I' для индекса), который принадлежит схеме *schema*, используя процессы очереди заданий Oracle. Параметр *command_type* задает тип выполняемого анализа, а параметр *command_opt* указывает дополнительные условия анализа. Параметр *sample_clause* определяет размер выборки в случае, если значение параметра *command_type* равно 'E' (*estimate* – оценка) при помощи инструкций 'SAMPLE N ROWS' или 'SAMPLE N PERCENT'.

Для параметра *command_type* допустимы следующие значения: 'C' – для вычисления статистики, 'E' – для оценки статистики, 'D' – для удаления статистики и 'V' – для проверки корректности структуры.

Если значение *command_type* равно 'C' или 'E', то значением *command_opt* может быть FOR TABLE, FOR ALL LOCAL INDEXES, FOR ALL COLUMNS или любая комбинация опций FOR команды ANALYZE.

Если значение *command_type* равно 'V', то в случае если *object_type* содержит 'T' (объект представляет собой таблицу), *command_opt* может иметь значение CASCADE.

```
PROCEDURE DBMS_UTILITY.ANALYZE_SCHEMA
    (schema IN VARCHAR2,
    method IN VARCHAR2,
    estimate_rows IN NUMBER DEFAULT NULL,
    estimate_percent IN NUMBER DEFAULT NULL,
    method_opt IN VARCHAR2 DEFAULT NULL);
```

Анализирует все таблицы, кластеры и индексы в схеме *schema* с помощью метода *method* (ESTIMATE, COMPUTE или DELETE). Если применяется метод ESTIMATE, то необходимо указать размер выборки, задав параметр *estimate_rows* или *estimate_percent*. Параметр *method_opt* (появился в Oracle8*i*) позволяет задать дополнительные условия анализа: FOR TABLE, FOR ALL COLUMNS [SIZE N], FOR ALL INDEXED COLUMNS [SIZE N] или FOR ALL INDEXES.

```
PROCEDURE DBMS_UTILITY.COMMA_TO_TABLE
    (list IN VARCHAR2,
    tablen OUT BINARY_INTEGER,
    tab OUT UNCL_ARRAY);
```

Проводит синтаксический разбор разделенного запятыми списка и возвращает маркеры в таблицу PL/SQL *tab* типа DBMS_UTILITY.UNCL_ARRAY. Количество строк в *tab* возвращается в параметр *tablen*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_UTILITY.COMPILE_SCHEMA
    (schema IN VARCHAR2);
```

Компилирует все хранимые программы PL/SQL (процедуры, функции и пакеты), принадлежащие схеме *schema*.

```
FUNCTION DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK
    (dba IN NUMBER)
    RETURN NUMBER;
```

Возвращает *смещение блока* (*block offset*) для указанного в параметре *dba* адреса блока данных. Появилась в Oracle8*i*.

```
FUNCTION DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE
    (dba IN NUMBER)
    RETURN NUMBER;
```

Возвращает составляющую номера файла для адреса блока данных, указанного в параметре *dba*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_UTILITY.DB_VERSION
    (version OUT VARCHAR2,
    compatibility OUT VARCHAR2);
```

Возвращает версию СУБД в параметр *version* и значение параметра COMPATIBLE из файла *INIT.ORA* или *SPFILE* – в *compatibility* (или NULL). Появилась в Oracle8*i*.

```
PROCEDURE DBMS_UTILITY.EXEC_DDL_STATEMENT
    (parse_string IN VARCHAR2);
```

Выполняет команду DDL, указанную в параметре *parse_string*. Появилась в Oracle8*i*.

```
FUNCTION DBMS_UTILITY.FORMAT_ВЫЗОВ_STACK
    RETURN VARCHAR2;
```

Возвращает текущий стек вызовов PL/SQL в виде форматированной строки.

```
FUNCTION DBMS_UTILITY.FORMAT_ERROR_STACK
    RETURN VARCHAR2;
```

Возвращает текущий стек ошибок PL/SQL в виде форматированной строки.

```
FUNCTION DBMS_UTILITY.GET_HASH_VALUE
    (name IN VARCHAR2,
    base IN NUMBER,
    hash_size IN NUMBER)
    RETURN NUMBER;
```

Возвращает для строки *name* значение хеш-функции (минимально возможное значение равно *base*), используя хеш-таблицу размера *hash_size*. Появилась в Oracle8*i*.

```
FUNCTION DBMS_UTILITY.GET_PARAMETER_VALUE
    (parnam IN VARCHAR2,
    intval IN OUT BINARY_INTEGER,
    strval IN OUT VARCHAR2)
    RETURN BINARY_INTEGER;
```

Возвращает информацию о текущем значении указанного параметра инициализации БД *parnam* (из файла INIT.ORA или SPFILE). Появилась в Oracle8*i*. Параметр *ntval* возвращает следующие значения:

- Значение для числового параметра *parnam*
- Длину для строкового параметра *parnam*

- Если *parnam* – это параметр типа Boolean, то возвращается 0 для значения FALSE и 1 – для TRUE

Параметр *strval* возвращает NULL или значение строкового параметра. Функция возвращает 0 для параметра типа Boolean и числового параметра, 1 – для строкового параметра.

```
FUNCTION DBMS_UTILITY.GET_TIME
    RETURN NUMBER;
```

Возвращает число, представляющее собой количество сотых долей секунды, прошедших начиная с некоторого (неизвестного) произвольного момента времени в прошлом. Появилась в Oracle8*i*.

```
FUNCTION DBMS_UTILITY.IS_[PARALLEL_SERVER | CLUSTER_SERVER]
    RETURN BOOLEAN;
```

Возвращает TRUE, если экземпляр БД работает в режиме Parallel Server (до версии Oracle9*i*), или использует Real Application Clusters в версии Oracle9*i*; в противном случае возвращает FALSE. Появилась в Oracle8*i*.

```
FUNCTION DBMS_UTILITY.MAKE_DATA_BLOCK_ADDRESS
    (file IN NUMBER,
    block IN NUMBER)
    RETURN NUMBER;
```

Возвращает корректный адрес блока данных на основе номера файла *file* и смещения блока *block*. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_UTILITY.NAME_RESOLVE
    (name IN VARCHAR2,
    context IN NUMBER,
    schema OUT VARCHAR2,
    part1 OUT VARCHAR2,
    part2 OUT VARCHAR2,
    dblink OUT VARCHAR2,
    part1_type OUT NUMBER,
    object_number OUT NUMBER);
```

Выполняет разрешение имени для ссылки с именем *name* и возвращает идентификационную информацию для данного объекта, а именно: *schema* – владелец объекта; *part1* – имя объекта или имя пакета для пакета; *part2* – имя программы в случае, если объект является пакетом; *dblink* – связь БД, если ссылка *name* указывает на удаленный объект; *part1_type* определяет тип объекта; *object_number* содержит локальный номер объекта или NULL, если *name* невозможно полностью разрешить локально.

Параметр *part1_type* равен 5, если объект – это синоним; 7 – для процедуры; 8 – для функции или 9 для пакета.

Параметр *context* должен быть установлен в 1.

```
PROCEDURE DBMS_UTILITY.NAME_TOKENIZE
    (name IN VARCHAR2,
    a OUT VARCHAR2,
    b OUT VARCHAR2,
    c OUT VARCHAR2,
    dblink OUT VARCHAR2,
    nextpos OUT BINARY_INTEGER);
```

С помощью синтаксического анализатора PL/SQL осуществляет токенизацию ссылки *name* – она раскладывается на составляющие в следующем формате:

```
    a [ . b [ . c ] ] [ @dblink ]
```

Параметр *nextpos* – это начальная позиция следующего токена. Появилась в Oracle8*i*.

```
FUNCTION DBMS_UTILITY.PORT_STRING
    RETURN VARCHAR2;
```

Возвращает строку со специфичной для ОС идентификационной информацией для работающей версии сервера Oracle. Появилась в Oracle8*i*.

```
PROCEDURE DBMS_UTILITY.TABLE_TO_COMMA
    (tab IN UNCL_ARRAY,
    tablen OUT BINARY_INTEGER,
    list OUT VARCHAR2);
```

Преобразует PL/SQL-таблицу *tab* типа DBMS_UTILITY.UNCL_ARRAY в разделенную запятыми строку, которая возвращается в параметр *list*, при этом количество преобразованных строк возвращается в параметр *tablen*. Появилась в Oracle8*i*.

DBMS_WM

Предоставляет интерфейс для управления рабочим пространством базы данных Oracle (Database Workspace Manager). Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_WM.AlterSavepoint
    (workspace IN VARCHAR2,
    sp_name IN VARCHAR2,
    sp_description IN VARCHAR2);
```

Изменяет описание *sp_description* для точки сохранения *sp_name* в рабочем пространстве *workspace*.

```
PROCEDURE DBMS_WM.AlterWorkspace
    (workspace IN VARCHAR2,
    workspace_description IN VARCHAR2);
```

Изменяет описание *workspace_description* для рабочего пространства *workspace*.

```
PROCEDURE DBMS_WM.BeginDDL
    (table_name IN VARCHAR2);
```

Запускает сеанс DDL для таблицы *table_name* и создает таблицу с именем *table_name*_LTS.

```
PROCEDURE DBMS_WM.BeginResolve
    (workspace IN VARCHAR2);
```

Запускает сеанс разрешения конфликтов для рабочего пространства *workspace*.

```
PROCEDURE DBMS_WM.CommitDDL
    (table_name IN VARCHAR2
    [, ignore_last_error IN BOOLEAN DEFAULT FALSE]);
```

Фиксирует DDL-изменения, произведенные в рамках DDL-сеанса для таблицы *table_name*. Если параметр *ignore_last_error* установлен в TRUE, то последняя ошибка, которая могла иметь место при последнем вызове данной процедуры, игнорируется.

```
PROCEDURE DBMS_WM.CommitResolve
    (workspace IN VARCHAR2);
```

Завершает сеанс разрешения конфликтов и фиксирует все изменения, сделанные после вызова процедуры DBMS_WM.BeginResolve.

```
PROCEDURE DBMS_WM.CompressWorkspace
    (workspace IN VARCHAR2,
    [, compress_view_wo_overwrite IN BOOLEAN]
    [, firstSP IN VARCHAR2 DEFAULT NULL]
    [, secondSP IN VARCHAR2 DEFAULT NULL]
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Убирает удаляемые точки сохранения и минимизирует структуры метаданных Workspace Manager для рабочего пространства *workspace*. Если задана только первая точка сохранения *firstSP*, то процедура работает со всеми удаляемыми точками сохранения от момента создания рабочего пространства до *firstSP*. Если указаны две точки сохранения, *firstSP* и *secondSP*, то удаляются все точки сохранения, которые можно удалить и которые находятся между указанными. Если параметр *secondSP* равен LATEST, то удаляются все точки сохранения от *firstSP* до конца рабочего пространства. Если параметр *compress_view_wo_overwrite* установлен в TRUE, то удаляется историческая информация между точками *firstSP* и *secondSP*. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.CompressWorkspaceTree
    (workspace IN VARCHAR2,
    [, compress_view_wo_overwrite IN BOOLEAN]
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Убирает удаляемые точки сохранения и минимизирует структуры метаданных Workspace Manager для рабочего пространства *workspace* и всех производных от него рабочих пространств. Если параметр *compress_view_wo_overwrite* установлен в TRUE, то удаляется историческая информация. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.CopyForUpdate
    (table_name IN VARCHAR2
    [, where_clause IN VARCHAR DEFAULT '']);
```

Позволяет изменять столбцы LOB в таблице *table_name*. При наличии инструкции *where_clause* изменение разрешено только для тех строк, которые удовлетворяют указанному условию.

```
PROCEDURE DBMS_WM.CreateSavepoint
    (workspace IN VARCHAR2,
    savepoint_name IN VARCHAR2
    [, description IN VARCHAR DEFAULT NULL]
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Создает точку сохранения *savepoint_name* с описанием *description* для рабочего пространства *workspace*. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.CreateWorkspace
    (workspace IN VARCHAR2
    [, isref_refreshed IN BOOLEAN]
    [, description IN VARCHAR DEFAULT NULL]
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Создает рабочее пространство *workspace* с описанием *description*. Если параметр *isref_refreshed* установлен в TRUE, то рабочее пространство постоянно обновляется.

Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.DeleteSavepoint
    (workspace IN VARCHAR2,
    savepoint_name IN VARCHAR2
    [, compress_view_wo_overwrite IN BOOLEAN DEFAULT FALSE]
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Удаляет точку сохранения *savepoint_name* из рабочего пространства *workspace*. Если параметр *compress_view_wo_overwrite* установлен в TRUE, то удаляется историческая информация. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.DisableVersioning
    (table_name IN VARCHAR2
    [, force IN BOOLEAN DEFAULT FALSE]
    [, ignore_last_error IN BOOLEAN DEFAULT FALSE]);
```

Отменяет поддержку многоверсионности для таблицы *table_name* и удаляет структуры, обеспечивающие такую поддержку. Если параметр *force* установлен в TRUE, то удаляются все данные рабочих пространств, отличных от LIVE. Если параметр *ignore_last_error* установлен в TRUE, то последняя ошибка, которая могла иметь место при последнем вызове данной процедуры, игнорируется.

```
PROCEDURE DBMS_WM.DropReplicationSupport;
```

Удаляет объекты поддержки тиражирования, созданные процедурой GenerateReplicationSupport.

```
PROCEDURE DBMS_WM.EnableVersioning
    (table_list IN VARCHAR2
    [, hist IN VARCHAR DEFAULT 'NONE']);
```

Включает многоверсионность для таблиц, перечисленных в списке *table_list*. Параметр *hist* может иметь значения NONE, VIEW_W_OVERWRITE (отображаются только последние изменения) и VIEW_WO_OVERWRITE (отслеживается история изменений).

```
PROCEDURE DBMS_WM.FreezeWorkspace
    (workspace IN VARCHAR
    [, session_duration IN BOOLEAN]
    [, freezemode IN VARCHAR2 DEFAULT 'NO ACCESS']
    [, freezewriter IN VARCHAR2 DEFAULT NULL]
    [, force IN BOOLEAN DEFAULT FALSE]);
```

Ограничивает доступ к рабочему пространству *workspace*. Если параметр *session_duration* установлен в TRUE, то рабочее пространство автоматически «размораживается» при завершении вызывающего сеанса. Параметр *freezemode* может иметь значения NO_ACCESS, READ_ONLY, 1WRITER (запись разрешена только для пользователя *freezewriter*) или WM_ONLY (разрешены только операции диспетчера управления рабочим пространством). Если значение параметра *force* равно FALSE, то рабочее пространство будет «замораживаться», даже если оно уже «заморожено».

```
PROCEDURE DBMS_WM.GenerateReplicationSupport
    (mastersites IN VARCHAR2,
    groupname IN VARCHAR2
    [, groupdescription IN VARCHAR2 DEFAULT 'Replication Group for OWM']);
```

Создает структуры, необходимые для тиражирования из списка разделенных запятыми главных узлов тиражирования *mastersites*, входящих в группу *groupname*. Параметр *groupdescription* содержит необязательный комментарий для группы.

```
FUNCTION DBMS_WM.GetConflictWorkspace
    RETURN VARCHAR2;
```

Возвращает имя конфликтного рабочего пространства, для которого сеансом была выполнена процедура SetConflictWorkspace.

```
FUNCTION DBMS_WM.GetDiffVersions
    RETURN VARCHAR2;
```

Возвращает имена пар (рабочее пространство, точка сохранения), которые использовались для процедуры SetDiffVersions.

```
FUNCTION DBMS_WM.GetLockMode
    RETURN VARCHAR2;
```

Возвращает режим блокировки для текущего сеанса.

```
FUNCTION DBMS_WM.GetMultiWorkspaces
    RETURN VARCHAR2;
```

Возвращает имена рабочих пространств, отображаемых в представлениях со многими рабочими пространствами для таблиц с поддержкой многоверсионности строк.

```
FUNCTION DBMS_WM.GetOpContext
    RETURN VARCHAR2;
```

Возвращает контекст текущей операции.

```
FUNCTION DBMS_WM.GetPrivs
    (workspace IN VARCHAR2)
    RETURN VARCHAR2;
```

Возвращает разделенный запятыми список всех привилегий, которыми пользователь обладает в рабочем пространстве *workspace*.

```
PROCEDURE DBMS_WM.GetSessionInfo
    (workspace OUT VARCHAR2,
    context OUT VARCHAR2,
    context_type OUT VARCHAR2);
```

Возвращает контекст *context* с типом *context_type* (LATEST, SAVEPOINT или INSTANT) текущего сеанса для рабочего пространства *workspace*.

```
FUNCTION DBMS_WM.GetWorkspace
    RETURN VARCHAR2;
```

Возвращает текущее рабочее пространство для сеанса.

```
PROCEDURE DBMS_WM.GotoDate
    (in_date IN DATE);
```

Переходит к точке с указанными датой и временем *in_date* или ближайшей к указанному моменту в текущем рабочем пространстве.

```
PROCEDURE DBMS_WM.GotoSavepoint
    ([savepoint_name IN VARCHAR2 DEFAULT 'LATEST']);
```

Переходит к точке сохранения *savepoint_name*. Если имя *savepoint_name* не указано, то переходит к LATEST.

```
PROCEDURE DBMS_WM.GotoWorkspace
    (workspace IN VARCHAR2);
```

Переходит к рабочему пространству *workspace*.

```
PROCEDURE DBMS_WM.GrantSystemPriv
    (priv_types IN VARCHAR2,
    grantee IN VARCHAR2
    [, grant_option IN VARCHAR2 DEFAULT 'NO']
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Предоставляет привилегии на уровне системы (не ограниченные каким-то одним рабочим пространством) *priv_types* пользователю (или роли) *grantee*. Если параметр *grant_option* установлен в значение YES, то получивший привилегию имеет право выдавать ее. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.GrantWorkspacePriv
    (priv_types IN VARCHAR2,
    workspace IN VARCHAR2,
    grantee IN VARCHAR2
    [, grant_option IN VARCHAR2 DEFAULT 'NO']
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Предоставляет привилегии *priv_types* в рамках рабочего пространства *workspace* пользователю (или роли) *grantee*. Если параметр *grant_option* установлен в значение YES, то получивший привилегию имеет право выдавать ее. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
FUNCTION DBMS_WM.IsWorkspaceOccupied
    (workspace IN VARCHAR2)
    RETURN VARCHAR2;
```

Возвращает YES, если для рабочего пространства *workspace* существуют активные сеансы.

```
PROCEDURE DBMS_WM.LockRows
    (workspace IN VARCHAR2,
    table_name IN VARCHAR2
    [, where_clause IN VARCHAR2 DEFAULT '']
    [, lock_mode IN VARCHAR2 DEFAULT 'E']);
```

Блокирует последние версии строк таблицы *table_name* в рабочем пространстве *workspace*. Инструкция *where_clause* задает подмножество строк для блокировки и может ссылаться только на столбцы первичного ключа. Параметр *lock_mode* может иметь значение 'E' (exclusive – монопольный режим блокировки) или 'S' (shared – разделяемый режим).

```
PROCEDURE DBMS_WM.MergeTable
    (workspace IN VARCHAR2,
    table_id IN VARCHAR2
    [, where_clause IN VARCHAR2 DEFAULT '']
    [, create_savepoint IN BOOLEAN DEFAULT FALSE]
    [, remove_data IN BOOLEAN DEFAULT FALSE]
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Применяет изменения в таблице *table_name* в рабочем пространстве *workspace* к родительскому рабочему пространству. Инструкция *where_clause* задает подмножество строк для слияния с родительским рабочим пространством и может ссылаться только на столбцы первичного ключа. Если параметр *create_savepoint* установлен

в TRUE, то создается неявная точка сохранения. Если параметр *remove_data* установлен в TRUE, то данные удаляются из дочернего рабочего пространства (только в случае, если *workspace* не имеет дочерних рабочих пространств, то есть является листом). Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.MergeWorkspace
    (workspace IN VARCHAR2
    [, create_savepoint IN BOOLEAN DEFAULT FALSE]
    [, remove_workspace IN BOOLEAN DEFAULT FALSE]
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Применяет изменения в дочернем пространстве *workspace* к его родительскому пространству. Если параметр *create_savepoint* установлен в TRUE, то создается неявная точка сохранения. Если параметр *remove_workspace* установлен в TRUE, то после выполнения операции *workspace* удаляется. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.RecoverAllMigratingTables
    ([ignore_last_error IN BOOLEAN DEFAULT FALSE]);
```

Предпринимает попытку завершения миграции для всех таблиц, которые остались в несогласованном состоянии после сбоя Workspace Manager. Если параметр *ignore_last_error* установлен в TRUE, то процедура игнорирует последнюю ошибку, произошедшую в процессе миграции.

```
PROCEDURE DBMS_WM.RecoverMigratingTable
    (table_id IN VARCHAR2
    [, ignore_last_error IN BOOLEAN DEFAULT FALSE]);
```

Предпринимает попытку завершения миграции для таблицы *table_id*, оставшейся в несогласованном состоянии после сбоя Workspace Manager. Если параметр *ignore_last_error* установлен в TRUE, то процедура игнорирует последнюю ошибку, произошедшую в процессе миграции.

```
PROCEDURE DBMS_WM.RefreshTable
    (workspace IN VARCHAR2,
    table_id IN VARCHAR2
    [, where_clause IN VARCHAR2 DEFAULT ``]
    [, create_savepoint IN BOOLEAN DEFAULT FALSE]
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Применяет к таблице *table_id* в рабочем пространстве *workspace* изменения, сделанные в родительском пространстве. Инструкция *where_clause* задает подмножество строк для обновления и может ссылаться только на столбцы первичного ключа. Если параметр *create_savepoint* установлен в TRUE, то создается неявная точка сохранения. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.RefreshWorkspace
    (workspace IN VARCHAR2
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Применяет к рабочему пространству *workspace* все изменения, сделанные в родительском пространстве. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.RelocateWriterSite
    (newwritersite IN VARCHAR2,
    oldwritersiteavailable IN BOOLEAN);
```

Делает непишущее рабочее пространство *newwritersite* новым пишущим узлом в среде тиражирования Workspace Manager. Если параметр *oldwritersiteavailable* установлен в TRUE, то старый пишущий узел обновляется для того, чтобы показать, что пишущий узел изменился. В противном случае необходимо, когда старый пишущий узел станет доступен, выполнить процедуру SynchronizeSite.

```
PROCEDURE DBMS_WM.RemoveWorkspace
    (workspace IN VARCHAR2
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Очищает все версии строк, сопоставленные рабочему пространству *workspace*, и удаляет *workspace*. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.RemoveWorkspaceTree
    (workspace IN VARCHAR2
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Очищает все версии строк, сопоставленные рабочему пространству *workspace* и его потомкам, и удаляет соответствующие рабочие пространства. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.ResolveConflicts
    (workspace IN VARCHAR2,
    table_name IN VARCHAR2,
    where_clause IN VARCHAR2,
    keep IN VARCHAR2);
```

Проверяет таблицу *table_name* в рабочем пространстве *workspace* на предмет конфликтов с другими рабочими пространствами. Инструкция *where_clause* задает подмножество строк для обновления из родительского рабочего пространства и может ссылаться только на столбцы первичного ключа. Параметр *keep* указывает, каким образом следует разрешать конфликт; он может иметь значение PARENT (строки родительского рабочего пространства будут скопированы в дочернее), CHILD (конфликт считается разрешенным, и при выполнении операции слияния для дочернего пространства дочерние строки копируются в родительские) или BASE (базовые строки копируются в дочернее, а не в родительское рабочее пространство).

```
PROCEDURE DBMS_WM.RevokeSystemPriv
    (priv_types IN VARCHAR2,
    grantee IN VARCHAR2
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Отзывает привилегии *priv_types* у *grantee*. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.RevokeWorkspacePriv
    (priv_types IN VARCHAR2,
    workspace IN VARCHAR2,
    grantee IN VARCHAR2
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Отзывает привилегии *priv_types* для рабочего пространства *workspace* у *grantee*. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.RollbackDDL
    (table_name IN VARCHAR2);
```

Откатывает DDL-изменения для таблицы *table_name* и завершает сеанс.

```
PROCEDURE DBMS_WM.RollbackResolve
    (workspace IN VARCHAR2);
```

Завершает сеанс разрешения конфликта и удаляет все изменения, которые были сделаны в рабочем пространстве *workspace* с момента вызова процедуры BeginResolve.

```
PROCEDURE DBMS_WM.RollbackTable
    (workspace IN VARCHAR2,
    table_id IN VARCHAR2
    [, sp_name IN VARCHAR2 DEFAULT ``]
    [, where_clause IN VARCHAR2 DEFAULT ``]
    [, remove_locks IN BOOLEAN DEFAULT TRUE]
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Удаляет изменения в таблице *table_id* рабочего пространства *workspace*. Параметр *sp_name* задает имя точки сохранения, до которой осуществляется откат. Инструкция *where_clause* задает подмножество строк для блокировки и может ссылаться только на столбцы первичного ключа. Если параметр *remove_locks* установлен в TRUE, то процедура освобождает блокировки на строки в родительском рабочем пространстве, удовлетворяющие условию инструкции *where_clause* (параметр не действует, если задан параметр *sp_name*). Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.RollbackToSP
    (workspace IN VARCHAR2,
    savepoint_name IN VARCHAR2
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Откатывает изменения в рабочем пространстве *workspace* до точки сохранения *savepoint_name*. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.RollbackWorkspace
    (workspace IN VARCHAR2
    [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

Откатывает изменения в рабочем пространстве *workspace* для таблиц с поддержкой многоверсионности строк. Если значение параметра *auto_commit* равно TRUE, то операция автоматически фиксируется после выполнения.

```
PROCEDURE DBMS_WM.SetConflictWorkspace
    (workspace IN VARCHAR2);
```

Определяет, существует ли конфликт между рабочим пространством *workspace* и его родителем.

```
PROCEDURE DBMS_WM.SetDiffVersions
    (workspace1 IN VARCHAR2,
    [savepoint1 IN VARCHAR2,]
    workspace2 IN VARCHAR2
    [, savepoint2 IN VARCHAR2]);
```

Проверяет различия в многоверсионных таблицах рабочих пространств *workspace1* и *workspace2* и изменяет содержимое соответствующих представлений для

отражения таких различий. Если переменные точек сохранения не заданы, то берется точка сохранения LATEST.

```
PROCEDURE DBMS_WM.SetLockingOff;
```

Выключает блокировку Workspace Manager для текущего сеанса.

```
PROCEDURE DBMS_WM.SetLockingOn
    (lockmode IN VARCHAR2);
```

Включает блокировку в режиме *lockmode*: 'E' (exclusive – монопольный режим блокировки), 'S' (shared – разделяемый режим) или 'C' (carry-forward – строки в текущем рабочем пространстве блокируются в том же режиме, что и соответствующие им строки в предыдущей версии).

```
PROCEDURE DBMS_WM.SetMultiWorkspaces
    (workspaces IN VARCHAR2);
```

Делает видимыми рабочие пространства из приведенного списка.

```
PROCEDURE DBMS_WM.SetWOOverwriteOff;
```

Отменяет запись исторической информации VIEW_WO_OVERWRITE, изменяя значение параметра на VIEW_W_OVERWRITE.

```
PROCEDURE DBMS_WM.SetWOOverwriteOn;
```

Включает запись исторической информации VIEW_W_OVERWRITE, изменяя значение параметра на VIEW_WO_OVERWRITE.

```
PROCEDURE DBMS_WM.SetWorkspaceLockModeOff
    (workspace IN VARCHAR2);
```

Выключает блокировку Workspace Manager для рабочего пространства *workspace*.

```
PROCEDURE DBMS_WM.SetWorkspaceLockModeOn
    (workspace IN VARCHAR2,
    lockmode IN VARCHAR2
    [, override IN BOOLEAN DEFAULT FALSE]);
```

Включает блокировку для рабочего пространства *workspace* в режиме *lockmode*: 'E' (exclusive – монопольный режим блокировки), 'S' (shared – разделяемый режим) или 'C' (carry-forward – строки в текущем рабочем пространстве блокируются в том же режиме, что и соответствующие им строки в предыдущей версии). Если параметр *override* установлен в TRUE, то разрешено изменение режима блокировки.

```
PROCEDURE DBMS_WM.SynchronizeSite
    (newwritersite IN VARCHAR2);
```

Обновляет новый пишущий узел *newwritersite*, после того как тот был перемещен процедурой RelocateWriterSite.

```
PROCEDURE DBMS_WM.UnfreezeWorkspace
    (workspace IN VARCHAR2);
```

«Размораживает» рабочее пространство *workspace*, то есть разрешает доступ к нему и его изменение.

```
PROCEDURE DBMS_WM.UnlockRows
    (workspace IN VARCHAR2,
    table_name IN VARCHAR2
    [, where_clause IN VARCHAR2 DEFAULT '']
    [, all_or_user IN VARCHAR2 DEFAULT 'USER']
    [, lock_mode IN VARCHAR2 DEFAULT 'ES']);
```

Снимает блокировку для многоверсионных строк таблицы *table_name* рабочего пространства *workspace* и его родителя. Инструкция *where_clause* задает подмножество строк для снятия блокировки и может ссылаться только на столбцы первичного ключа. Параметр *all_or_user* определяет область действия процедуры: для всех блокировок или только для блокировок пользователя. Параметр *lock_mode* может иметь значения 'E', 'S', или 'ES', указывая тип рассматриваемых блокировок.

DBMS_XDB

Содержит процедуры и функции, обеспечивающие интерфейс для XML-данных, хранящихся в БД Oracle. Появилась в Oracle9*i*.

Вызовы

```
FUNCTION DBMS_XDB.getAclDocument
    (abspath IN VARCHAR);
    RETURN SYS.XMLTYPE;
```

Извлекает ACL-документ (Access Control List – список контроля доступа) для XML-документа, путь к которому указан в *abspath*.

```
FUNCTION DBMS_XDB.getPrivileges
    (res_path IN VARCHAR2)
    RETURN SYS.XMLTYPE;
```

Возвращает привилегии, предоставленные текущему пользователю на XML-документ, абсолютный путь к которому указан в *res_path*.

```
FUNCTION DBMS_XDB.changePrivileges
    (res_path IN VARCHAR2,
    ace IN XMLTYPE)
    RETURN PLS_INTEGER;
```

Добавляет привилегию *ace* для XML-документа, абсолютный путь к которому указан в *res_path*. В случае успешного изменения ACL возвращает положительное целое число.

```
FUNCTION DBMS_XDB.checkPrivileges
    (res_path IN VARCHAR2,
    privs IN XMLTYPE)
    RETURN PLS_INTEGER;
```

Проверяет, имеет ли XML-документ, абсолютный путь к которому указан в *res_path*, привилегию *privs*. В случае положительного ответа возвращает положительное целое число.

```
PROCEDURE DBMS_XDB.setacl
    (res_path IN VARCHAR2,
    acl_path IN VARCHAR2);
```

Задает ACL для XML-документа, абсолютный путь к которому указан в иерархии *res_path*, как ACL, абсолютный путь к которому указан в *acl_path*.

```
FUNCTION DBMS_XDB.AclCheckPrivileges
    (acl_path IN VARCHAR2,
    owner IN VARCHAR2,
    privs IN XMLTYPE)
    RETURN PLS_INTEGER;
```

Проверяет привилегии *privs* для документа, принадлежащего *owner*, на соответствие ACL, абсолютный путь к которому указан в *acl_path*. В случае если все привилегии были предоставлены, возвращает положительное целое число.

```
FUNCTION DBMS_XDB.LockResource
    (path IN VARCHAR2,
    depthzero IN BOOLEAN,
    shared IN BOOLEAN)
    RETURN BOOLEAN;
```

Создает блокировку для XML-ресурса, путь к которому указан в *path*. Если значение параметра *shared* равно TRUE, то функция создает разделяемую блокировку. Параметр *depthzero* в настоящее время не поддерживается. Возвращает TRUE в случае получения блокировки.

```
PROCEDURE DBMS_XDB.GetLockToken
    (path IN VARCHAR2,
    locktoken OUT VARCHAR2);
```

Получает токен блокировки для XML-ресурса, путь к которому указан в *path*.

```
FUNCTION DBMS_XDB.UnlockResource
    (path IN VARCHAR2,
    deltoken IN VARCHAR2)
    RETURN BOOLEAN;
```

Снимает блокировку для XML-ресурса, путь к которому указан в *path*, удаляя токен *deltoken*. Возвращает TRUE в случае успеха.

```
FUNCTION DBMS_XDB.CreateResource
    (path IN VARCHAR2,
    input IN datatype)
    RETURN BOOLEAN;
```

Создает новый XML-ресурс, путь к которому указан в *path*. Значениями *datatype* могут быть типы VARCHAR2, SYS.XMLTYPE, REF SYS.XMLTYPE, CLOB или BFILE. Возвращает TRUE в случае успеха.

```
FUNCTION DBMS_XDB.CreateFolder
    (path IN VARCHAR2)
    RETURN BOOLEAN;
```

Создает в иерархии XML папку, путь к которой указан в *path*. Возвращает TRUE в случае успеха.

```
PROCEDURE DBMS_XDB.DeleteResource
    (path IN VARCHAR2);
```

Удаляет XML-ресурс, путь к которому указан в *path*.

```
PROCEDURE DBMS_XDB.Link
    (srcpath IN VARCHAR2,
    linkfolder IN VARCHAR2,
    linkname IN VARCHAR2);
```

Создает ссылку *linkname* в папке *linkfolder*, указывающую на существующий ресурс, путь к которому задан параметром *srcpath*.

```
PROCEDURE DBMS_XDB.CFGRefresh;
```

Обновляет сведения о конфигурации для сеанса.

```
FUNCTION DBMS_XDB.CFG_Get
   RETURN SYS.XMLTYPE;
```

Возвращает сведения о конфигурации для сеанса.

```
PROCEDURE DBMS_XDB.CFG_Update
   (xdbconfig IN SYS.XMLTYPE);
```

Обновляет данные о конфигурации на *xdbconfig*.

DBMS_XDBT

Предоставляет интерфейс для создания индекса Oracle ConText для иерархии БД XML. Появилась в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_XDBT.dropPreferences;
```

Удаляет предпочтения для индекса ConText в иерархии XML.

```
PROCEDURE DBMS_XDBT.createPreferences;
```

Создает предпочтения по умолчанию для индекса ConText в иерархии XML.

```
PROCEDURE DBMS_XDBT.createDatastorePreferences
```

Создает предпочтение хранения данных USER для индекса ConText в иерархии XML.

```
PROCEDURE DBMS_XDBT.createFilterPref;
```

Создает предпочтение фильтра NULL для индекса ConText в иерархии XML.

```
PROCEDURE DBMS_XDBT.createLexerPref;
```

Создает предпочтение фильтра лексического анализатора BASIC (*lexer*) для индекса ConText в иерархии XML.

```
PROCEDURE DBMS_XDBT.createWordlistPref;
```

Создает предпочтение списка слов (*wordlist* – список всех полезных слов в текстовом файле) для индекса ConText в иерархии XML.

```
PROCEDURE DBMS_XDBT.createStoplistPref;
```

Создает предпочтение стоп-списка (*stoplist* – список несущественных слов, не включаемых в индекс и не выбираемых пользователем) для индекса ConText в иерархии XML.

```
PROCEDURE DBMS_XDBT.createStoragePref;
```

Создает предпочтение основного хранилища BASIC_STORAGE для индекса ConText в иерархии XML.

```
PROCEDURE DBMS_XDBT.createSectiongroupPref;
```

Создает предпочтение группы секций для индекса ConText в иерархии XML.

```
PROCEDURE DBMS_XDBT.createIndex;
```

Создает индекс ConText в иерархии XML.

```
PROCEDURE DBMS_XDBT.configureAutoSync
```

Создает задания для автоматической синхронизации индекса ConText в иерархии XML.

DBMS_XDB_VERSION

Содержит процедуры и функции, предназначенные для создания и управления версиями в XDB. Появился в Oracle9*i*.

Вызовы

```
FUNCTION DBMS_XDB_VERSION.MakeVersioned
    (pathname IN VARCHAR2)
    RETURN DBMS_XDD.RESID_TYPE;
```

Включает управление версиями для ресурса, путь к которому указан в *pathname*.

```
PROCEDURE DBMS_XDB_VERSION.Checkout
    (pathname IN VARCHAR2);
```

Проверяет включенный в управление версиями ресурс, путь к которому указан в *pathname*, прежде чем обновить или удалить его. Возвращает идентификатор ресурса для первой версии.

```
FUNCTION DBMS_XDB_VERSION.Checkin
    (pathname IN VARCHAR2)
    RETURN DBMS_XDD.RESID_TYPE;
```

Проверяет включенный в контроль версий ресурс, путь к которому указан в *pathname*, и возвращает идентификатор ресурса для новой версии.

```
FUNCTION DBMS_XDB_VERSION.Uncheckout
    (pathname IN VARCHAR2)
    RETURN DBMS_XDD.RESID_TYPE;
```

Проверяет включенный в управление версиями ресурс, путь к которому указан в *pathname*, и возвращает идентификатор ресурса для старой версии.

```
PROCEDURE DBMS_XDB_VERSION.GetPredecessors
    (pathname IN VARCHAR2)
    RETURN DBMS_XDD.RESID_TYPE;
```

Возвращает список предшественников для ресурса, путь к которому указан в *pathname*.

```
FUNCTION DBMS_XDB_VERSION.GetPredsByResID
    (resid IN RESID_TYPE)
    RETURN RESID_LIST_TYPE;
```

Возвращает список предшественников для ресурса с идентификатором *resid*.

```
FUNCTION DBMS_XDB_VERSION.GetResourceByResID
    (resid IN RESID_TYPE)
    RETURN XMLTYPE;
```

Возвращает ресурс, идентифицируемый параметром *resid*.

```
FUNCTION DBMS_XDB_VERSION.GetSuccessors
    (pathname IN VARCHAR2)
    RETURN RESID_LIST_TYPE;
```

Возвращает список преемников для ресурса, путь к которому указан в *pathname*.

```
FUNCTION DBMS_XDB_VERSION.GetSuccsByResId
    (resid IN RESID_TYPE)
    RETURN RESID_LIST_TYPE;
```

Возвращает список преемников для ресурса с идентификатором *resid*.

DBMS_XMLDOM

Обеспечивает доступ к объектам XMLType через программный интерфейс DOM (Document Object Model). Этот пакет применяется для реализации различных частей интерфейсов консорциума World Wide Web Consortium (W3C) для различных частей DOM; входящие в него подпрограммы сгруппированы в соответствии с такими интерфейсами.

Каталоги, из которых будет происходить считывание, и каталоги для записи должны быть заданы в файле *INIT.ORA* или *SPFILE* (например, в таких параметрах, как UTL_FILE_DIR). Появился в Oracle9*i*.

Вызовы

В разделе описаны вызовы DBMS_XMLDOM различных категорий: общие вызовы, методы узла DOM, методы карты именованных узлов DOM, методы списка узлов DOM, методы атрибута DOM, методы символьных данных DOM, методы реализации DOM, методы типа документа DOM, методы элемента DOM, методы сущностей DOM, методы нотации DOM, методы директивы DOM, методы текста DOM и методы документа DOM.

Общие вызовы

Представленные далее вызовы могут использоваться практически во всех составляющих модели DOM. Значениями DOMdatatype могут быть: DOMNode, DOMNamedNodeMap, DOMNodeList, DOMAttr, DOMCDateSection, DOMCharacterData, DOMComment, DOMImplementation, DOMDocumentFragment, DOMDocumentType, DOMElement, DOMEntity, DOMEntityRef, DOMNotation, DOMText и DOMDocument.

```
FUNCTION DBMS_XMLDOM.isNull
    (n IN DOMdatatype)
    RETURN BOOLEAN;
```

Возвращает TRUE, если *n* содержит NULL.

```
FUNCTION DBMS_XMLDOM.makeNode
    (n IN DOMdatatype)
    RETURN DOMNode;
```

Создает узел из *n* и возвращает узел DOMNode. Эта функция может использоваться в любой части DOM, за исключением DOMNode, DOMNamedNodeMap, DOMNodeList и DOMImplementation.

Методы узла DOM

```
FUNCTION isNull
    (n IN DOMNode)
    RETURN BOOLEAN;
```

Возвращает TRUE, если *n* содержит NULL.

```
FUNCTION DBMS_XMLDOM.makeAttr
    (n IN DOMNode)
    RETURN DOMAttr;
```

Приводит *n* к узлу вида DOMAttr.

```
FUNCTION DBMS_XMLDOM.makeCDataSection
    (n IN DOMNode)
    RETURN DOMDataSection;
```

Приводит *n* к узлу вида DOMDataSection.

```
FUNCTION DBMS_XMLDOM.makeCharacterData
    (n IN DOMNode)
    RETURN DOMCharacterData;
```

Приводит *n* к узлу вида DOMCharacterData.

```
FUNCTION DBMS_XMLDOM.makeComment
    (n IN DOMNode)
    RETURN DOMComment;
```

Приводит *n* к узлу вида DOMComment.

```
FUNCTION DBMS_XMLDOM.makeDocumentFragment
    (n IN DOMNode)
    RETURN DOMDocumentFragment;
```

Приводит *n* к узлу вида DOMDocumentFragment.

```
FUNCTION DBMS_XMLDOM.makeDocumentType
    (n IN DOMNode)
    RETURN DOMDocumentType;
```

Приводит *n* к узлу вида DOMDocumentType.

```
FUNCTION DBMS_XMLDOM.makeElement
    (n IN DOMNode)
    RETURN DOMElement;
```

Приводит *n* к узлу вида DOMElement.

```
FUNCTION DBMS_XMLDOM.makeEntity
    (n IN DOMNode)
    RETURN DOMEntity;
```

Приводит *n* к узлу вида DOMEntity.

```
FUNCTION DBMS_XMLDOM.makeEntityReference
    (n IN DOMNode)
    RETURN DOMEntityReference;
```

Приводит *n* к узлу вида DOMEntityReference.

```
FUNCTION DBMS_XMLDOM.makeNotation
    (n IN DOMNode)
    RETURN DOMNotation;
```

Приводит *n* к узлу вида DOMNotation.

```
FUNCTION DBMS_XMLDOM.makeProcessingInstruction
    (n IN DOMNode)
    RETURN DOMProcessingInstruction;
```

Приводит *n* к узлу вида DOMProcessingInstruction.

```
FUNCTION DBMS_XMLDOM.makeText
    (n IN DOMNode)
    RETURN DOMText;
```

Приводит *n* к узлу вида DOMText.

```
FUNCTION DBMS_XMLDOM.makeDocument
    (n IN DOMNode)
    RETURN DOMDocument;
```

Приводит *n* к узлу вида DOMDocument.

```
PROCEDURE DBMS_XMLDOM.WriteToFile
    (n IN DOMNode,
```

```
    filename IN VARCHAR2
    [, charset IN VARCHAR2);
```

Записывает узел *n* в файл *filename*, используя набор символов *charset* (если параметр задан) или набор символов БД (если параметр *charset* не задан).

```
PROCEDURE DBMS_XMLDOM.WriteToBuffer
    (n IN DOMNode,
    buffer IN OUT VARCHAR2
    [, charset IN VARCHAR2);
```

Записывает узел *n* в буфер *buffer*, используя набор символов *charset* (если параметр задан) или набор символов БД (если параметр *charset* не задан).

```
PROCEDURE DBMS_XMLDOM.WriteToBuffer
    (n IN DOMNode,
    cl IN OUT CLOB
    [, charset IN VARCHAR2);
```

Записывает узел *n* в объект CLOB *cl*, используя набор символов *charset* (если параметр задан) или набор символов БД (если параметр *charset* не задан).

```
FUNCTION DBMS_XMLDOM.getNodeName
    (n IN DOMNode)
    RETURN VARCHAR2;
```

Возвращает имя для узла *n*.

```
FUNCTION DBMS_XMLDOM.getNodeValue
    (n IN DOMNode)
    RETURN VARCHAR2;
```

Возвращает значение для узла *n*.

```
PROCEDURE DBMS_XMLDOM.setNodeValue
    (n IN DOMNode,
    modeValue IN VARCHAR2);
```

Устанавливает значение *n* в *modeValue*.

```
FUNCTION DBMS_XMLDOM.getNodeType
    (n IN DOMNode)
    RETURN NUMBER;
```

Возвращает тип для узла *n*.

```
FUNCTION DBMS_XMLDOM.getParentNode
    (n IN DOMNode)
    RETURN DOMNode;
```

Возвращает родительский узел для *n*.

```
FUNCTION DBMS_XMLDOM.getChildNodes
    (n IN DOMNode)
    RETURN DOMNodeList;
```

Возвращает потомков узла *n*.

```
FUNCTION DBMS_XMLDOM.getFirstChild
    (n IN DOMNode)
    RETURN DOMNode;
```

Возвращает первого потомка узла *n*.

```
FUNCTION DBMS_XMLDOM.getLastChild
    (n IN DOMNode)
```

```
    RETURN DOMNode;
```

Возвращает последнего потомка для узла *n*.

```
FUNCTION DBMS_XMLDOM.getPreviousSibling
    (n IN DOMNode)
    RETURN DOMNode;
```

Возвращает сестринский узел, расположенный непосредственно перед узлом *n*.

```
FUNCTION DBMS_XMLDOM.getNextSibling
    (n IN DOMNode)
    RETURN DOMNode;
```

Возвращает сестринский узел, расположенный непосредственно после узла *n*.

```
FUNCTION DBMS_XMLDOM.getAtributes
    (n IN DOMNode)
    RETURN DOMNamedNodeMap;
```

Возвращает атрибуты узла *n*.

```
FUNCTION DBMS_XMLDOM.getOwnerDocument
    (n IN DOMNode)
    RETURN DOMDocument;
```

Возвращает объект документа, сопоставленный узлу *n*.

```
FUNCTION DBMS_XMLDOM.insertBefore
    (n IN DOMNode,
    newChild IN DOMNode,
    refChild IN DOMNode)
    RETURN DOMNode;
```

Вставляет нового потомка *newChild* перед указанным потомком *refChild* узла *n*.

```
FUNCTION DBMS_XMLDOM.replaceChild
    (n IN DOMNode,
    newChild IN DOMNode,
    oldChild IN DOMNode)
    RETURN DOMNode;
```

Заменяет потомка *oldChild* узла *n* на *newChild*.

```
FUNCTION DBMS_XMLDOM.removeChild
    (n IN DOMNode,
    oldChild IN DOMNode)
    RETURN DOMNode;
```

Удаляет потомка *oldChild* узла *n* и возвращает его.

```
FUNCTION DBMS_XMLDOM.appendChild
    (n IN DOMNode,
    newChild IN DOMNode)
    RETURN DOMNode;
```

Добавляет узел *newChild* в список потомков узла *n*.

```
FUNCTION DBMS_XMLDOM.hasChildNodes
    (n IN DOMNode)
    RETURN BOOLEAN;
```

Возвращает TRUE, если узел *n* имеет потомков.

```
PROCEDURE DBMS_XMLDOM.cloneNode
    (n IN DOMNode,
    deep IN BOOLEAN)
```

```
    RETURN DOMNode;
```

Создает и возвращает клон узла *n*. Если значение параметра *deep* равно TRUE, то клонируются и потомки узла.

Методы карты именованных узлов DOM

```
FUNCTION DBMS_XMLDOM.getNamedItem
    (nnm DOMNamedNodeMap,
    name IN VARCHAR2)
    RETURN DOMNode;
```

Извлекает узел DOMNode из карты именованных узлов *nnm* по имени *name*.

```
FUNCTION DBMS_XMLDOM.setNamedItem
    (nnm DOMNamedNodeMap,
    arg IN VARCHAR2)
    RETURN DOMNode;
```

Добавляет узел в карту *nnm* с использованием его NodeName *arg*.

```
FUNCTION DBMS_XMLDOM.removeNamedItem
    (nnm DOMNamedNodeMap,
    name IN VARCHAR2)
    RETURN DOMNode;
```

Удаляет узел с именем *name* из *nnm*.

```
FUNCTION DBMS_XMLDOM.item
    (nnm DOMNamedNodeMap,
    index IN NUMBER)
    RETURN DOMNode;
```

Возвращает элемент карты *nnm*, который соответствует индексу *index*. Если значение параметра *index* превышает (или равно) количество узлов в *nnm*, то возвращается NULL.

```
FUNCTION DBMS_XMLDOM.getLength
    (nnm DOMNamedNodeMap,
    RETURN NUMBER;
```

Возвращает количество элементов карты *nnm*.

Методы списка узлов DOM

```
FUNCTION DBMS_XMLDOM.item
    (nl IN DOMNodeList,
    index IN NUMBER)
    RETURN DOMNode;
```

Возвращает элемент списка узлов *nl*, который соответствует индексу *index*. Если значение параметра *index* превышает (или равно) количество узлов в *nl*, то возвращается NULL.

```
FUNCTION DBMS_XMLDOM.getLength
    (nl DOMNodeList,
    RETURN NUMBER;
```

Возвращает количество элементов в списке узлов *nl*.

Методы атрибутов DOM

```
FUNCTION DBMS_XMLDOM.getQualifiedName
    (a IN DOMAttr)
    RETURN VARCHAR2;
```

Возвращает уточненное имя атрибута *a*.

```
FUNCTION DBMS_XMLDOM.getNamespace
    (a IN DOMAttr)
    RETURN VARCHAR2;
```

Возвращает пространство имен атрибута *a*.

```
FUNCTION DBMS_XMLDOM.getLocalName
    (a IN DOMAttr)
    RETURN VARCHAR2;
```

Возвращает локальное имя атрибута *a*.

```
FUNCTION DBMS_XMLDOM.getExpandedName
    (a IN DOMAttr)
    RETURN VARCHAR2;
```

Возвращает расширенное имя атрибута *a*.

```
FUNCTION DBMS_XMLDOM.getName
    (a IN DOMAttr)
    RETURN VARCHAR2;
```

Возвращает имя атрибута *a*.

```
FUNCTION DBMS_XMLDOM.getSpecified
    (a IN DOMAttr)
    RETURN BOOLEAN;
```

Возвращает TRUE, если в исходном документе было явно указано значение атрибута *a*.

```
FUNCTION DBMS_XMLDOM.getValue
    (a IN DOMAttr)
    RETURN VARCHAR2;
```

Возвращает значение атрибута *a*.

```
FUNCTION DBMS_XMLDOM.setValue
    (a IN DOMAttr,
    value IN VARCHAR2);
```

Задает значение атрибута *a*.

Методы символьных данных DOM

```
FUNCTION DBMS_XMLDOM.getData
    (cd IN DOMCharacterData)
    RETURN VARCHAR2;
```

Возвращает символьные данные узла *cd* вида DOMCharacterData.

```
FUNCTION DBMS_XMLDOM.setData
    (cd IN DOMCharacterData,
    data IN VARCHAR2);
```

Задает для *cd* символьные данные *data*.

```
FUNCTION DBMS_XMLDOM.getLength
    (cd IN DOMCharacterData)
    RETURN NUMBER;
```

Возвращает количество 16-битных элементов, содержащихся в *cd* или в результате, возвращаемом методом substringData() для *cd*.

```
FUNCTION DBMS_XMLDOM.substringData
    (cd IN DOMCharacterData,
    offset IN NUMBER,
```

```
    cnt IN NUMBER)
    RETURN VARCHAR2;
```

Возвращает данные *data* из *cd*, расположенные начиная с позиции *offset* и имеющие длину *cnt* символов.

```
 PROCEDURE DBMS_XMLDOM.appendData
    (cd IN DOMCharacterData,
    arg IN VARCHAR2);
```

Добавляет строку *arg* в конец символьных данных *cd*.

```
PROCEDURE DBMS_XMLDOM.insertData
    (cd IN DOMCharacterData,
    offset IN NUMBER,
    arg IN VARCHAR2);
```

Вставляет строку *arg* в *cd* начиная с позиции *offset*.

```
PROCEDURE DBMS_XMLDOM.deleteData
    (cd IN DOMCharacterData,
    offset IN NUMBER,
    cnt IN NUMBER);
```

Удаляет *cnt* символов из *cd* начиная с позиции *offset*.

```
PROCEDURE DBMS_XMLDOM.replaceData
    (cd IN DOMCharacterData,
    offset IN NUMBER,
    cnt IN NUMBER,
    arg IN VARCHAR2);
```

Заменяет *cnt* символов в *cd*, начиная с позиции *offset*, на *arg*.

Методы реализации DOM

```
FUNCTION DBMS_XMLDOM.hasFeature
    (di IN DOMImplementation,
    feature IN VARCHAR2,
    version IN VARCHAR2);
```

Возвращает TRUE, если узел DOMImplementation *di* реализует свойство *feature* для версии *version*.

Методы типа документа DOM

```
FUNCTION DBMS_XMLDOM.findEntity
    (dt IN DOMDocumentType,
    name IN VARCHAR2,
    par IN BOOLEAN)
    RETURN DOMEntity;
```

Находит сущность *name* в узле DOMDocumentType *dt* и возвращает ее. Если значение параметра *par* равно TRUE, то речь идет о параметрической сущности, иначе – об обычной.

```
FUNCTION DBMS_XMLDOM.findNotation
    (dt IN DOMDocumentType,
    name IN VARCHAR2)
    RETURN DOMEntity;
```

Находит нотацию *name* в узле DOMDocumentType *dt* и возвращает ее.

```
FUNCTION DBMS_XMLDOM.getPublicId
    (dt IN DOMDocumentType)
```

```
    RETURN VARCHAR2;
```

Возвращает открытый идентификатор *dt*.

```
FUNCTION DBMS_XMLDOM.getSystemId
    (dt IN DOMDocumentType)
    RETURN VARCHAR2;
```

Возвращает системный идентификатор *dt*.

```
PROCEDURE DBMS_XMLDOM.writeExternalDTDToFile
    (dt IN DOMDocumentType,
    filename IN VARCHAR2
    [, charset IN VARCHAR2]);
```

Записывает *dt* в файл *filename*, используя набор символов *charset* (если параметр задан) или набор символов БД (если параметр *charset* не задан).

```
PROCEDURE DBMS_XMLDOM.writeExternalDTDToBuffer
    (dt IN DOMDocumentType,
    buffer IN OUT VARCHAR2
    [, charset IN VARCHAR2]);
```

Записывает *dt* в буфер *buffer*, используя набор символов *charset* (если параметр задан) или набор символов БД (если параметр *charset* не задан).

```
PROCEDURE DBMS_XMLDOM.writeExternalDTDToClob
    (dt IN DOMDocumentType,
    cl IN CLOB
    [, charset IN VARCHAR2]);
```

Записывает *dt* в объект CLOB *cl*, используя набор символов *charset* (если параметр задан) или набор символов БД (если параметр *charset* не задан).

```
FUNCTION DBMS_XMLDOM.getName
    (dt IN DOMDocumentType)
    RETURN VARCHAR2;
```

Возвращает имя *dt*.

```
FUNCTION DBMS_XMLDOM.getEntities
    (dt IN DOMDocumentType)
    RETURN DOMNamedNodeMap;
```

Возвращает карту NamedNodeMap, содержащую общие внутреннюю и внешнюю секции *dt*.

```
FUNCTION DBMS_XMLDOM.getNotations
    (dt IN DOMDocumentType)
    RETURN DOMNamedNodeMap;
```

Возвращает карту NamedNodeMap, содержащую нотации *dt*.

Методы элемента DOM

```
FUNCTION DBMS_XMLDOM.getQualifiedName
    (elem IN DOMElement)
    RETURN VARCHAR2;
```

Возвращает уточненное имя элемента *elem*.

```
FUNCTION DBMS_XMLDOM.getNamespace
    (elem IN DOMElement)
    RETURN VARCHAR2;
```

Возвращает пространство имен элемента *elem*.

```
FUNCTION DBMS_XMLDOM.getLocalName
    (elem IN DOMElement)
    RETURN VARCHAR2;
```

Возвращает локальное имя элемента *elem*.

```
FUNCTION DBMS_XMLDOM.getExpandedName
    (elem IN DOMElement)
    RETURN VARCHAR2;
```

Возвращает расширенное имя элемента *elem*.

```
FUNCTION DBMS_XMLDOM.getChildrenByTagName
    (elem IN DOMElement,
    name IN VARCHAR2
    [, ns IN VARCHAR2])
    RETURN DOMNodeList;
```

Возвращает список потомков для *elem* на основе имени *name* или имени *name* с учетом пространства имен *ns*.

```
FUNCTION DBMS_XMLDOM.getElementsByTagName
    (elem IN DOMElement,
    name IN VARCHAR2
    [, ns IN VARCHAR2])
    RETURN DOMNodeList;
```

Возвращает список всех элементов-потомков *elem* на основе имени *name* или имени *name* с учетом пространства имен *ns*.

```
FUNCTION DBMS_XMLDOM.resolveNamespacePrefix
    (elem IN DOMElement,
    prefix IN VARCHAR2
    RETURN VARCHAR2;
```

Возвращает пространство имен для *elem* на основе префикса *prefix*.

```
FUNCTION DBMS_XMLDOM.getTagName
    (elem IN DOMElement)
    RETURN VARCHAR2;
```

Возвращает имя *elem*.

```
FUNCTION DBMS_XMLDOM.getAttribute
    (elem IN DOMElement,
    name IN VARCHAR2)
    RETURN VARCHAR2;
```

Возвращает значение атрибута *name* элемента *elem*.

```
PROCEDURE DBMS_XMLDOM.setAttribute
    (elem IN DOMElement,
    name IN VARCHAR2,
    value IN VARCHAR);
```

Присваивает значение *value* атрибуту *name* элемента *elem*.

```
PROCEDURE DBMS_XMLDOM.removeAttribute
    (elem IN DOMElement,
    name IN VARCHAR2);
```

Удаляет атрибут *name* элемента *elem*.

```
FUNCTION DBMS_XMLDOM.getAttributeNode
    (elem IN DOMElement,
```

```
    name IN VARCHAR2)
    RETURN DOMAttr;
```

Возвращает узел атрибута с именем *name* элемента *elem*.

```
FUNCTION DBMS_XMLDOM.setAttributeNode
    (elem IN DOMElement,
    newAttr IN DOMAttr)
    RETURN DOMAttr;
```

Добавляет узел атрибута *newAttr* в *elem*.

```
FUNCTION DBMS_XMLDOM.removeAttributeNode
    (elem IN DOMElement,
    oldAttr IN DOMAttr)
    RETURN DOMAttr;
```

Удаляет узел атрибута *oldAttr* из *elem*.

```
PROCEDURE DBMS_XMLDOM.normalize
    (elem IN DOMElement);
```

Нормализует текст потомков *elem*.

Методы сущности DOM

```
FUNCTION DBMS_XMLDOM.getPublicID
    (ent IN DOMEntity)
    RETURN VARCHAR2;
```

Возвращает открытый идентификатор сущности *ent*.

```
FUNCTION DBMS_XMLDOM.getSystemID
    (ent IN DOMEntity)
    RETURN VARCHAR2;
```

Возвращает системный идентификатор сущности *ent*.

```
FUNCTION DBMS_XMLDOM.getNotationName
    (ent IN DOMEntity)
    RETURN VARCHAR2;
```

Возвращает имя нотации сущности *ent*.

Методы нотации DOM

```
FUNCTION DBMS_XMLDOM.getPublicID
    (n IN DOMNotation)
    RETURN VARCHAR2;
```

Возвращает открытый идентификатор нотации *n*.

```
FUNCTION DBMS_XMLDOM.getSystemID
    (n IN DOMNotation)
    RETURN VARCHAR2;
```

Возвращает системный идентификатор нотации *n*.

Методы директивы DOM

```
FUNCTION DBMS_XMLDOM.getData
    (pi IN DOMProcessingInstruction)
    RETURN VARCHAR2;
```

Возвращает данные – содержимое директивы *pi*.

```
FUNCTION DBMS_XMLDOM.getTarget
    (pi IN DOMProcessingInstruction)
```

```
    RETURN VARCHAR2;
```

Возвращает цель директивы *pi*.

```
PROCEDURE DBMS_XMLDOM.setData
    (pi IN DOMProcessingInstruction,
    data IN VARCHAR2);
```

Устанавливает содержимое директивы *pi* в *data*.

Методы текста DOM

```
FUNCTION DBMS_XMLDOM.splitText
    (t IN DOMText,
    offset IN NUMBER)
    RETURN DOMText;
```

Разбивает документ *t* на два узла DOMText по позиции *offset*.

Методы документа DOM

```
FUNCTION DBMS_XMLDOM.newDOMDocument
    RETURN DOMDocument;
```

Возвращает новый экземпляр узла DOMDocument.

```
PROCEDURE DBMS_XMLDOM.freeDocument
    (doc IN DOMDocument);
```

Освобождает документ *doc*.

```
FUNCTION DBMS_XMLDOM.getVersion
    (doc IN DOMDocument)
    RETURN VARCHAR2;
```

Возвращает информацию о версии для документа *doc*.

```
PROCEDURE DBMS_XMLDOM.setVersion
    (doc IN DOMDocument
    version IN VARCHAR2);
```

Указывает версию для документа *doc*.

```
FUNCTION DBMS_XMLDOM.getCharset
    (doc IN DOMDocument)
    RETURN VARCHAR2;
```

Возвращает набор символов документа *doc*.

```
PROCEDURE DBMS_XMLDOM.setCharset
    (doc IN DOMDocument
    charset IN VARCHAR2);
```

Задает набор символов документа *doc*.

```
FUNCTION DBMS_XMLDOM.getStandalone
    (doc IN DOMDocument)
    RETURN VARCHAR2;
```

Возвращает автономную информацию для документа *doc*.

```
PROCEDURE DBMS_XMLDOM.setStandalone
    (doc IN DOMDocument
    value IN VARCHAR2);
```

Указывает значение *value* для автономной информации документа *doc*.

```
PROCEDURE DBMS_XMLDOM.writeToFile
    (doc IN DOMDocument,
```

```
    filename IN VARCHAR2
    [, charset IN VARCHAR2]);
```

Записывает документ *doc* в файл *filename*, используя набор символов *charset* (если параметр задан) или набор символов БД (если параметр *charset* не задан).

```
PROCEDURE DBMS_XMLDOM.writeToBuffer
    (doc IN DOMDocument,
    buffer IN OUT VARCHAR2
    [, charset IN VARCHAR2]);
```

Записывает документ *doc* в буфер *buffer*, используя набор символов *charset* (если параметр задан) или набор символов БД (если параметр *charset* не задан).

```
PROCEDURE DBMS_XMLDOM.writeToClob
    (doc IN DOMDocument,
    cl IN CLOB
    [, charset IN VARCHAR2]);
```

Записывает документ *doc* в объект CLOB *cl*, используя набор символов *charset* (если параметр задан) или набор символов БД (если параметр *charset* не задан).

```
PROCEDURE DBMS_XMLDOM.writeExternalDTDToFile
    (doc IN DOMDocument,
    filename IN VARCHAR2
    [, charset IN VARCHAR2]);
```

Записывает внешнее определение типа документа (Document Type Definition – DTD) *doc* в файл *filename*, используя набор символов *charset* (если параметр задан) или набор символов БД (если параметр *charset* не задан).

```
PROCEDURE DBMS_XMLDOM.writeExternalDTDToBuffer
    (doc IN DOMDocument,
    buffer IN OUT VARCHAR2
    [, charset IN VARCHAR2]);
```

Записывает внешнее DTD *doc* в буфер *buffer*, используя набор символов *charset* (если параметр задан) или набор символов БД (если параметр *charset* не задан).

```
PROCEDURE DBMS_XMLDOM.writeExternalDTDToClob
    (doc IN DOMDocument,
    cl IN CLOB
    [, charset IN VARCHAR2]);
```

Записывает внешнее DTD *doc* в объект CLOB *cl*, используя набор символов *charset* (если параметр задан) или набор символов БД (если параметр *charset* не задан).

```
FUNCTION DBMS_XMLDOM.getDoctype
    (doc IN DOMDocument)
    RETURN DOMDocumentType;
```

Возвращает определение DTD, соответствующее документу *doc*.

```
FUNCTION DBMS_XMLDOM.getImplementation
    (doc IN DOMDocument)
    RETURN DOMImplementation;
```

Возвращает объект DOMImplementation, обрабатывающий документ *doc*.

```
FUNCTION DBMS_XMLDOM.getDocumentElement
    (doc IN DOMDocument)
    RETURN DOMDocumentElement;
```

Возвращает дочерний узел (элемент документа) для *doc*.

```
FUNCTION DBMS_XMLDOM.createElement
    (doc IN DOMDocument,
    tagName IN VARCHAR2)
    RETURN DOMElement;
```

Создает в документе *doc* элемент с именем *tagName* и возвращает его.

```
FUNCTION DBMS_XMLDOM.createDocumentFragment
    (doc IN DOMDocument)
    RETURN DOMDocumentFragment;
```

Создает в документе *doc* фрагмент и возвращает его.

```
FUNCTION DBMS_XMLDOM.createTextNode
    (doc IN DOMDocument,
    data IN VARCHAR2)
    RETURN DOMText;
```

Создает в документе *doc* текстовый узел с содержимым *data* и возвращает его.

```
FUNCTION DBMS_XMLDOM.createComment
    (doc IN DOMDocument,
    data IN VARCHAR2)
    RETURN DOMComment;
```

Создает в документе *doc* узел комментария с содержимым *data* и возвращает его.

```
FUNCTION DBMS_XMLDOM.createCDataSection
    (doc IN DOMDocument,
    data IN VARCHAR2)
    RETURN DOMCDATASection;
```

Создает в документе *doc* узел CdataSection с содержимым *data* и возвращает его.

```
FUNCTION DBMS_XMLDOM.createProcessingInstruction
    (doc IN DOMDocument,
    target IN VARCHAR2,
    data IN VARCHAR2)
    RETURN DOMProcessingInstruction;
```

Создает в документе *doc* узел директивы с содержимым *data* и целью *target* и возвращает его.

```
FUNCTION DBMS_XMLDOM.createAttribute
    (doc IN DOMDocument,
    data IN VARCHAR2)
    RETURN DOMAttr;
```

Создает в документе *doc* узел атрибута с содержимым *data* и возвращает его.

```
FUNCTION DBMS_XMLDOM.createEntityReference
    (doc IN DOMDocument,
    name IN VARCHAR2)
    RETURN DOMEntityReference;
```

Создает в документе *doc* узел ссылки на сущность с именем *name* и возвращает его.

```
FUNCTION DBMS_XMLDOM.getElementsByTagName
    (doc IN DOMDocument,
    tagname IN VARCHAR)
    RETURN DOMNodeList;
```

Возвращает список элементов для документа *doc* с именем тега *tagname*.

DBMS_XMLGEN

Преобразует результаты SQL-запроса в формат XML и возвращает их в виде объекта CLOB. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_XMLGEN.newContext
    (query IN VARCHAR2 | SYS_REFCURSOR)
    RETURN ctxHandle;
```

Формирует дескриптор контекста (*context handle*) ctxHandle из строки запроса или PL/SQL-курсора типа REF CURSOR.

```
PROCEDURE DBMS_XMLGEN.setRowTag
    (ctx IN ctxHandle,
    rowTag IN VARCHAR2);
```

Устанавливает для XML-документа *ctx* имя элемента, разделяющего строки, в *rowTag*.

```
PROCEDURE DBMS_XMLGEN.setRowSetTag
    (ctx IN ctxHandle,
    rowSetTag IN VARCHAR2);
```

Указывает для документа *ctx* имя корневого элемента – *rowSetTag*.

```
FUNCTION DBMS_XMLGEN.getXML[Type]
    ({ctx IN ctxHandle | sqlQuery IN VARCHAR2},
    [clobval IN OUT NCOPY clob,]
    dtdOrSchema IN NUMBER := NONE)
    RETURN {BOOLEAN | CLOB | SYS.XMLType};
```

Существует пять разновидностей данной функции:

- Дописывает строки в конец *clobval*, используя *ctx _ clobval*
- Возвращает XML-документ *ctx* в виде объекта CLOB
- Отправляет строку запроса *sqlQuery* и возвращает объект CLOB
- Возвращает XML-документ *ctx* в виде объекта типа SYS.XMLType
- Возвращает XML-документ в виде объекта типа SYS.XMLType, используя *sqlQuery*

Во всех вариантах задается параметр *dtdOrSchema*, который на настоящий момент поддерживает только значение по умолчанию.

```
FUNCTION DBMS_XMLGEN.getNumRowsProcessed
    (ctx IN ctxHandle)
    RETURN NUMBER;
```

Возвращает количество строк в XML-документе *ctx*.

```
PROCEDURE DBMS_XMLGEN.setMaxRows
    (ctx IN ctxHandle,
    maxRows IN NUMBER);
```

Задает максимальное количество строк *maxRows*, которое может быть извлечено каждым вызовом функции *getXML* из XML-документа *ctx*.

```
PROCEDURE DBMS_XMLGEN.setSkipRows
    (ctx IN ctxHandle,
```

```
    skipRows IN NUMBER);
```

Задает количество пропускаемых строк *skipRows* в XML-документе *ctx* для каждого вызова функции *getXML*.

```
PROCEDURE DBMS_XMLGEN.setConvertSpecialCharacters
    (ctx IN ctxHandle,
    conv IN BOOLEAN);
```

Указывает, следует ли преобразовывать специальные символы, возвращенные запросом для XML-документа *ctx*.

```
FUNCTION DBMS_XMLGEN.convert
    (xmlData IN {VARCHAR2 | CLOB},
    flag IN NUMBER := ENTITY_ENCODE)
    RETURN {VARCHAR2 | CLOB};
```

Преобразует XML-данные *xmlData* к их закодированному (escaped) или раскодированному (unescaped) эквиваленту и возвращает их в виде объекта типа CLOB или VARCHAR. Тип данных *xmlData* должен соответствовать возвращаемому типу. Кроме значения по умолчанию параметр *flag* может принимать значение ENTITY_DECODE.

```
PROCEDURE DBMS_XMLGEN.useItemTagsForColl
    (ctx IN ctxHandle);
```

Заменяет формат имен тегов элементов XML-документа *ctx*, принятый по умолчанию, на *itemname*_ITEM.

```
PROCEDURE DBMS_XMLGEN.restartQUERY
    (ctx IN ctxHandle);
```

Перезапускает запрос для XML-документа *ctx*.

```
PROCEDURE DBMS_XMLGEN.restartQUERY
    (ctx IN ctxHandle);
```

Закрывает контекст и освобождает ресурсы для XML-документа *ctx*.

DBMS_XMLPARSER

Содержит процедуры и функции, обеспечивающие доступ к содержимому и структуре XML-документов. Появился в Oracle9*i*.

Вызовы

```
FUNCTION DBMS_XMLPARSER.parse
    (url IN VARCHAR2)
    RETURN DOMDocument;
```

```
PROCEDURE DBMS_XMLPARSER.parse
    (p IN PARSER,
    url IN VARCHAR2);
```

Функция применяет функциональность синтаксического анализатора по умолчанию к документу, находящемуся по адресу *url*, и возвращает XML-документ. К документу, находящемуся по адресу *url*, процедурой применяется экземпляр синтаксического анализатора *p*.

```
FUNCTION DBMS_XMLPARSER.newParser
    RETURN Parser;
```

Возвращает новый экземпляр синтаксического анализатора. Функция должна вызываться в случае, если необходимо изменить поведение анализатора или прибегнуть к другим методам анализа.

```
PROCEDURE DBMS_XMLPARSER.parseBuffer
    (p IN Parser,
    doc IN VARCHAR2);
```

Выполняет синтаксический анализ документа *doc* при помощи экземпляра анализатора *p*.

```
PROCEDURE DBMS_XMLPARSER.parseClob
    (p IN Parser,
    doc IN CLOB);
```

Выполняет синтаксический анализ документа *doc* при помощи экземпляра анализатора *p*.

```
PROCEDURE DBMS_XMLPARSER.parseDTD
    (p IN Parser,
    url IN VARCHAR2,
    root IN VARCHAR2);
```

Выполняет синтаксический анализ DTD с адресом *url* для корневого элемента *root* при помощи экземпляра анализатора *p*.

```
PROCEDURE DBMS_XMLPARSER.parseDTDBuffer
    (p IN Parser,
    dtd IN VARCHAR2,
    root IN VARCHAR2);
```

Выполняет синтаксический анализ DTD, находящегося в буфере *dtd*, для корневого элемента *root* при помощи экземпляра анализатора *p*.

```
PROCEDURE DBMS_XMLPARSER.parseDTDClob
    (p IN Parser,
    dtd IN CLOB,
    root IN VARCHAR2);
```

Выполняет синтаксический анализ DTD, находящегося в объекте *dtd*, для корневого элемента *root* при помощи экземпляра анализатора *p*.

```
PROCEDURE DBMS_XMLPARSER.setBaseDir
    (p IN Parser,
    dir IN VARCHAR2);
```

Задает корневой каталог *dir* для экземпляра анализатора *p*.

```
PROCEDURE DBMS_XMLPARSER.showWarnings
    (p IN Parser,
    yes IN BOOLEAN);
```

Определяет, будут ли выводиться предупреждения экземпляром анализатора *p*.

```
PROCEDURE DBMS_XMLPARSER.setErrorLog
    (p IN Parser,
    fileName IN BOOLEAN);
```

Определяет файл вывода ошибок *fileName* для экземпляра анализатора *p*.

```
PROCEDURE DBMS_XMLPARSER.setPreserveWhitespace
    (p IN Parser,
    yes IN BOOLEAN);
```

Определяет, будут ли сохраняться пробельные символы для экземпляра анализатора *p*.

```
PROCEDURE DBMS_XMLPARSER.setValidationMode
    (p IN Parser,
    yes IN BOOLEAN);
```

Устанавливает/отменяет режим проверки корректности для экземпляра анализатора *p*.

```
FUNCTION DBMS_XMLPARSER.getValidationMode
    (p IN Parser)
    RETURN BOOLEAN;
```

Возвращает режим проверки корректности для экземпляра анализатора *p*.

```
PROCEDURE DBMS_XMLPARSER.setDoctype
    (p IN Parser,
    dtd IN DOMDocument);
```

Указывает определение типа документа *dtd*, применяемое экземпляром анализатора *p* для проверки корректности.

```
FUNCTION DBMS_XMLPARSER.getDoctype
    (p IN Parser)
    RETURN DOMDocument;
```

Возвращает определение DTD, применяемое экземпляром анализатора *p*. Вызывается после синтаксического анализа DTD.

```
FUNCTION DBMS_XMLPARSER.getDocument
    (p IN Parser)
    RETURN DOMDocument;
```

Возвращает корень древовидного документа DOM, построенный экземпляром анализатора *p*. Вызывается после синтаксического анализа DTD.

```
PROCEDURE DBMS_XMLPARSER.freeParser
    (p IN Parser);
```

Освобождает объект экземпляра анализатора *p*.

```
FUNCTION DBMS_XMLPARSER.getReleaseVersion
    RETURN VARCHAR2;
```

Возвращает версию синтаксического анализатора Oracle XML.

DBMS_XMLQUERY

Выполняет преобразование данных БД в формат XML. Корпорация Oracle рекомендует по возможности вместо данного пакета применять DBMS_XMLGEN. Появился в Oracle9*i*.

Вызовы

```
FUNCTION DBMS_XMLQUERY.newContext
    (sqlQuery IN VARCHAR2 | CLOB)
    RETURN ctxType;
```

Создает контекст запроса из *sqlQuery* и возвращает дескриптор контекста.

```
PROCEDURE DBMS_XMLQUERY.closeContext
    (ctxHandle IN ctxType);
```

Закрывает контекст, определяемый дескриптором *ctxHandle*.

```
PROCEDURE DBMS_XMLQUERY.setRowsetTag
    (ctxHandle IN ctxType,
    tag IN VARCHAR2);
```

В дескрипторе *ctxHandle* определяет тег *tag*, в который будет вложен набор данных.

```
PROCEDURE DBMS_XMLQUERY.setRowTag
    (ctxHandle IN ctxType,
    tag IN VARCHAR2);
```

В дескрипторе *ctxHandle* определяет тег *tag*, в который будут вложены записи БД.

```
PROCEDURE DBMS_XMLQUERY.setErrorTag
    (ctxHandle IN ctxType,
    tag IN VARCHAR2);
```

В дескрипторе *ctxHandle* определяет тег *tag*, в который будут вложены ошибочные документы.

```
PROCEDURE DBMS_XMLQUERY.setRowIdAttrName
    (ctxHandle IN ctxType,
    attrName IN VARCHAR2);
```

В дескрипторе *ctxHandle* определяет имя *attrName* для атрибута id тега, хранящего строки.

```
PROCEDURE DBMS_XMLQUERY.setRowAttrValue
    (ctxHandle IN ctxType,
    colName IN VARCHAR2);
```

В дескрипторе *ctxHandle* определяет имя *colName* для столбца, значение которого будет присвоено атрибуту id тега, хранящего строки.

```
PROCEDURE DBMS_XMLQUERY.setCollIdAttrName
    (ctxHandle IN ctxType,
    attrName IN VARCHAR2);
```

В дескрипторе *ctxHandle* определяет имя *attrName* для атрибута id тега, разделяющего элементы коллекции.

```
PROCEDURE DBMS_XMLQUERY.useNullAttributeIndicator
    (ctxHandle IN ctxType,
    tag IN BOOLEAN);
```

Определяет, следует ли применять атрибут XML для обозначения NULL.

```
PROCEDURE DBMS_XMLQUERY.useTypeForCollElemTag
    (ctxHandle IN ctxType,
    tag IN BOOLEAN);
```

Определяет для *ctxHandle*, следует ли использовать имя типа коллекции элементов как имя тега коллекции элементов.

```
PROCEDURE DBMS_XMLQUERY.setTagCase
    (ctxHandle IN ctxType,
    tCase IN NUMBER);
```

Для дескриптора *ctxHandle* задает регистр символов сформированных XML-тегов: 0 (как есть), 1 (нижний регистр) или 2 (верхний регистр).

```
PROCEDURE DBMS_XMLQUERY.setDateFormat
    (ctxHandle IN ctxType,
    mask IN VARCHAR2);
```

Устанавливает формат даты *mask* для *ctxHandle*.

```
PROCEDURE DBMS_XMLQUERY.setMaxRows
    (ctxHandle IN ctxType,
    rows IN NUMBER);
```

Устанавливает максимальное количество строк *ctxHandle*.

```
PROCEDURE DBMS_XMLQUERY.setSkipRows
    (ctxHandle IN ctxType,
    rows IN NUMBER);
```

Устанавливает количество пропускаемых строк для *ctxHandle*.

```
PROCEDURE DBMS_XMLQUERY.setStylesheetHeader
    (ctxHandle IN ctxType,
    uri IN VARCHAR2,
    type IN VARCHAR2 DEFAULT := 'text/xsl');
```

Указывает для *ctxHandle* заголовок таблицы стилей *uri* типа *type*.

```
PROCEDURE DBMS_XMLQUERY.setXSLT
    (ctxHandle IN ctxType,
    uri IN VARCHAR2 | CLOB,
    ref IN VARCHAR2 DEFAULT := NULL);
```

Регистрирует для применения к *ctxHandle* таблицу стилей, находящуюся по адресу *uri* или включенную как объект CLOB, на который указывает ссылка *ref* (URL для включаемых, экспортируемых и внешних сущностей).

```
PROCEDURE DBMS_XMLQUERY.setXSLTParam
    (ctxHandle IN ctxType,
    name IN VARCHAR2,
    value IN VARCHAR2);
```

Задает для *ctxHandle* значение *value* параметра *name* таблицы стилей верхнего уровня.

```
PROCEDURE DBMS_XMLQUERY.removeXSLTParam
    (ctxHandle IN ctxType,
    name IN VARCHAR2);
```

Удаляет для *ctxHandle* параметр *name* из таблицы стилей верхнего уровня.

```
PROCEDURE DBMS_XMLQUERY.setBindValue
    (ctxHandle IN ctxType,
    bindName IN VARCHAR2,
    bindValue IN VARCHAR2);
```

Устанавливает для *ctxHandle* значение *bindValue* переменной связывания *bindName*.

```
PROCEDURE DBMS_XMLQUERY.setMetaHeader
    (ctxHandle IN ctxType,
    header IN CLOB := NULL);
```

Задает метазаголовок XML *header* для *ctxHandle*.

```
PROCEDURE DBMS_XMLQUERY.setDataHeader
    (ctxHandle IN ctxType,
    header IN CLOB := NULL,
    tag IN VARCHAR2 := NULL);
```

Задает для *ctxHandle* метазаголовок XML *header*, заключенный в тег *tag*.

```
PROCEDURE DBMS_XMLQUERY.setEncodingTag
    (ctxHandle IN ctxType,
```

```
    enc IN VARCHAR2 := DB_ENCODING);
```

Указывает директиву кодирования *enc* для *ctxHandle*.

```
PROCEDURE DBMS_XMLQUERY.setRaiseException
    (ctxHandle IN ctxType,
    flag IN BOOLEAN);
```

Определяет, следует ли обрабатывать исключения, порожденные для *ctxHandle*.

```
PROCEDURE DBMS_XMLQUERY.setRaiseNoRowsException
    (ctxHandle IN ctxType,
    flag IN BOOLEAN);
```

Определяет, следует ли генерировать исключение OracleXMLNoRowsException для *ctxHandle*.

```
PROCEDURE DBMS_XMLQUERY.setSQLToXMLNamesEscaping
    (ctxHandle IN ctxType,
    flag IN BOOLEAN := TRUE);
```

Определяет для *ctxHandle*, следует ли экранировать теги XML, если имя объекта SQL не является действительным XML-идентификатором.

```
PROCEDURE DBMS_XMLQUERY.propagateOriginalException
    (ctxHandle IN ctxType,
    flag IN BOOLEAN);
```

Определяет для *ctxHandle*, следует ли передавать исключения необработанными вместо того, чтобы упаковывать их при помощи OracleXMLSQLException.

```
PROCEDURE DBMS_XMLQUERY.getExceptionContent
    (ctxHandle IN ctxType,
    errNo OUT NUMBER,
    errMsg OUT VARCHAR2);
```

Возвращает номер ошибки *errNo* и сообщение об ошибке *errMsg* для исключений *ctxHandle*. Позволяет обойти препятствие, состоящее в том, что виртуальная машина Java (JVM) генерирует собственное исключение поверх исходного исключения, делая последнее недоступным для PL/SQL.

```
FUNCTION DBMS_XMLQUERY.getDTD
    (ctxHandle IN ctxType,
    withVer IN BOOLEAN := FALSE)
    RETURN CLOB;

PROCEDURE DBMS_XMLQUERY.getDTD
    (ctxHandle IN ctxType,
    xDoc IN CLOB,
    withVer IN BOOLEAN := FALSE);
```

Формирует определение типа документа DTD для *ctxHandle* и возвращает его в виде объекта CLOB из функции или объекта CLOB *xDoc* из процедуры.

```
FUNCTION DBMS_XMLQUERY.getNumRowProcessed
    RETURN NUMBER;
```

Возвращает количество строк, обработанных запросом.

```
PROCEDURE DBMS_XMLQUERY.getVersion;
```

Выводит номер версии утилиты XSU (XML SQL Utility).

```
FUNCTION DBMS_XMLQUERY.getXML
    ({sqlQuery IN VARCHAR2 | sqlQuery IN CLOB | ctxHandle IN ctxType},
```

```
    metaType IN NUMBER := NONE)
    RETURN CLOB;

PROCEDURE DBMS_XMLQUERY.getXML
    (ctxHandle IN ctxType,
    xDoc IN CLOB,
    metaType IN NUMBER := NONE);
```

Функция может принимать запрос *sqlQuery* в виде строки VARCHAR2, объекта CLOB или *ctxHandle* и возвращает объект CLOB. Процедура получает XML-документ в *xDoc*. Параметр *metaType* может иметь значения NONE, DTD или SCHEMA.

DBMS_XMLSAVE

Реализует интерфейс между XML и базой данных. Появился в Oracle9*i*.

Вызовы

```
FUNCTION DBMS_XMLSAVE.newContext
    (targetTable IN VARCHAR2)
    RETURN ctxType;
```

Возвращает дескриптор контекста для *targetTable*.

```
PROCEDURE DBMS_XMLSAVE.closeContext
    (ctxHandle IN ctxType);
```

Закрывает и освобождает дескриптор *ctxHandle*.

```
PROCEDURE DBMS_XMLSAVE.setRowTag
    (ctxHandle IN ctxType,
    tag IN VARCHAR2);
```

Определяет для *ctxHandle* тег *tag*, в который будут вкладываться элементы XML, соответствующие записям БД.

```
PROCEDURE DBMS_XMLSAVE.setIgnoreCase
    (ctxHandle IN ctxType,
    flag IN NUMBER);
```

Указывает для *ctxHandle*, следует ли игнорировать регистр при отображении элементов XML на столбцы БД и атрибуты.

```
PROCEDURE DBMS_XMLSAVE.setDateFormat
    (ctxHandle IN ctxType,
    mask IN VARCHAR2);
```

Задает формат даты *mask* для *ctxHandle*.

```
PROCEDURE DBMS_XMLSAVE.setBatchSize
    (ctxHandle IN ctxType,
    batchsize IN NUMBER);
```

Устанавливает размер пакета *batchsize* для *ctxHandle*.

```
PROCEDURE DBMS_XMLSAVE.setCommitBatch
    (ctxHandle IN ctxType,
    batchsize IN NUMBER);
```

Устанавливает размер пакета фиксации *batchsize* для *ctxHandle*.

```
PROCEDURE DBMS_XMLSAVE.setSQLToXMLNameEscaping
    (ctxHandle IN ctxType,
```

```
    flag IN BOOLEAN := TRUE);
```

Определяет для *ctxHandle*, следует ли экранировать теги XML, если имя объекта SQL не является действительным XML-идентификатором.

```
PROCEDURE DBMS_XMLSAVE.setUpdateColumn
    (ctxHandle IN ctxType,
    colname IN VARCHAR2);
```

Добавляет столбец *colname* в список столбцов обновления для *ctxHandle*.

```
PROCEDURE DBMS_XMLSAVE.clearUpdateColumnList
    (ctxHandle IN ctxType);
```

Очищает список столбцов обновления для *ctxHandle*.

```
PROCEDURE DBMS_XMLSAVE.setPreserveWhitespace
    (ctxHandle IN ctxType,
    flag IN BOOLEAN := TRUE);
```

Определяет, будут ли сохраняться пробельные символы для *ctxHandle*.

```
PROCEDURE DBMS_XMLSAVE.setKeyColumn
    (ctxHandle IN ctxType,
    colname IN VARCHAR2);
```

Добавляет столбец *colname* в список ключевых столбцов для *ctxHandle*.

```
PROCEDURE DBMS_XMLSAVE.clearKeyColumnList
    (ctxHandle IN ctxType);
```

Очищает список ключевых столбцов для *ctxHandle*.

```
PROCEDURE DBMS_XMLSAVE.setXSLT
    (ctxHandle IN ctxType,
    uri IN {VARCHAR2 | CLOB},
    ref IN VARCHAR2 DEFAULT := NULL);
```

Регистрирует для применения к *ctxHandle* таблицу стилей, находящуюся по адресу *uri* или включенную как объект CLOB, на который указывает ссылка *ref* (URL для включаемых, экспортируемых и внешних сущностей).

```
PROCEDURE DBMS_XMLSAVE.setXSLTParam
    (ctxHandle IN ctxType,
    name IN VARCHAR2,
    value IN VARCHAR2);
```

Задает для *ctxHandle* значение *value* параметра *name* таблицы стилей верхнего уровня.

```
PROCEDURE DBMS_XMLSAVE.removeXSLTParam
    (ctxHandle IN ctxType,
    name IN VARCHAR2);
```

Удаляет для *ctxHandle* параметр *name* из таблицы стилей верхнего уровня.

```
FUNCTION DBMS_XMLSAVE.insertXML
    (ctxHandle IN ctxType,
    xDoc IN VARCHAR2 | CLOB)
    RETURN NUMBER;
```

Вставляет документ *xDoc* в *ctxHandle* и возвращает количество вставленных строк.

```
FUNCTION DBMS_XMLSAVE.updateXML
    (ctxHandle IN ctxType,
```

```
    xDoc IN VARCHAR2 | CLOB)
    RETURN NUMBER;
```

Обновляет таблицу, указанную для контекста *ctxHandle*, данными документа *xDoc* и возвращает количество обновленных строк.

```
FUNCTION DBMS_XMLSAVE.deleteXML
    (ctxHandle IN ctxType,
    xDoc IN VARCHAR2 | CLOB)
    RETURN NUMBER;
```

Удаляет из таблицы, указанной для контекста *ctxHandle*, записи, соответствующие данным документа *xDoc*, и возвращает количество удаленных строк.

```
PROCEDURE DBMS_XMLSAVE.propagateOriginalException
    (ctxHandle IN ctxType,
    flag IN BOOLEAN);
```

Определяет для *ctxHandle*, следует ли передавать исключения необработанными вместо того, чтобы упаковывать их при помощи OracleXMLSQLException.

```
PROCEDURE DBMS_XMLSAVE.getExceptionContent
    (ctxHandle IN ctxType,
    errNo OUT NUMBER,
    errMsg OUT VARCHAR2);
```

Возвращает номер ошибки *errNo* и сообщение об ошибке *errMsg* для исключений *ctxHandle*. Позволяет обойти препятствие, состоящее в том, что виртуальная машина Java (JVM) генерирует собственное исключение поверх исходного исключения, делая последнее недоступным для PL/SQL.

DBMS_XMLSCHEMA

Содержит процедуры и функции для регистрации и удаления XML-схем. Появился в Oracle9*i*.

Вызовы

```
PROCEDURE DBMS_XMLSCHEMA.registerSchema
    (schemaURL IN VARCHAR2,
    schemaDoc IN {VARCHAR2 | CLOB | BFILE | SYS.XMLType | SYS.URIType},
    local IN BOOLEAN := TRUE,
    genTypes IN BOOLEAN := TRUE,
    genbean IN BOOLEAN := FALSE,
    [genTables IN BOOLEAN := TRUE,]
    force IN BOOLEAN := FALSE,
    owner IN VARCHAR2 := NULL);
```

Регистрирует для использования сервером Oracle XML-схему *schemaDoc*, тип данных которой зависит от типа данных регистрируемой схемы, с адресом *schemaURL*. Если значение параметра *local* равно TRUE, то схема является локальной, в противном случае – глобальной. Если параметр *genTypes* установлен в TRUE, то компилятор схемы будет генерировать объектные типы. Если значение параметра *genbean* равно TRUE, то компилятор схемы будет генерировать объекты JavaBeans. Если параметр *force* установлен в TRUE, то регистрация схемы не будет порождать ошибок, а создаст вместо этого недействительную схему XML. Параметр *owner* определяет владельца БД, владеющего схемой XML. Если значение *owner* равно NULL, то схема принадлежит пользователю, регистрирующему ее. Пара-

метр *genTables* применяется только для *schemaDoc* типа VARCHAR2, и если он равен TRUE, то компилятор схемы формирует таблицы.

```
PROCEDURE DBMS_XMLSCHEMA.registerURI
    (schemaURL IN VARCHAR2,
    schemaDocURI IN VARCHAR2,
    local IN BOOLEAN := TRUE,
    genTypes IN BOOLEAN := TRUE,
    genbean IN BOOLEAN := FALSE,
    genTables IN BOOLEAN := TRUE,
    force IN BOOLEAN := FALSE,
    owner IN VARCHAR2 := NULL);
```

Регистрирует схему с уникальным идентификатором *schemaDocURI*, находящуюся по адресу *schemaURL*. Если значение параметра *local* равно TRUE, то схема является локальной, в противном случае – глобальной. Если параметр *genTypes* установлен в TRUE, то компилятор схемы будет генерировать объектные типы. Если значение параметра *genbean* равно TRUE, то компилятор схемы будет генерировать объекты JavaBeans. Если значение параметра *genTables* равно TRUE, то компилятор схемы будет генерировать таблицы. Если параметр *force* установлен в TRUE, то регистрация схемы не будет порождать ошибок, а создаст вместо этого недействительную схему XML. Параметр *owner* определяет владельца БД, который владеет схемой XML. Если значение *owner* равно NULL, то схема принадлежит пользователю, регистрирующему ее.

```
PROCEDURE DBMS_XMLSCHEMA.deleteSchema
    (schemaURL IN VARCHAR2,
    delete_option IN PLS_INTEGER := DELETE_RESTRICT);
```

Удаляет схему *schemaURL*. Параметр *delete_option* может иметь значения DELETE_RESTRICT, DELETE_INVALIDATE, DELETE_CASCADE или DELETE_CASCADE_FORCE.

```
PROCEDURE DBMS_XMLSCHEMA.generateBean
    (schemaURL IN VARCHAR2);
```

Формирует код JavaBeans для схемы с именем *schemaURL*.

```
PROCEDURE DBMS_XMLSCHEMA.compileSchema
    (schemaURL IN VARCHAR2);
```

Повторно компилирует схему *schemaURL*.

```
FUNCTION DBMS_XMLSCHEMA.generateSchema
    (schemaName IN VARCHAR2,
    typeName IN VARCHAR2,
    elementName IN VARCHAR2 := NULL,
    {schemaURL IN VARCHAR2 := NULL | recurse IN BOOLEAN := TRUE},
    annotate IN BOOLEAN := TRUE,
    embedCol IN BOOLEAN := TRUE)
    RETURN {SYS.XMLSequenceType | SYS.XMLType};
```

Формирует XML-схему из схемы БД *schemaName*, содержащей тип *typeName*. Элементом верхнего уровня является *elementName*. Параметр *schemaURL* определяет базовый URL для хранения схем. Если задан параметр *schemaURL*, то функция возвращает одну схему XMLSchema для каждой схемы БД. Параметр *recurse* указывает, следует ли также формировать схемы для всех типов, на которые ссылается указанный тип. Если значение параметра *recurse* равно TRUE, то функция возвращает все схемы в один тип XMLType. Если параметр *annotate* установлен

в TRUE, то функция помещает в XMLSchema аннотации. Если параметр *embedCol* установлен в TRUE, то функция встраивает коллекции в тип, который на них ссылается; в противном случае функция создает сложный тип.

DBMS_XPLAN

Предоставляет функцию для форматирования вывода команды EXPLAIN PLAN. Появился в Oracle9*i*. Данная функция применяется в составе команды SQL, например:

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY();
```

Вызов

```
FUNCTION DBMS_XPLAN.DISPLAY
    (table_name IN VARCHAR2 DEFAULT 'PLAN_TABLE',
    statement_id IN VARCHAR2 DEFAULT NULL,
    format IN VARCHAR2 DEFAULT 'TYPICAL');
```

Выводит информацию из плана EXPLAIN PLAN, хранящегося в таблице *table_name*, при этом дополнительно может быть указан идентификатор команды *statement_id*, для которой отображается план. Для параметра *format* допустимы следующие значения: BASIC, TYPICAL и ALL, которые соответствуют уровням детализации выводимой информации (по возрастанию), а также SERIAL – при этом значении не выводится информация о параллелизме плана.

DBMS_XSLPROCESSOR

Применяется для доступа к содержимому и структуре документов XML. Появился в Oracle9*i*.

Вызовы

```
FUNCTION DBMS_XSLPROCESSOR.newProcessor
    RETURN Processor;
```

Создает новый экземпляр XSL-процессора и возвращает указатель на него.

```
FUNCTION DBMS_XSLPROCESSOR.processXSL
    (p IN Processor,
    ss IN Stylesheet,
    {xmldoc IN DOMDoc | url IN VARCHAR})
    RETURN DOMDocumentFragment;
```

Преобразует *xmldoc* или документ, находящийся по адресу *url*, используя экземпляр процессора *p* и экземпляр таблицы стилей *ss*.

```
PROCEDURE DBMS_XSLPROCESSOR.showWarnings
    (p IN Processor,
    yes IN BOOLEAN);
```

В зависимости от значения параметра *yes* включает и выключает вывод предупреждений процессором *p*.

```
PROCEDURE DBMS_XSLPROCESSOR.setErrorLog
    (p IN Processor,
    fileName IN VARCHAR2);
```

Задает для процессора *p* местоположение журнала ошибок *fileName*.

```
FUNCTION DBMS_XSLPROCESSOR.newStylesheet
    ({xmldoc IN DOMDocument | inp IN VARCHAR2},
    ref IN VARCHAR2)
    RETURN Stylesheet;
```

Создает и возвращает новый экземпляр таблицы стилей, используя *xmldoc* или *inp* в качестве URL для создания таблицы стилей и *ref* как URL ссылки.

```
FUNCTION DBMS_XSLPROCESSOR.transformNode
    (n IN DOMNode,
    ss IN STYLESHEET)
    RETURN DOMDocumentFragment;
```

Преобразует узел *n*, основываясь на таблице стилей *ss*, и возвращает результат.

```
FUNCTION DBMS_XSLPROCESSOR.selectNodes
    (n IN DOMNode,
    pattern IN VARCHAR2)
    RETURN DOMNodeList;
```

Выбирает и возвращает узлы дерева XML с корневыми элементами *n*, соответствующими образцу *pattern*.

```
FUNCTION DBMS_XSLPROCESSOR.selectNodes
    (n IN DOMNode,
    pattern IN VARCHAR2)
    RETURN DOMNode;
```

Выбирает и возвращает первый узел дерева XML с корневым элементом *n*, соответствующим образцу *pattern*.

```
PROCEDURE DBMS_XSLPROCESSOR.valueOf
    (n IN DOMNode,
    pattern IN VARCHAR2
    val OUT VARCHAR2);
```

Выбирает первый узел дерева XML с корневым элементом *n*, соответствующим образцу *pattern*, и возвращает его в параметре *val*.

```
PROCEDURE DBMS_XSLPROCESSOR.setParam
    (ss IN STYLESHEET,
    name IN VARCHAR2,
    value IN VARCHAR2);
```

Устанавливает параметр *name* таблицы стилей *ss* в значение *value*.

```
PROCEDURE DBMS_XSLPROCESSOR.setParam
    (ss IN STYLESHEET,
    name IN VARCHAR2);
```

Удаляет параметр *name* из таблицы стилей *ss*.

```
PROCEDURE DBMS_XSLPROCESSOR.resetParams
    (ss IN Stylesheet);
```

Сбрасывает значения параметров таблицы стилей верхнего уровня *ss*.

```
PROCEDURE DBMS_XSLPROCESSOR.freeStylesheet
    (ss IN Stylesheet);
```

Освобождает таблицу стилей верхнего уровня *ss*.

```
PROCEDURE DBMS_XSLPROCESSOR.freeProcessor
    (p IN Processor);
```

Освобождает процессор *p*.

DEBUG_EXTPROC

Запускает агент *extproc* для сеанса и способствует отладке внешних процедур. Для инсталляции данного пакета необходимо запустить сценарий *DBGEXTP.SQL*. Появился в Oracle9*i*.

Вызов

```
PROCEDURE DEBUG_EXTPROC;
```

Запускает процесс *extproc* для сеанса, что позволяет получить идентификатор выполняемого процесса.

OUTLN_PKG

В версии Oracle9*i* этот пакет был заменен на DBMS_OUTLINE.

UTL_COLL

Позволяет PL/SQL-программам использовать для запроса и обновления указатели коллекций. Появился в Oracle9*i*.

Вызов

```
FUNCTION UTL_COLL.IS_LOCATOR
    (collection IN ANY)
    RETURN BOOLEAN;
```

Возвращает TRUE, если элемент коллекции *collection* является указателем.

UTL_ENCODE

Содержит функции кодирования данных типа RAW для передачи между узлами. Появился в Oracle9*i*.

Вызовы

```
FUNCTION UTL_ENCODE.BASE64_ENCODE
    (r IN RAW)
    RETURN RAW;
```

Кодирует двоичное представление значения *r* типа RAW в элементы формата base64 и возвращает его как строку RAW.

```
FUNCTION UTL_ENCODE.BASE64_DECODE
    (r IN RAW)
    RETURN RAW;
```

Читает строку RAW в формате base64, декодирует ее и возвращает ее исходное значение RAW.

```
FUNCTION UTL_ENCODE.UUENCODE
    (r IN RAW,
    type IN PLS_INTEGER DEFAULT 1,
    filename IN VARCHAR2 DEFAULT NULL,
    permission IN VARCHAR2 DEFAULT NULL)
    RETURN RAW;
```

Кодирует значение *r* типа RAW и возвращает его как строку UUENCODE. Параметр *type* может иметь значения COMPLETE (умолчание), HEADER_PIECE, MIDDLE_PIECE или END_PIECE. Необязательный параметр *filename* определяет имя файла *uuencode*.

```
FUNCTION UTL_ENCODE.UUDECODE
    (r IN RAW)
    RETURN RAW;
```

Декодирует строку *r* типа RAW из формата UUENCODE и возвращает исходное значение типа RAW.

```
FUNCTION UTL_ENCODE.QUOTED_PRINTABLE_ENCODE
    (r IN RAW)
    RETURN RAW;
```

Кодирует *r* и возвращает его как печатаемую строковую константу.

```
FUNCTION UTL_ENCODE.QUOTED_PRINTABLE_DECODE
    (r IN RAW)
    RETURN RAW;
```

Декодирует печатаемую строковую константу *r* и возвращает исходное значение типа RAW.

UTL_FILE

Позволяет программам PL/SQL читать и записывать файлы операционной системы на сервере, где хранится БД Oracle.

Вызовы

```
PROCEDURE UTL_FILE.FCLOSE
    (file IN OUT FILE_TYPE);
```

Закрывает файл, определяемый дескриптором *file*, и устанавливает значение идентификатора файла в NULL.

```
PROCEDURE UTL_FILE.FCLOSE_ALL;
```

Закрывает все открытые файлы, но при этом поля идентификаторов для дескрипторов не устанавливаются в NULL.

```
PROCEDURE UTL_FILE.FCOPY
    (location IN VARCHAR2,
    filename IN VARCHAR2,
    dest_dir IN VARCHAR2,
    dest_file IN VARCHAR2,
    start_line IN PLS_INTEGER DEFAULT 1,
    end_line IN PLS_INTEGER DEFAULT NULL);
```

Копирует файл *filename* из каталога *location* в файл *dest_file* в каталоге *dest_dir*. Параметры *start_line* и *end_line* задаются для копирования части файла. Появилась в Oracle9*i*.

```
PROCEDURE UTL_FILE.FFLUSH
    (file IN FILE_TYPE
    [, invalid_maxlinesize EXCEPTION]#);
```

Выполняет незамедлительную запись данных из буфера (при их наличии) в файл с дескриптором *file*. Параметр *invalid_maxlinesize* (появился в Oracle9*i*) может иметь значения INVALID_FILENAME, INVALID_OPERATION или WRITE_ERROR.

```
PROCEDURE UTL_FILE.FGETATTR
    (location IN VARCHAR2,
    filename IN VARCHAR2,
    exists OUT BOOLEAN,
    file_length OUT NUMBER,
    blocksize OUT NUMBER);
```

Возвращает атрибуты *exists*, *file_length* и *blocksize* для файла *filename* в каталоге *location*. Появилась в Oracle9*i*.

```
FUNCTION UTL_FILE.FGETPOS
    (fileid IN FILE_TYPE)
    RETURN PLS_INTEGER;
```

Возвращает текущее смещение (в байтах) для файла *fileid*. Появилась в Oracle9*i*.

```
PROCEDURE UTL_FILE.FREMOVE
    (location IN VARCHAR2,
    filename IN VARCHAR2);
```

Удаляет файл *filename* из каталога *location*. Появилась в Oracle9*i*.

```
PROCEDURE UTL_FILE.FRENAME
    (location IN VARCHAR2,
    filename IN VARCHAR2,
    dest_dir IN VARCHAR2,
    dest_file IN VARCHAR2,
    overwrite IN BOOLEAN DEFAULT FALSE);
```

Перемещает файл *filename* из каталога *location* в файл *dest_file* в каталоге *dest_dir*. Если значение параметра *overwrite* равно TRUE, то существующий файл с тем же именем будет перезаписан. Появилась в Oracle9*i*.

```
PROCEDURE UTL_FILE.FSEEK
    (fid IN UTL_FILE.FILE_TYPE,
    absolute_offset IN PLS_INTEGER DEFAULT NULL,
    relative_offset IN PLS_INTEGER DEFAULT NULL);
```

Перемещает указатель чтения файла *fid* на абсолютную позицию (в байтах) *absolute_offset* или смещает его на относительную величину (в байтах) *relative_offset*, которая может быть положительной, отрицательной или NULL. Появилась в Oracle9*i*.

```
FUNCTION UTL_FILE.FOPEN
    (location IN VARCHAR2,
    filename IN VARCHAR2,
    open_mode IN VARCHAR2
    [, max_linesize IN BINARY_INTEGER]#);
    RETURN FILE_TYPE;
```

Возвращает дескриптор файла типа UTL_FILE.FILE_TYPE в случае успешного открытия файла *filename* в каталоге *location* в режиме *open_mode* или генерирует исключение. Значениями *open_mode* могут быть:

R – открыть файл только для чтения.

W – открыть файл для чтения и записи с заменой содержимого.

A – открыть файл для чтения и записи и добавлять данные в конец файла.

Допустимыми значениями параметра *location* являются каталоги, указанные в параметре инициализации Oracle UTL_FILE_DIR. Параметр *max_linesize* (появился в Oracle9*i*) определяет максимальное количество символов в строке файла (включая символ новой строки).

```
FUNCTION UTL_FILE.FOPEN_NCHAR
    (location IN VARCHAR2,
    filename IN VARCHAR2,
    open_mode IN VARCHAR2
    [, max_linesize IN BINARY_INTEGER]#);
    RETURN FILE_TYPE;
```

Аналогична FOPEN, но применяется для файлов Unicode. Появилась в Oracle9*i*.

```
PROCEDURE UTL_FILE.GET_LINE
    (file IN FILE_TYPE,
    buffer OUT VARCHAR2
    [,linesize IN NUMBER]#
    [,len IN PLS_INTEGER DEFAULT NULL]#);
```

Считывает следующую строку файла *file* в буфер *buffer*. Порождает исключение NO_DATA_FOUND, когда файл считан до конца, и VALUE_ERROR – когда размер буфера слишком мал для данных. Параметры *linesize* (максимальное количество байт для считывания) и *len* (количество прочитанных байт файла) появились в Oracle9*i*.

```
PROCEDURE UTL_FILE.GET_LINE_NCHAR
    (file IN FILE_TYPE,
    buffer OUT VARCHAR2
    [,linesize IN NUMBER]#
    [,len IN PLS_INTEGER DEFAULT NULL]#);
```

Аналогична GET_LINE, но применяется для файлов Unicode. Появилась в Oracle9*i*.

```
FUNCTION UTL_FILE.IS_OPEN
    (file IN FILE_TYPE)
    RETURN BOOLEAN;
```

Возвращает TRUE, если файл с дескриптором *file* в настоящее время открыт в любом режиме, и FALSE – в противном случае.

```
FUNCTION UTL_FILE.GET_RAW
    (fid IN UTL_FILE.FILE_TYPE,
    r OUT NOCOPY RAW,
    len IN PLS_INTEGER DEFAULT NULL);
```

Возвращает строку RAW *r* из файла *fid*. Параметр *len* определяет количество прочитанных байт файла. Появилась в Oracle9*i*.

```
PROCEDURE UTL_FILE.NEW_LINE
    (file IN FILE_TYPE,
    lines IN NATURAL := 1);
```

Помещает *lines* символов новой строки в файл *file*.

```
PROCEDURE UTL_FILE.PUT
    (file IN FILE_TYPE,
    buffer IN VARCHAR2);
```

Помещает данные из буфера *buffer* в файл *file*, не добавляя символ конца строки.

```
PROCEDURE UTL_FILE.PUT_NCHAR
    (file IN FILE_TYPE,
    buffer IN VARCHAR2);
```

Аналогична PUT, но применяется для файлов Unicode. Появилась в Oracle9*i*.

```
PROCEDURE UTL_FILE.PUT_RAW
    (fid IN UTL_FILE.FILE_TYPE,
```

```
    r IN RAW
    [,autoflush IN BOOLEAN DEFAULT FALSE]#);
```

Записывает строку RAW *r* в файл *fid*. Если параметр *autoflush* (появился в Oracle9*i*) установлен в TRUE, то после записи значения выполняется запись буфера на диск.

```
PROCEDURE UTL_FILE.PUT_LINE
    (file IN FILE_TYPE,
    buffer IN VARCHAR2
    [, autoflush IN BOOLEAN DEFAULT FALSE]#);
```

Помещает данные буфера *buffer* в файл *file* с добавлением символа начала строки. Если параметр *autoflush* (появился в Oracle9*i*) установлен в TRUE, то после записи значения выполняется запись буфера на диск.

```
PROCEDURE UTL_FILE.PUT_LINE_NCHAR
    (file IN FILE_TYPE,
    buffer IN VARCHAR2);
```

Аналогична PUT_LINE, но применяется для файлов Unicode. Появилась в Oracle9*i*.

```
PROCEDURE UTL_FILE.PUTF
    (file IN FILE_TYPE,
    format IN VARCHAR2,
    arg1 IN VARCHAR2 DEFAULT NULL,
    arg2 IN VARCHAR2 DEFAULT NULL,
    arg3 IN VARCHAR2 DEFAULT NULL,
    arg4 IN VARCHAR2 DEFAULT NULL,
    arg5 IN VARCHAR2 DEFAULT NULL);
```

Записывает форматированную строку в файл *file*, принимая за образец строку формата *format*, заменяя до пяти элементов формата %s значениями аргументов *arg1* – *arg5*. Параметр *format* может включать в себя следующие элементы: любой текстовый литерал, %s – замена аргумента *argN* (до пяти вхождений) или \n для символа новой строки (разрешены в любом количестве).

```
PROCEDURE UTL_FILE.PUTF_NCHAR
    (file IN FILE_TYPE,
    format IN VARCHAR2,
    arg1 IN VARCHAR2 DEFAULT NULL,
    arg2 IN VARCHAR2 DEFAULT NULL,
    arg3 IN VARCHAR2 DEFAULT NULL,
    arg4 IN VARCHAR2 DEFAULT NULL,
    arg5 IN VARCHAR2 DEFAULT NULL);
```

Аналогична PUTF, но применяется для файлов Unicode. Появилась в Oracle9*i*.

UTL_HTTP

Содержит процедуры и функции для обращения к HTTP из SQL и PL/SQL. В версии Oracle8 данный пакет содержал только функции REQUEST и REQUEST_PIECES.

Вызовы

В разделе приведены вызовы UTL_HTTP, относящиеся к различным категориям: простым выборкам HTTP, параметрам сеансов HTTP, HTTP-запросам, HTTP-ответам, файлам «cookie» HTTP, постоянным соединениям HTTP и условиям ошибок HTTP.

Простые выборки HTTP

```
FUNCTION UTL_HTTP.REQUEST
    (url IN VARCHAR2,
    proxy IN VARCHAR2 DEFAULT NULL
    [,wallet_path IN VARCHAR2 DEFAULT NULL]#
    [,wallet_password IN VARCHAR2 DEFAULT NULL]#)
    RETURN VARCHAR2;
```

Возвращает первые 2000 байт ресурса с адресом *url*, работая через прокси-сервер *proxy* (если он указан); это необязательный параметр, появившийся в Oracle8*i*. Может содержать параметры *wallet_path* и *wallet_password*, оба они появились в Oracle9*i*.

```
FUNCTION UTL_HTTP.REQUEST_PIECES
    (url IN VARCHAR2,
    max_pieces IN NATURAL DEFAULT 32767,
    proxy IN VARCHAR2 DEFAULT NULL
    [,wallet_path IN VARCHAR2 DEFAULT NULL]#
    [,wallet_password IN VARCHAR2 DEFAULT NULL]#)
    RETURN VARCHAR2;
```

Возвращает *max_pieces* 2000-байтных порций ресурса с адресом *url*, работая через необязательный прокси-сервер *proxy*. Может содержать параметры *wallet_path* и *wallet_password*, оба они появились в Oracle9*i*.

Параметры сеанса HTTP

```
PROCEDURE UTL_HTTP.SET_PROXY
    (proxy IN VARCHAR2,
    no_proxy_domains IN VARCHAR2);
```

Указывает прокси-сервер *proxy*, а также может определять домены, для которых прокси-сервер не будет применяться, в параметре *no_proxy_domains*.

```
PROCEDURE UTL_HTTP.GET_PROXY
    (proxy OUT NOCOPY VARCHAR2,
    no_proxy_domains OUT NOCOPY VARCHAR2);
```

Возвращает прокси-сервер *proxy* и перечень доменов, для которых прокси-сервер не будет использоваться, в параметре *no_proxy_domains*.

```
PROCEDURE UTL_HTTP.GET_COOKIE_SUPPORT
    (enable IN BOOLEAN,
    max_cookies IN PLS_INTEGER DEFAULT 300,
    max_cookies_per_site IN PLS_INTEGER DEFAULT 20);
```

Определяет, разрешено ли использование файлов «cookie», и задает их максимальное количество для сеанса (*max_cookies*) и для базы данных (*max_cookies_per_site*).

```
PROCEDURE UTL_HTTP.SET_COOKIE_SUPPORT
    (enable OUT BOOLEAN,
    max_cookies OUT PLS_INTEGER,
    max_cookies_per_site OUT PLS_INTEGER);
```

Возвращает информацию о том, разрешено ли использование файлов «cookie», а также их максимальное количество для сеанса (*max_cookies*) и для базы данных (*max_cookies_per_site*).

```
PROCEDURE UTL_HTTP.SET_FOLLOW_REDIRECT
    (max_redirects IN PLS_INTEGER DEFAULT 3);
```

Задает максимальное количество раз, которое UTL_HTTP будет следовать инструкции переадресации HTTP.

```
PROCEDURE UTL_HTTP.GET_FOLLOW_REDIRECT
    (max_redirects OUT PLS_INTEGER);
```

Возвращает максимальное количество раз, которое UTL_HTTP будет следовать инструкции переадресации HTTP.

```
PROCEDURE UTL_HTTP.SET_BODY_CHARSET
    (charset IN VARCHAR2 DEFAULT NULL);
```

Задает набор символов по умолчанию для тела всех последующих запросов HTTP.

```
PROCEDURE UTL_HTTP.GET_BODY_CHARSET
    (charset OUT VARCHAR2 NOCOPY);
```

Возвращает набор символов по умолчанию для тела всех последующих запросов HTTP.

```
PROCEDURE UTL_HTTP.SET_PERSISTENT_CONN_SUPPORT
    (enable IN BOOLEAN,
    max_conns IN PLS_INTEGER DEFAULT 0);
```

Указывает, разрешены ли постоянные соединения (параметр *enable*) и в каком количестве (*max_conns*).

```
PROCEDURE UTL_HTTP.GET_PERSISTENT_CONN_SUPPORT
    (enable OUT BOOLEAN,
    max_conns OUT PLS_INTEGER);
```

Возвращает информацию о том, разрешены ли постоянные соединения (параметр *enable*) и в каком количестве (*max_conns*).

```
PROCEDURE UTL_HTTP.SET_RESPONSE_ERROR_CHECK
    (enable IN BOOLEAN DEFAULT FALSE);
```

Определяет, будет ли функция GET_RESPONSE порождать исключение при ошибке веб-сервера.

```
PROCEDURE UTL_HTTP.GET_RESPONSE_ERROR_CHECK
    (enable IN BOOLEAN);
```

Возвращает информацию о том, будет ли функция GET_RESPONSE порождать исключение при ошибке веб-сервера.

```
PROCEDURE UTL_HTTP.SET_DETAILED_EXCP_SUPPORT
    (enable IN BOOLEAN DEFAULT FALSE);
```

Определяет, будет ли функция GET_RESPONSE порождать детальное исключение при ошибке веб-сервера.

```
PROCEDURE UTL_HTTP.GET_DETAILED_EXCP_SUPPORT
    (enable IN BOOLEAN);
```

Возвращает информацию о том, будет ли функция GET_RESPONSE порождать детальное исключение при ошибке web-сервера.

```
PROCEDURE UTL_HTTP.SET_WALLET
    (path IN VARCHAR2,
    password IN VARCHAR2 DEFAULT NULL);
```

Задает путь к накопителю Oracle и отправляет пароль для открытия накопителя, который применяется всеми HTTP-запросами, выполняемыми по SSL.

```
PROCEDURE UTL_HTTP.SET_TRANSFER_TIMEOUT
    (timeout IN PLS_INTEGER DEFAULT 60);
```

Задает таймаут по умолчанию для всех последующих запросов UTL_HTTP, пытающихся получить HTTP-ответы от веб- или прокси-сервера.

HTTP-запросы

```
FUNCTION UTL_HTTP.BEGIN_REQUEST
    (url IN VARCHAR2,
    method IN VARCHAR2 DEFAULT 'GET',
    http_version IN VARCHAR2 DEFAULT NULL)
    RETURN REQ;
```

Начинает новый HTTP-запрос к ресурсу *url* для метода *method* с применением версии протокола *http_version*. Возвращает дескриптор запроса.

```
PROCEDURE UTL_HTTP.SET_HEADER
    (r IN OUT NOCOPY REQ,
    name IN VARCHAR2,
    value IN VARCHAR2);
```

Задает значение *value* для заголовка *name* запроса *r*.

```
PROCEDURE UTL_HTTP.SET_AUTHENTICATION
    (r IN OUT NOCOPY req,
    username IN VARCHAR2,
    password IN VARCHAR2,
    scheme IN VARCHAR2 DEFAULT 'Basic',
    for_proxy IN BOOLEAN DEFAULT FALSE);
```

Указывает имя пользователя и пароль для HTTP-аутентификации запроса *r*. Если параметр *for_proxy* установлен в TRUE, то речь идет об аутентификации для прокси-сервера.

```
PROCEDURE UTL_HTTP.SET_COOKIE_SUPPORT
    (r IN OUT NOCOPY REQ,
    enable IN BOOLEAN DEFAULT TRUE);
```

Разрешает поддержку файлов «cookie» для запроса r.

```
PROCEDURE UTL_HTTP.SET_FOLLOW_REDIRECT
    (r IN OUT NOCOPY REQ,
    max_redirects IN PLS_INTEGER DEFAULT 3);
```

Изменяет максимальное количество переадресаций *max_redirects*, которое запрос *r* унаследовал из параметров сеанса.

```
PROCEDURE UTL_HTTP.SET_BODY_CHARSET
    (r IN OUT NOCOPY REQ,
    charset IN VARCHAR2 DEFAULT NULL);
```

Изменяет набор символов, унаследованный запросом *r* из параметров сеанса по умолчанию.

```
PROCEDURE UTL_HTTP.SET_PERSISTENT_CONN_SUPPORT
    (r IN OUT NOCOPY REQ,
    enable IN BOOLEAN DEFAULT TRUE);
```

Разрешает поддержку постоянного соединения для запроса *r*.

```
PROCEDURE UTL_HTTP.WRITE_TEXT
    (r IN OUT NOCOPY REQ,
    data IN VARCHAR2);
```

Записывает данные в тело HTTP-запроса *r*.

```
PROCEDURE UTL_HTTP.WRITE_LINE
    (r IN OUT NOCOPY REQ,
    data IN VARCHAR2);
```

Записывает данные в тело HTTP-запроса *r* и завершает строку символом новой строки.

```
PROCEDURE UTL_HTTP.WRITE_RAW
    (r IN OUT NOCOPY REQ,
    data IN RAW);
```

Записывает данные типа RAW в тело HTTP-запроса *r*.

```
PROCEDURE UTL_HTTP.END_REQUEST
    (r IN OUT NOCOPAY REQ);
```

Завершает запрос *r*.

HTTP-ответы

```
FUNCTION UTL_HTTP.GET_RESPONSE
    (r IN OUT NOCOPY REQ)
    RETURN resp;
```

Возвращает указатель на ответ на запрос *r*.

```
FUNCTION UTL_HTTP.GET_HEADER_COUNT
    (r IN OUT NOCOPY REQ)
    RETURN PLS_INTEGER;
```

Возвращает количество заголовков ответа *r*.

```
PROCEDURE UTL_HTTP.GET_HEADER
    (r IN OUT NOCOPY RESP,
    n IN PLS_INTEGER,
    name OUT NOCOPY VARCHAR2,
    value OUT NOCOPY VARCHAR2);
```

Возвращает имя и значение *n*-го заголовка ответа *r*.

```
PROCEDURE UTL_HTTP.GET_HEADER_BY_NAME
    (r IN OUT NOCOPY RESP,
    name IN VARCHAR2,
    value OUT NOCOPY VARCHAR2,
    n IN PLS_INTEGER DEFAULT 1);
```

Возвращает значение *n*-го заголовка с именем *name* ответа *r*.

```
PROCEDURE UTL_HTTP.GET_AUTHENTICATION
    (r IN OUT NOCOPY RESP,
    scheme OUT VARCHAR2,
    realm OUT VARCHAR2,
    for_proxy IN BOOLEAN DEFAULT FALSE);
```

Возвращает схему *scheme* и область *realm* для HTTP-аутентификации ответа *r*. Если параметр *for_proxy* установлен в TRUE, то речь идет о доступе к прокси-серверу.

```
PROCEDURE UTL_HTTP.SET_BODY_CHARSET
    (r IN OUT NOCOPY RESP,
    charset IN VARCHAR2 DEFAULT NULL);
```

Задает набор символов для ответа *r* для случая, когда содержимое имеет тип 'TEXT', но в заголовке 'Content-Type' набор символов не указан.

```
PROCEDURE UTL_HTTP.READ_TEXT
    (r IN OUT NOCOPY RESP,
```

```
    data OUT NOCOPY VARCHAR2,
    len IN PLS_INTEGER DEFAULT NULL);
```

Возвращает в параметре *data* тело ответа *r*. Параметр *len* определяет максимальное количество считываемых символов. Если параметр *len* установлен в NULL, то заполняется весь буфер.

```
PROCEDURE UTL_HTTP.READ_LINE
    (r IN OUT NOCOPY RESP,
    data OUT NOCOPY VARCHAR2,
    remove_crlf IN BOOLEAN DEFAULT FALSE);
```

Возвращает в параметре *data* тело ответа *r* вплоть до конца строки. Если параметр *remove_crlf* установлен в TRUE, то удаляются символы новой строки.

```
PROCEDURE UTL_HTTP.READ_RAW
    (r IN OUT NOCOPY RESP,
    data OUT NOCOPY RAW,
    len IN PLS_INTEGER DEFAULT NULL);
```

Возвращает в параметре *data* тело ответа *r* в виде данных типа RAW. Параметр *len* определяет максимальное количество считываемых символов. Если параметр *len* равен NULL, то заполняется весь буфер.

```
PROCEDURE UTL_HTTP.END_RESPONSE
    (r IN OUT NOCOPY RESP);
```

Завершает ответ *r*.

Файлы cookie HTTP

```
FUNCTION UTL_HTTP.GET_COOKIE_COUNT
   RETURN PLS_INTEGER;
```

Возвращает количество файлов cookie, поддерживаемых на настоящий момент пакетом UTL_HTTP для всех серверов.

```
PROCEDURE UTL_HTTP.GET_COOKIES
    (cookies IN OUT NOCOPY COOKIE_TABLE);
```

Возвращает файлы cookie, поддерживаемые на настоящий момент пакетом UTL_HTTP для всех серверов.

```
PROCEDURE UTL_HTTP.ADD_COOKIES
    (cookies IN NOCOPY COOKIE_TABLE);
```

Добавляет файл «cookie» для поддержки пакетом UTL_HTTP.

```
PROCEDURE UTL_HTTP.CLEAR_COOKIES;
```

Очищает файлы cookie, поддерживаемые на настоящий момент пакетом UTL_HTTP для всех серверов.

Постоянные соединения HTTP

```
FUNCTION UTL_HTTP.GET_PERSISTENT_CONN_COUNT
    RETURN PLS_INTEGER;
```

Возвращает количество постоянных соединений, поддерживаемых пакетом UTL_HTTP.

```
PROCEDURE UTL_HTTP.GET_PERSISTENT_CONN
    (connections IN OUT NOCOPY CONNECTION_TABLE);
```

Возвращает постоянное соединение, поддерживаемое пакетом UTL_HTTP.

```
PROCEDURE UTL_HTTP.CLOSE_PERSISTENT_CONN
    (conn IN connection);
```

Закрывает постоянное соединение *conn*.

```
PROCEDURE UTL_HTTP.CLOSE_PERSISTENT_CONNS
    (host IN VARCHAR2 DEFAULT NULL,
    port IN PLS_INTEGER DEFAULT NULL,
    proxy_host IN VARCHAR2 DEFAULT NULL,
    proxy_port IN PLS_INTEGER DEFAULT NULL,
    ssl IN BOOLEAN DEFAULT NULL);
```

Закрывает постоянные соединения для порта *port* хоста *host* или порта *proxy_port* прокси-сервера *proxy_host*. Если параметр *ssl* установлен в TRUE, то закрывается постоянное SSL-соединение. Вызов без каких бы то ни было параметров приводит к закрытию всех постоянных соединений.

Условия ошибок HTTP

```
FUNCTION UTL_HTTP.GET_DETAILED_SQLCODE
    RETURN PLS_INTEGER;
```

Извлекает подробный SQLCODE для последнего исключения.

```
FUNCTION UTL_HTTP.GET_DETAILED_SQLERRM
    RETURN VARCHAR2;
```

Извлекает подробное сообщение об ошибке SQLERRM для последнего исключения.

UTL_INADDR

Содержит функции для поддержки интернет-адресации. Появился в Oracle9*i*.

Вызовы

```
FUNCTION UTL_INADDR.GET_HOST_NAME
    (ip IN VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Возвращает имя локального или удаленного хоста с IP-адресом *ip*.

```
FUNCTION UTL_INADDR.GET_HOST_ADDRESS
    (name IN VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Возвращает адрес хоста *name*. Если *name* содержит NULL, то возвращает адрес локального хоста.

UTL_RAW

Содержит функции для доступа и работы с данными типа RAW. Данные функции выполняют преобразования, деления, объединения и побитовые операции над данными RAW.

Вызовы

```
FUNCTION UTL_RAW.BIT_AND
    (r1 IN RAW,
    r2 IN RAW)
    RETURN RAW;
```

Возвращает результат поразрядного логического И для *r1* и *r2*.

```
FUNCTION UTL_RAW.BIT_COMPLEMENT
    (r1 IN RAW,
    r2 IN RAW)
    RETURN RAW;
```

Возвращает результат поразрядного логического дополнения для *r1* и *r2*.

```
FUNCTION UTL_RAW.BIT_OR
    (r1 IN RAW,
    r2 IN RAW)
    RETURN RAW;
```

Возвращает результат поразрядного логического ИЛИ для *r1* и *r2*.

```
FUNCTION UTL_RAW.BIT_XOR
    (r1 IN RAW,
    r2 IN RAW)
    RETURN RAW;
```

Возвращает результат поразрядного логического исключающего ИЛИ для *r1* и *r2*.

```
FUNCTION UTL_RAW.CAST_FROM_BINARY_INTEGER
    (n IN BINARY_INTEGER,
    endianess IN PLS_INTEGER DEFAULT BIG_ENDIAN)
    RETURN RAW;
```

Возвращает двоичное представление *n* как значение типа RAW. Параметр *endianess* имеет значение BIG_ENDIAN или LITTLE_ENDIAN. Появилась в Oracle9*i*.

```
FUNCTION UTL_RAW.CAST_FROM_NUMBER
    (n IN BINARY_INTEGER,
    include_length IN BOOLEAN)
    RETURN RAW;
```

Возвращает двоичное представление *n* как значение типа RAW. Если параметр *include_length* установлен в FALSE, то возвращается значение, длина которого ограничена максимумом в 21 байт. Появилась в Oracle9*i*.

```
FUNCTION UTL_RAW.CAST_TO_BINARY_INTEGER
    (r IN RAW,
    endianess IN PLS_INTEGER DEFAULT BIG_ENDIAN)
    RETURN BINARY_INTEGER;
```

Возвращает двоичное представление *r*. Параметр *endianess* имеет значение BIG_ENDIAN или LITTLE_ENDIAN. Появилась в Oracle9*i*.

```
FUNCTION UTL_RAW.CAST_TO_NUMBER
    (r IN RAW,
    include_length IN BOOLEAN)
    RETURN RAW;
```

Возвращает двоичное представление *r*. Если параметр *include_length* установлен в FALSE, то возвращается значение, длина которого ограничена максимумом в 21 байт плюс байт длины (который будет первым байтом результата). Появилась в Oracle9*i*.

```
FUNCTION UTL_RAW.CAST_TO_RAW
    (c IN VARCHAR2)
    RETURN RAW;
```

Преобразует *c* типа VARCHAR2 в RAW, меняя только тип данных, но не сами данные.

```
FUNCTION UTL_RAW.CAST_TO_VARCHAR2
    (r IN RAW)
    RETURN VARCHAR2;
```

Преобразует *c* типа RAW в VARCHAR2, меняя только тип данных, но не сами данные.

```
FUNCTION UTL_RAW.COMPARE
    (r1 IN RAW,
    r2 IN RAW,
    pad IN RAW DEFAULT NULL)
    RETURN NUMBER;
```

Возвращает 0, если *r1* и *r2* идентичны. Возвращает первую позицию (в байтах), в которой существует различие *r1* и *r2*. Если *r1* и *r2* имеют разные длины, то более короткое значение дополняется справа данными из *pad*.

```
FUNCTION UTL_RAW.CONCAT
    (r1 IN RAW DEFAULT NULL,
    r2 IN RAW DEFAULT NULL,
    r3 IN RAW DEFAULT NULL,
    r4 IN RAW DEFAULT NULL,
    r5 IN RAW DEFAULT NULL,
    r6 IN RAW DEFAULT NULL,
    r7 IN RAW DEFAULT NULL,
    r8 IN RAW DEFAULT NULL,
    r9 IN RAW DEFAULT NULL,
    r10 IN RAW DEFAULT NULL,
    r11 IN RAW DEFAULT NULL,
    r12 IN RAW DEFAULT NULL)
    RETURN RAW;
```

Возвращает результат конкатенации значений *r1* – *r12*. Длина результата не должна превышать 32 Кбайт. Параметры *r3* – *r12* являются необязательными.

```
FUNCTION UTL_RAW.CONVERT
    (r IN RAW,
    to_charset IN VARCHAR2,
    from_charset IN VARCHAR2)
    RETURN RAW;
```

Возвращает *r*, преобразованное из набора символов *from_charset* в набор символов *to_charset*. Параметры *from_charset* и *to_charset* определяют наборы символов NLS.

```
FUNCTION UTL_RAW.COPIES
    (r IN RAW,
    n IN NUMBER)
    RETURN RAW;
```

Конкатенирует *n* экземпляров *r* и возвращает результат.

```
FUNCTION UTL_RAW.LENGTH
    (r IN RAW)
    RETURN NUMBER;
```

Возвращает количество байт в *r*.

```
FUNCTION UTL_RAW.OVERLAY
    (overlay_str IN RAW,
    target IN RAW,
    pos IN BINARY_INTEGER DEFAULT 1,
```

```
    len IN BINARY_INTEGER DEFAULT NULL,
    pad IN RAW DEFAULT NULL)
    RETURN RAW;
```

Накладывает на *target* строку *overlay_str* начиная с позиции *pos* (в байтах) в *target* на протяжении *len* байт. Возвращает *target*. Если значение *pos* превышает длину *target*, то недостающая часть *target* заполняется данными из *pad*.

```
FUNCTION UTL_RAW.REVERSE
    (r IN RAW)
    RETURN RAW;
```

Возвращает байты *r* в обратном порядке.

```
FUNCTION UTL_RAW.SUBSTR
    (r IN RAW,
    pos IN BINARY_INTEGER,
    len IN BINARY_INTEGER DEFAULT NULL)
    RETURN RAW;
```

Возвращает часть *r* начиная с позиции *pos* длиной *len* байт.

```
FUNCTION UTL_RAW.TRANSLATE
    (r IN RAW,
    from_set IN RAW,
    to_set IN RAW)
    RETURN RAW;
```

Возвращает содержимое *r*, преобразуя байты, найденные в *from_set*, в находящиеся в той же позиции байты *to_set*. Если набор *from_set* шире, чем *to_set*, то байты, которым не найдено соответствие, удаляются из *r*.

```
FUNCTION UTL_RAW.TRANSLITERATE
    (r IN RAW,
    to_set IN RAW DEFAULT NULL,
    from_set IN RAW DEFAULT NULL,
    pad IN RAW DEFAULT NULL)
    RETURN RAW;
```

Возвращает содержимое *r*, преобразуя байты, найденные в *from_set*, в находящиеся в той же позиции байты *to_set*. Если набор *from_set* шире, чем *to_set*, то байты, которым не найдено соответствие, преобразуются в *pad*.

```
FUNCTION UTL_RAW.XRANGE
    (start_byte IN RAW DEFAULT NULL,
    end_byte IN RAW DEFAULT NULL)
    RETURN RAW;
```

Возвращает строку типа RAW, содержащую все байты по порядку начиная с *start_byte* и заканчивая *end_byte* (включительно). Если значение *start_byte* превышает *end_byte*, то результат переходит от 0xFF к 0x00.

UTL_REF

Содержит процедуры для выборки и изменения экземпляров объектных типов, хранящихся в объектной таблице. Имя таблицы знать не требуется. Появился в Oracle 8.0.4.

Вызовы

```
PROCEDURE UTL_REF.DELETE_OBJECT
    (reference IN REF ANY);
```

Удаляет объект (точнее, строку, содержащую объект), который определяется ссылкой *reference*.

```
PROCEDURE UTL_REF.LOCK_OBJECT
    (reference IN REF ANY
    [, object IN OUT ANY]#);
```

Блокирует объект, на который ссылается параметр *reference.object* (появилась в Oracle8*i*).

```
PROCEDURE UTL_REF.SELECT_OBJECT
    (reference IN REF ANY,
    object IN OUT ANY);
```

Извлекает объект, идентифицируемый ссылкой *reference*, в *object*.

```
PROCEDURE UTL_REF.UPDATE_OBJECT
    (reference IN REF ANY,
    object IN ANY);
```

Заменяет все объекты БД, идентифицируемые ссылкой *reference*, объектом *object*.

UTL_SMTP

Обеспечивает интерфейс с простым протоколом электронной почты (Simple Mail Transfer Protocol – SMTP) (SMTP), что позволяет отправлять электронную почту из пакета. Появился в Oracle9*i*.

Вызовы

```
FUNCTION UTL_SMTP.OPEN_CONNECTION
    (host IN VARCHAR2,
    port IN PLS_INTEGER DEFAULT 25,
    [c OUT CONNECTION,]
    tx_timeout IN PLS_INTEGER DEFAULT NULL)
    RETURN {REPLY | CONNECTION};
```

Открывает соединение с сервером SMTP через порт *port* хоста *host* с таймаутом *tx_timeout*. Если соединение передается в параметре *c*, то возвращается тип REPLY; в противном случае – тип CONNECTION.

```
FUNCTION UTL_SMTP.COMMAND
    (c IN CONNECTION,
    ord IN VARCHAR2,
    arg IN VARCHAR2 DEFAULT NULL)
    RETURN REPLY;

PROCEDURE UTL_SMTP.COMMAND
    (c IN CONNECTION,
    cmd IN VARCHAR2,
    arg IN VARCHAR2 DEFAULT NULL);

FUNCTION UTL_SMTP.COMMAND_REPLIES
    (c IN CONNECTION,
    ord IN VARCHAR2,
    arg IN VARCHAR2 DEFAULT NULL)
    RETURN REPLIES;
```

Выполняет команду *cmd* с аргументом *arg* для соединения *c*. Если *cmd* формирует ответ, применяется функция. Если *cmd* генерирует несколько ответов, следует применять COMMAND_REPLIES.

```
FUNCTION UTL_SMTP.HELO
   (c IN NOCOPY CONNECTION,
    domain IN NOCOPY)
    RETURN REPLY;

PROCEDURE UTL_SMTP.HELO
    (c IN NOCOPY CONNECTION,
    domain IN NOCOPY);
```

Устанавливает взаимодействие с соединением *c*. Параметр *domain* определяет локальный хост.

```
FUNCTION UTL_SMTP.EHLO
    (c IN NOCOPY CONNECTION,
    domain IN NOCOPY)
    RETURN REPLIES;

PROCEDURE UTL_SMTP.EHLO
    (c IN NOCOPY CONNECTION,
    domain IN NOCOPY);
```

Устанавливает взаимодействие с соединением *c* и возвращает расширенную информацию. Параметр *domain* определяет локальный хост.

```
FUNCTION UTL_SMTP.MAIL
    (c IN NOCOPY CONNECTION,
    sender IN OUT NOCOPY,
    parameters IN OUT NOCOPY)
    RETURN REPLY;

PROCEDURE UTL_SMTP.MAIL
    (c IN NOCOPY CONNECTION,
    sender IN OUT NOCOPY,
    parameters IN OUT NOCOPY);
```

Определяет отправителя *sender* электронной почты для соединения *c* с параметрами *parameters*.

```
FUNCTION UTL_SMTP.RCPT
    (c IN NOCOPY CONNECTION,
    recipient IN OUT NOCOPY,
    parameters IN OUT NOCOPY)
    RETURN REPLY;

PROCEDURE UTL_SMTP.RCPT
    (c IN NOCOPY CONNECTION,
    recipient IN OUT NOCOPY,
    parameters IN OUT NOCOPY);
```

Определяет получателя *recipient* электронной почты для соединения *c* с параметрами *parameters*.

```
FUNCTION UTL_SMTP.DATA
    (c IN NOCOPY CONNECTION,
    body IN OUT NOCOPY)
    RETURN REPLY;

PROCEDURE UTL_SMTP.DATA
    (c IN NOCOPY CONNECTION,
    body IN OUT NOCOPY);
```

Определяет тело *body* сообщения для соединения *c*.

```
FUNCTION UTL_SMTP.OPEN_DATA
    (c IN NOCOPY connection)
    RETURN REPLY;

PROCEDURE UTL_SMTP.OPEN_DATA
    (c IN NOCOPY CONNECTION);
```

Отправляет команду DATA соединению *c*.

```
FUNCTION UTL_SMTP.WRITE_DATA
    (c IN NOCOPY CONNECTION,
    data IN OUT NOCOPY)
    RETURN REPLY;

PROCEDURE UTL_SMTP.WRITE_DATA
    (c IN NOCOPY CONNECTION,
    data IN OUT NOCOPY);
```

Записывает данные *data* в соединение *c* после выполнения команды OPEN_DATA.

```
FUNCTION UTL_SMTP.WRITE_RAW_DATA
    (c IN NOCOPY CONNECTION,
    sender IN OUT NOCOPY);
```

Записывает данные *data* типа RAW в соединение *c* после выполнения команды OPEN_DATA.

```
FUNCTION UTL_SMTP.CLOSE_DATA
    (c IN NOCOPY CONNECTION)
    RETURN REPLY;

PROCEDURE UTL_SMTP.CLOSE_DATA
    (c IN NOCOPY CONNECTION);
```

Закрывает электронное сообщение, отправляя последовательность <CR><LF>. <CR><LF> соединению *c*.

```
FUNCTION UTL_SMTP.RSET
    (c IN NOCOPY CONNECTION)
    RETURN REPLY;

PROCEDURE UTL_SMTP.RSET
    (c IN NOCOPY CONNECTION);
```

Прерывает текущую операцию по отправке почты соединению *c*.

```
FUNCTION UTL_SMTP.VRFY
    (c IN OUT NOCOPY CONNECTION,
    recipient IN OUT NOCOPY)
    RETURN REPLY;
```

Проверяет корректность адреса электронной почты *recipient*.

```
FUNCTION UTL_SMTP.NOOP
    (c IN NOCOPY CONNECTION)
    RETURN VARCHAR2;

PROCEDURE UTL_SMTP.NOOP
    (c IN NOCOPY CONNECTION);
```

Отправляет пустую команду.

```
FUNCTION UTL_SMTP.QUIT
    (c IN OUT NOCOPY CONNECTION)
    RETURN VARCHAR2;
```

Закрывает соединение *c*.

UTL_TCP

Применяется для организации взаимодействия с внешними серверами напрямую при помощи протокола TCP/IP. Появился в Oracle8*i*.

Вызовы

```
FUNCTION UTL_TCP.OPEN_CONNECTION
    (remote_host IN VARCHAR2,
    remote_port IN PLS_INTEGER,
    local_host IN VARCHAR2 DEFAULT NULL,
    local_port IN PLS_INTEGER DEFAULT NULL,
    in_buffer_size IN PLS_INTEGER DEFAULT NULL,
    out_buffer_size IN PLS_INTEGER DEFAULT NULL,
    charset IN VARCHAR2 DEFAULT NULL,
    newline DEFAULT CRLF,
    tx_timeout IN PLS_INTEGER DEFAULT NULL)
    RETURN CONNECTION;
```

Устанавливает соединение с портом *remote_port* удаленного хоста *remote_host*; если же эти параметры содержат NULL, то используется порт *local_port* на локальном хосте *local_host*. Задает размеры буферов ввода и вывода, набор символов для передачи *charset*, последовательность для символа новой строки *newline* и таймаут *tx_timeout* (период времени, в течение которого пакет будет ожидать операцию чтения или записи). Возвращает дескриптор соединения.

```
FUNCTION UTL_TCP.AVAILABLE
    (c IN OUT NOCOPY CONNECTION,
    timeout IN PLS_INTEGER DEFAULT 0)
    RETURN PLS_INTEGER;
```

Определяет для соединения *c* объем данных, доступных для чтения. Ожидает в течение *timeout* секунд.

```
FUNCTION UTL_TCP.READ_RAW
    (c IN OUT NOCOPY CONNECTION,
    data IN OUT NOCOPY RAW,
    len IN PLS_INTEGER DEFAULT 1,
    peek IN BOOLEAN DEFAULT FALSE)
    RETURN PLS_INTEGER;
```

Читает *slen* байт данных *data* типа RAW из *c*. Если параметр *peek* установлен в TRUE, то данные остаются в очереди ввода. Возвращает фактическое количество полученных байтов.

```
FUNCTION UTL_TCP.WRITE_RAW
    (c IN OUT NOCOPY CONNECTION,
    data IN RAW,
    len IN PLS_INTEGER DEFAULT NULL)
    RETURN PLS_INTEGER;
```

Записывает в *c* двоичные данные *data* длиной *len* байт. Если параметр *len* содержит NULL, то отправляются все данные *data*. Возвращает фактическое количество переданных байтов.

```
FUNCTION UTL_TCP.READ_TEXT
    (c IN OUT NOCOPY CONNECTION,
    data IN OUT NOCOPY VARCHAR2,
    len IN PLS_INTEGER DEFAULT 1,
```

```
    peek IN BOOLEAN DEFAULT FALSE)
    RETURN PLS_INTEGER;
```

Читает *slen* байт текстовых данных *data* из *c*. Если параметр *peek* установлен в TRUE, то данные остаются в очереди ввода. Возвращает фактическое количество полученных байтов.

```
FUNCTION UTL_TCP.WRITE_TEXT
    (c IN OUT NOCOPY CONNECTION,
    data IN VARCHAR2,
    len IN PLS_INTEGER DEFAULT NULL)
    RETURN PLS_INTEGER;
```

Записывает в *c* текстовые данные *data* длиной *len* байт. Если параметр *len* содержит NULL, то отправляются все данные *data*. Возвращает фактическое количество переданных байтов.

```
FUNCTION UTL_TCP.READ_LINE
    (c IN OUT NOCOPY CONNECTION,
    data IN OUT NOCOPY VARCHAR2,
    remove_crlf IN PLS_INTEGER DEFAULT 1,
    peek IN BOOLEAN DEFAULT FALSE)
    RETURN PLS_INTEGER;
```

Читает текстовую строку данных *data* из *c*. Если параметр *remove_crlf* установлен в TRUE, то завершающие символы конца строки и перевода каретки удаляются. Если параметр *peek* установлен в TRUE, то данные остаются в очереди ввода. Возвращает фактическое количество полученных байтов.

```
FUNCTION UTL_TCP.WRITE_LINE
    (c IN OUT NOCOPY CONNECTION,
    data IN VARCHAR2)
    RETURN PLS_INTEGER;
```

Записывает текстовую строку данных *data* в *c*. Возвращает фактическое количество переданных байтов.

```
FUNCTION UTL_TCP.GET_RAW
    (c IN OUT NOCOPY CONNECTION,
    len IN PLS_INTEGER DEFAULT 1,
    peek IN BOOLEAN DEFAULT FALSE)
    RETURN RAW;

FUNCTION UTL_TCP.GET_TEXT
    (c IN OUT NOCOPY CONNECTION,
    len IN PLS_INTEGER DEFAULT 1,
    peek IN BOOLEAN DEFAULT FALSE)
    RETURN VARCHAR2;

FUNCTION UTL_TCP.GET_LINE
    (c IN OUT NOCOPY CONNECTION,
    remove_crlf IN BOOLEAN DEFAULT FALSE
    peek IN BOOLEAN DEFAULT FALSE)
    RETURN VARCHAR2;
```

Различные формы функций READ возвращают фактические данные.

```
PROCEDURE UTL_TCP.FLUSH
    (c IN OUT NOCOPY CONNECTION);
```

Передает все данные буфера вывода в *c*.

```
PROCEDURE UTL_TCP.CLOSE_CONNECTION
    (c IN OUT NOCOPY connection);
```

Закрывает соединение *c*.

```
FUNCTION UTL_TCP.CLOSE_ALL_CONNECTIONS;
```

Закрывает все соединения TCP/IP.

UTL_URL

Функция пакета ESCAPE устанавливает экранирование символов URL, а функция UNESCAPE отменяет его.

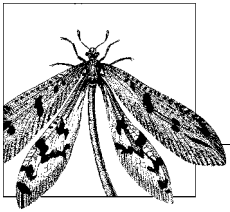
Вызовы

```
FUNCTION UTL_URL.ESCAPE
    (url IN VARCHAR2,
    escape_reserved_chars IN BOOLEAN DEFAULT FALSE,
    url_charset IN VARCHAR2 DEFAULT UTL_HTTP.BODY_CHARSET)
    RETURN VARCHAR2;
```

Если параметр *escape_reserved_chars* установлен в TRUE, экранируются как неразрешенные, так и зарезервированные символы адреса *url*; иначе экранируются только неразрешенные символы. Если задано значение параметра *url_charset*, то перед экранированием символы преобразуются к данной кодировке.

```
FUNCTION UTL_URL.UNESCAPE
    (url IN VARCHAR2,
    url_charset IN VARCHAR2 DEFAULT UTL_HTTP.BODY_CHARSET)
    RETURN VARCHAR2;
```

Отменяет экранирование символов в адресе *url*, возвращая его исходную форму. Если задано значение параметра *url_charset*, то после отмены экранирования символы преобразуются к данной кодировке.



11

Oracle и Java

Рост популярности Java в последние годы оказал большое влияние на развитие компьютерного мира. Кроссплатформенность и средства объектно-ориентированного программирования языка Java обусловили его широкое применение в качестве среды разработки и исполнения. С появлением Oracle8*i* технология Java стала неотъемлемой частью всей среды Oracle.

Существуют два основных способа доступа к данным Oracle из Java:

SQLJ

Язык высокого уровня, предназначенный для встраивания команд SQL в код Java. Применяемые в SQLJ команды SQL проверяются на этапе компиляции. До выхода Oracle9*i* в SQLJ допускалось использование только статических команд SQL, а в Oracle9*i* появилась поддержка и динамических команд SQL.

JDBC

Применение этого Java API обычно требует большего объема кода по сравнению с SQLJ. Проверка команд SQL осуществляется во время исполнения, что приводит к более слабой типизации, чем в SQLJ.

Оба интерфейса признаны стандартными для Java, и каждый из них имеет свои достоинства – меньший объем кодирования в SQLJ, более полный контроль над выполнением в JDBC. В какой-то мере выбор – это дело вкуса, хотя язык Java с его ориентацией на низкоуровневое программирование предназначен не для тех, кого может отпугнуть более детальное кодирование в JDBC. Кроме того, при желании можно чередовать вызовы JDBC с обращениями к SQLJ.

Эта глава посвящена применению Java для работы с базой данных Oracle. В ней рассмотрены следующие вопросы:

- Драйверы Java, работающие с Oracle
- Применение Java в базе данных Oracle
- Соответствие типов данных Java и Oracle
- Интерфейсы SQLJ и JDBC для доступа к Oracle

Общие вопросы использования Java выходят за рамки этой книги. В качестве хорошего справочника по синтаксису и возможностям Java рекомендуем книгу «Java in a Nutshell» Дэвида Флэнагана (David Flanagan), выпущенную издательством O'Reilly.[1]

---

[1] Д. Флэнаган «Java. Справочник», 4-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2004.

Дополнительные сведения об SQLJ и JDBC можно почерпнуть из книг «Java Programming with Oracle SQLJ» Джейсона Прайса (Jason Price) и «Java Programming with Oracle JDBC» Дона Бэйлса (Don Bales) того же издательства. Обратитесь также к документации Oracle по соответствующим продуктам.

Драйверы Java

Oracle поддерживает два типа драйверов JDBC:

- Драйвер JDBC Oracle Call Interface (OCI), или «толстый драйвер» (fat driver), – драйвер типа 2. Позволяет полностью использовать возможности Oracle Net Services.
- «Тонкий драйвер» (thin driver) JDBC – драйвер типа 4, полностью реализованный на Java, не поддерживающий полностью возможности Oracle Net Services.

Драйверы каждого из этих типов существуют в клиентском и серверном исполнениях.

Сведения о совместимости конкретной версии Oracle с драйверами JDBC можно найти в списке совместимости в документации Oracle, содержащем номера версий Oracle и соответствующие номера версий JDK и JDBC.

Java и база данных Oracle

Начиная с версии Oracle8*i* в состав сервера Oracle входит виртуальная машина Java (Java Virtual Machine – JVM). Процедуры Oracle могут быть написаны на Java подобно тому, как они пишутся на PL/SQL.

Последующие разделы описывают шаги, необходимые для загрузки Java-программ в базу данных и их исполнения. Поставляемый Oracle инструмент разработки JDeveloper позволяет создать профиль развертывания, который выполнит нужные действия автоматически.

Компиляция и загрузка Java

Разрабатывая на Java программу, которая должна выполняться в среде БД Oracle, следует соблюдать ряд условий:

- В SQLJ нет необходимости импортировать стандартные библиотеки Java, такие как *oracle.sqlj.runtime.Oracle*. При работе с JDBC используемые библиотеки надо импортировать.
- В SQLJ нет необходимости устанавливать соединение с базой данных, т. к. этим занимается промежуточный драйвер JDBC KPRB. При работе с JDBC соединение должно быть установлено.
- Ни в SQLJ, ни в JDBC нельзя использовать *автофиксацию* (*autocommit*).

Код, написанный на Java, следует загрузить в базу данных Oracle программой *loadjava*, описанной в следующем разделе.

loadjava

Скомпилировав Java-код, воспользуйтесь программой *loadjava*, чтобы загрузить в базу данных классы или JAR-архивы. Программа запускается такой командой:

```
loadjava -user имя_пользователя/пароль[@URL] [список_параметров] список_файлов
```

где *имя_пользователя* и *пароль* идентифицируют схему, в которую будут загружены классы; *URL* – это URL базы данных (о нем более подробно рассказано ниже в описании функции соединения в разделе «Методы SQLJ» этой главы). В *списке_файлов* перечислены загружаемые файлы, а *список_параметров* включает в себя один или несколько из перечисленных ниже параметров:

-action

Выполняет все действия. Принимается по умолчанию. Появился в Oracle9*i*.

-andresolve

Компилирует загруженные исходные файлы и выполняет разрешение имен при загрузке. Несовместим с параметром -resolve.

-casesensitivepub

Сообщает о необходимости учитывать регистр символов при публикации имен. Появился в Oracle9*i*.

-cleargrants

Отзывает все выданные привилегии на классы, исходные тексты и ресурсы. Используется вместе с параметром -grants для переустановки привилегий.

-debug

Генерирует и отображает отладочную информацию в процессе загрузки.

-definer | d

Указывает, что методы загружаемых классов должны исполняться с правами владельца, а не текущего пользователя.

-dirprefix *префикс*

Удаляет префикс из имени объекта схемы перед определением такого имени. Появился в Oracle9*i*.

-encoding | e

Определяет кодировку исходных файлов.

-fileout *файл*

Выводит все сообщения в файл. Появился в Oracle9*i*.

-force | f

Перезагружает ранее загруженные файлы классов Java.

-genmissing

Указывает на необходимость генерации фиктивных определений отсутствующих классов, упомянутых в загружаемых методах. Появился в Oracle9*i*.

-genmissingjar *jar_файл*

Действует как -genmissing и создает JAR-файл, содержащий сгенерированные определения. Появился в Oracle9*i*.

-grant | g

Предоставляет привилегию EXECUTE на класс всем перечисленным пользователям и ролям.

-help

Выводит список параметров команды.

-jararesource

Загружает файл JAR целиком, не распаковывая его. Появился в Oracle9*i*.

-noaction

Не предпринимает никаких действий, кроме проверки наличия элемента META-INF/*loadjava*-options в файле JAR; в случае обнаружения элемента будет обработан соответствующий файл параметров. Появился в Oracle9*i*.

-nocasesensitivepub

Указывает на необходимость преобразования всех символов нижнего регистра к верхнему регистру с добавлением символа подчеркивания. Появился в Oracle9*i*.

-nocleargrants

Отменяет отзыв привилегии EXECUTE, заданный параметром -cleargrants. Появился в Oracle9*i*.

-nodefiner

Предоставляет загружаемым классам права вызывающего их пользователя. Появился в Oracle9*i*.

-nograntt

Отменяет выдачу загружаемым классам привилегии EXECUTE. Применяется по умолчанию. Появился в Oracle9*i*.

-norecursivejars

Считает, что JAR-файлы, вложенные в другие JAR-файлы, представляют собой ресурсы. Применяется по умолчанию. Появился в Oracle9*i*.

-noschema

Помещает загружаемые классы, исходные коды и ресурсы в пользовательскую схему. Применяется по умолчанию. Появился в Oracle9*i*.

-noserverside

Указывает, что доступ к объектам на стороне сервера осуществляется через драйвер JDBC, а не напрямую, как принято по умолчанию. Появился в Oracle9*i*.

-nosynonym

Указывает, что для классов не будут создаваться открытые синонимы. Применяется по умолчанию. Появился в Oracle9*i*.

-nousage

Подавляет вывод сообщения о параметрах при запуске без оных. Появился в Oracle9*i*.

-noverify

Указывает, что загружаемые классы не проверяются верификатором байт-кода. Применяется вместе с параметром -resolve. Появился в Oracle9*i*.

-oci8 | o | oci

Для доступа к БД использует драйвер Oracle Call Interface (OCI).

-optionfile *файл*

Задает файл параметров, добавляемых в конец командной строки. Появился в Oracle9*i*.

-optiontable *таблица*

Действует аналогично -optionfile, но использует SQL-таблицу. Появился в Oracle9*i*.

-oracleresolver

Находит отсутствующие классы, задействованные в загружаемых классах. Применяется по умолчанию. Только для Oracle8*i*.

-publish *пакет*

Создает интерфейсный PL/SQL-пакет для соответствующих методов. Критерии соответствия описаны в документации Oracle. Появился в Oracle9*i*.

-pubmain *число*

Создает несколько вариантов процедуры или функции, каждый из которых принимает различное количество аргументов типа VARCHAR вплоть до значения *число*. Применимо к main и к любому методу с единственным аргументом типа java.lang.String. Появился в Oracle9*i*.

-recursivejars

Обрабатывает вложенные JAR-файлы так же, как и файлы верхнего уровня. Появился в Oracle9*i*.

-resolve | r

Разрешает внешние ссылки в загруженных и скомпилированных (при необходимости) классах. Несовместим с параметром –andresolve.

-R | resolver "*resolver_spec*"

resolver_spec содержит шаблоны, с которыми сверяются имена создаваемых или замещаемых в схеме классов.

-resolveonly

Пропускает начальный этап создания. Появился в Oracle9*i*.

-schema *схема*

Определяет схему БД, в которую будут загружены объекты Java. По умолчанию это текущая схема.

-stdout

Указывает, что вывод осуществляется в stdout, а не в stderr. Появился в Oracle9*i*.

-stoponerror

Прерывает обработку при возникновении ошибки Java. Появился в Oracle9*i*.

-synonym | s

Создает открытый синоним для каждого загруженного класса Java.

-tableschema *схема*

Создает внутренние таблицы *loadjava* в указанной *схеме*. Появился в Oracle9*i*.

-thin

Указывает, что доступ к БД осуществляется через «тонкий драйвер». Несовместим с устанавливаемым по умолчанию параметром –oci8.

-time

Добавляет метку времени в каждое сообщение. Появился в Oracle9*i*.

-unresolvedok

В сочетании с параметром -resolve заставляет игнорировать ошибки при разрешении имен. Появился в Oracle9*i*.

-verbose

Включает детализацию сообщений о процессе загрузки.

После загрузки классов Java в БД их можно просмотреть посредством такой команды SQL:

```
SELECT OBJECT_NAME FROM USER_OBJECTS WHERE OBJECT_TYPE = 'JAVA CLASS';
```

dropjava

Удаление классов Java из базы данных выполняется программой *dropjava*. Синтаксис команды для запуска *dropjava* таков:

```
dropjava -user имя_пользователя/пароль[@URL] [список_параметров] список_файлов
```

Параметры командной строки те же, что и для программы *loadjava*, описанной в предыдущем разделе, но *список_параметров* отличается:

-genmissingjar *jar-файл*

Операнд рассматривается как файл, подлежащий обработке. Появился в Oracle9*i*.

-help

Выводит список доступных параметров программы.

-jararesource

Целиком удаляет файл JAR, ранее загруженный в качестве ресурса. Появился в Oracle9*i*.

-oci8 | o | oci

Использует для доступа к БД драйвер Oracle Call Interface (OCI).

-optionfile *файл*

См. аналогичный параметр в описании программы *loadjava*. Появился в Oracle9*i*.

-optiontable *таблица*

См. аналогичный параметр в описании программы *loadjava*. Появился в Oracle9*i*.

-schema | S *схема*

Указывает, из какой схемы удаляются объекты.

-stdout

См. аналогичный параметр в описании программы *loadjava*. Появился в Oracle9*i*.

-synonym|s

Удаляет открытые синонимы для всех загруженных классов Java. Появился в Oracle9*i*.

-thin | t

Указывает, что доступ к БД осуществляется через «тонкий драйвер». Несовместим с устанавливаемым по умолчанию параметром –oci8.

-time

См. аналогичный параметр в описании программы *loadjava*. Появился в Oracle9*i*.

-verbose

Включает детализацию сообщений о процессе исполнения.

Создание процедур-оберток PL/SQL

После загрузки класса Java в БД необходимо создать процедуру-обертку PL/SQL для его вызова.

На классы, вызываемые из PL/SQL, накладываются два ограничения:

- Методы, публикуемые в SQL и PL/SQL, должны быть объявлены как статические. В PL/SQL отсутствуют механизмы для работы с нестатическими классами Java.

- Класс во время исполнения не должен обращаться к интерфейсу GUI, в частности к AWT (Abstract Windowing Toolkit).

Команда создания такой процедуры имеет следующий вид:

```
CREATE [OR REPLACE] {
   PROCEDURE имя_процедуры [(параметр[, параметр ...])]     |
   FUNCTION  имя_функции [(параметр[, параметр ...])]
   RETURN plsql_тип]}

    [AUTHID {DEFINER | CURRENT_USER}]
    [PARALLEL_ENABLE]
    [DETERMINISTIC]
   {IS | AS} LANGUAGE JAVA
   NAME 'java_метод (java_тип[, java_тип] ...)
   [RETURN java_тип]';
```

где *параметр* определяется как:

```
параметр := имя_параметра [IN | OUT | IN OUT] plsql_тип
```

где:

имя_процедуры/имя_функции

Указывает имя создаваемой процедуры-обертки PL/SQL или функции-обертки.

plsql_тип

Определяет тип PL/SQL для значения, возвращаемого функцией-оберткой.

AUTHID {DEFINER|CURRENT_USER}

Определяет, выполняется ли хранимая подпрограмма с привилегиями создавшего ее владельца (DEFINER) или с привилегиями текущего пользователя (CURRENT_USER). По умолчанию это CURRENT_USER.

PARALLEL_ENABLE

Указывает, что функция может применяться в дочерних сеансах при параллельном выполнении DML. Параллельный DML позволяет распределить DML-операции между несколькими одновременно выполняющимися процессами для увеличения производительности.

DETERMINISTIC

Дает указание оптимизатору избегать избыточных вызовов функции. При этом предполагается, что она детерминирована, т. е. возвращаемый результат зависит только от значений входных параметров. В этом случае, если функция уже вызывалась с тем же набором параметров, будет взят результат, полученный ранее.

LANGUAGE JAVA инструкция

Указывает, что процедура или функция вызывает метод Java.

NAME инструкция

Определяет имя упаковываемого метода Java, а также его параметры. В *инструкции* NAME можно указать:

java_метод

Упаковываемый метод Java должен быть однозначно определен. Для указания класса и его метода следует применять точечную нотацию.

java_тип

Определяет типы параметров метода Java, причем эти типы должны быть совместимы с соответствующими *plsql_типами*.

имя_параметра

Определяет имя параметра PL/SQL.

Созданную процедуру-обертку можно вызывать так же, как и любую другую PL/SQL-процедуру.

Соответствие типов данных

Между типами данных Oracle и типами данных Java нет полного соответствия. В табл. 11.1 представлено соответствие между типами данных Java и Oracle.

Некоторые типы данных java.lang представляют собой классы-обертки для примитивных типов Java, способные принимать значения NULL, т. к. обычные классы Java этого не могут. Для получения значения из класса-обертки применяется соответствующий метод *typeValue*().

Таблица 11.1. Соответствие типов данных Java и Oracle

Тип данных Java	Тип данных Oracle
java.sql.Array	ARRAY
java.sql.Blob	BLOB
Boolean java.lang.Boolean	NUMBER
Byte java.lang.byte	NUMBER
byte[]	RAW LONGRAW
java.sql.Clob	CLOB
java.sql.Date	DATE
Double java.lang.double	NUMBER
Float java.lang.float	NUMBER
Int java.lang.Integer	NUMBER
Long java.lang.Long	NUMBER
java.sql.Ref	REF
Short java.lang.Short	NUMBER
java.lang.string	VARCHAR2 LONGRAW
java.sql.Struct	STRUCT
java.sql.time	DATE
java.sql.timestamp	DATE
java.math.BigDecimal (рекомендуется)	NUMBER

Oracle также предлагает ряд расширений для своих типов данных. Расширение oracle.sql.NUMBER позволяет сохранять и извлекать данные Oracle типа NUMBER без потери точности.

Oracle.sql.ARRAY	Oracle.sql.NUMBER
oracle.sql.BFILE	Oracle.sql.RAW
oracle.sql.BLOB	oracle.sql.REF
oracle.sql.CHAR	oracle.sql.ROWID
oracle.sql.CLOB	oracle.sql.STRUCT
oracle.sql.DATE	

SQLJ

SQLJ состоит из двух основных компонентов:

- Библиотеки времени исполнения, осуществляющей фактические обращения к базе данных при выполнении программы
- Компилятора, транслирующего код SQLJ в вызовы библиотеки времени исполнения

Чтобы выполнить код, написанный на SQLJ, на компьютере помимо транслятора должны быть установлены драйверы Oracle JDBC и комплект разработчика Java (Java Development Kit – JDK).

Импорт

Для того чтобы программы, написанные на SQLJ, могли выполняться за пределами БД Oracle, необходимо в коде Java импортировать следующие библиотеки:

- *java.sql.date*
- *java.sql.SQLException*
- *oracle.sqlj.runtime.Oracle*

Библиотека *oracle.sqlj.runtime.Oracle* содержит большинство вызовов, которые могут понадобиться для обращения к БД Oracle. Мы рассмотрим эти вызовы в последующих разделах.

Компиляция кода SQLJ

Код на Java, написанный с применением SQLJ, должен быть откомпилирован программой *sqlj*. Программа вызывается такой командой:

```
sqlj [список_параметров] список_файлов
```

В списке параметров допустимы следующие флаги:

-cache [=True | False]

Логическое значение, указывающее, надо ли кэшировать результаты оперативного семантического контроля. По умолчанию равен False.

-checkfilename [=True | False]

Логическое значение, указывающее, надо ли выводить предупреждения, если имя файла не совпадает с именем открытого класса. По умолчанию равен True.

-classpath

Определяет CLASSPATH для Java.

-codegen=oracle

Подавляет создание файлов профиля. Появился в Oracle9*i*.

-compile [=True | False]

Логическое значение, указывающее, следует ли компилировать сформированный файл *.java*. По умолчанию равен True.

-compiler-executable

Исполняемый файл компилятора Java.

-compiler-output-file

Файл вывода для компилятора. Если параметр не указан, вывод отображается на экране.

-C*параметр*

Параметр, передаваемый компилятору Java.

-d

Каталог для сформированных файлов *.ser* и *.class*.

-default-customizer

Имя класса для *модуля настройки профиля* (*profile customizer*). По умолчанию равен oracle.sqlj.runtime.util.OraCustomizer.

-dir

Каталог для сформированных файлов *.java*.

-driver

Имя класса для используемого драйвера JDBC. По умолчанию равен oracle.jdbc.driver.OracleDriver.

-explain [=True | False]

Логическое значение, указывающее, следует ли включать вывод подробных сообщений об ошибках. По умолчанию равен False.

-g

Заставляет компилятор генерировать отладочный код.

-help

Выводит список доступных параметров.

-J*параметр*

Параметр, передаваемый виртуальной машине Java.

-linemap [=True | False]

Указывает на необходимость хранения информации о соответствии номеров строк SQLJ и сформированного кода Java. По умолчанию равен False.

-n

Предписывает транслятору SQLJ вывести командную строку, не выполняя ее.

-nowarn

Отменяет параметр -warn компилятора Java.

-O

Отменяет параметр -linemap компилятора Java.

-online

Имя класса программы семантического контроля SQL. Вместе с данным параметром необходимо задавать параметры -user и -password. По умолчанию равен oracle.sqlj.checker.OracleChecker.

-P*параметр*

Параметр, передаваемый модулю настройки профиля SQLJ.

-password

Указывает пароль, передаваемый базе данных при семантической проверке.

-profile [=True | False]

Указывает, следует ли разрешать настройку профиля. По умолчанию равен True.

-props

Задает файл, содержащий параметры, которые будут добавлены в командную строку.

-ser2class [=True | False]

Указывает, следует ли преобразовывать сформированные профили *.ser* в файлы *.class*. По умолчанию – False.

-status

Указывает, следует ли разрешить отображение статусных сообщений. По умолчанию – False.

-url

Указывает URL базы данных для оперативной проверки семантики.

-user

Пользователь БД, указываемый при подключении к БД при проверке семантики.

-verbose

Передается компилятору Java и включает параметр -status.

-version

Выводит версию SQLJ.

-version-long

Выводит подробную информацию о версии.

-warn

Выводит разделенный запятыми список флагов, управляющих выводом предупреждений SQLJ. Возможны следующие флаги: precision/noprecision, nulls/nonulls, portable/noportable, strict/nostrict, verbose/noverbose и all/none. Последняя пара разрешает или запрещает вывод любых предупреждений. Значениями по умолчанию являются precision, nulls, noportable, strict и noverbose.

Более полная информация о параметрах, которые могут задаваться в команде sqlj, приведена в документации Oracle, а также в руководстве по программированию на Java для Oracle SQL.

Применение SQL в SQLJ

Установив соединение с БД Oracle при помощи функции Oracle.connect() (она будет описана немного позже), можно выполнять команды SQL. Синтаксис команд таков:

```
#sql{ [контекст_соединения] [[,]контекст_исполнения] SQL_команда };
```

Переменная *SQL_команда* может быть практически любой командой SQL, допустимой для БД Oracle. Необязательная переменная *контекст_соединения* задает контекст конкретного соединения, а необязательная переменная *контекст_исполнения* – контекст исполнения для данного соединения (см. подробную информацию в описаниях функций *getConnection( )* и *getExecutionContext( )* далее в разделе «Методы SQL»).

Уровень изоляции для транзакции SQL можно указать следующим образом:

```
#sql [контекст_соединения] {SET TRANSACTION (READ ONLY | READ WRITE)
    ISOLATION LEVEL (SERIALIZATION | READ COMMITTED) }
```

Если комада SQL возвращает значение, то его можно получить посредством такой конструкции:

```
#sql{ :переменная_базового_языка = SQL_команда };
```

Переменная *переменная_базового_языка* может содержать указатель режима: IN, OUT или же IN OUT. Если *переменная_базового_языка* входит в список INTO или в присваивание с участием команды SET (обо всем этом мы поговорим чуть позже в этом же разделе), то по умолчанию устанавливается режим OUT; в противном случае по умолчанию устанавливается режим IN.

Можно также применять следующую команду:

```
#sql { SET :выражение_базового_языка = выражение };
```

для получения данных в *переменную_базового_языка*. Если выражение представляет собой функцию PL/SQL, то можно применить инструкцию VALUES, которая обсуждается в разделе «PL/SQL в SQLJ».

В версии Oracle9*i* можно работать с переменными связывания для имен столбцов и таблиц с целью создания динамических команд SQL. Если имя переменной связывания совпадает с именем столбца или таблицы, то поставьте перед ней двоеточие (:). Если же вы планируете использовать переменную связывания для замены столбца или таблицы с другим именем, необходимо применить такую конструкцию:

```
: переменная_базового_языка :: заменяемое_имя_БД
```

Нельзя применять переменную связывания для имени столбца или таблицы в инструкции INTO, а также указывать значение для команд CALL, VALUES, SET, FETCH и CAST.

Если команда SQL возвращает единственную строку, то для извлечения данных можно применить такую команду:

```
SELECT ... INTO : переменная_базового_языка ...
```

Если команда SQL возвращает несколько строк, то для получения данных придется применить итератор SQL. Итераторам посвящен следующий раздел.

Итераторы SQL

Итератор SQL предназначен для обработки нескольких строк, возвращенных из команды SELECT. Для того чтобы использовать итератор SQL, необходимо объявить класс итератора, объявить объект итератора, затем заполнить его возвращаемыми командой SELECT данными. После того как итератор заполнен, из него можно читать строки. Если вы имеете дело с вложенной командой SELECT, то для получения данных необходимо применять вложенные итераторы, по одному для каждой команды SELECT.

Можно создать итератор и применять его совместно с результирующим множеством JDBC, присвоив результат функции getResultSet() результирующему множеству JDBC.

Существуют два вида итераторов: *именованные* (*named*) и *позиционные* (*positional*).

Именованные итераторы

Для объявления класса именованного итератора используйте такой синтаксис:

```
#sql [модификаторы] iterator имя_класса
    [implements класс_интерфейса [, класс_интерфейса . . .]]
    [with имя_константы = значение [,имя_константы = значение . . .]]
    (java_тип имя_столбца [, java_тип имя_столбца ...]);
```

где:

модификаторы

Необязательный модификатор класса Java: public, private, protected, static.

имя_класса

Имя класса итератора.

класс_интерфейса

Интерфейс, реализуемый классом итератора. Если итератор реализует класс sqlj.runtime.Scrollable (доступен начиная с версии Oracle 8.1.7), то итератор будет поддерживать следующие функции перемещения:

- previous()
- first()
- last()
- absolute(*номер_строки*)
- relative(*относительный_номер*) – количество строк относительно текущей строки
- afterLast()
- beforeFirst()

а также другие функции:

- setFetchDirection(*направление*), указывающая направление перемещения по итератору, где направление может быть константой класса sqlj.runtime.ResultSetIterator: FETCH_FORWARD, FETCH_REVERSE или FETCH_UNKNOWN
- getFetchDirection()
- isFirst()
- isLast()
- isBeforeFirst()
- isAfterLast()

имя_константы

Имя константы, которая доступна в итераторе.

java_тип

Java-тип данных столбца итератора.

имя_столбца

Имя столбца в итераторе. Если оно отлично от имени столбца, то для идентификации столбца итератора необходимо применить AS.

Объявив именованный итератор, можно осуществлять выборки в него, применяя такой синтаксис:

```
#sql имя_итератора = {SQL_команда};
```

После того как итератор заполнен данными, вы можете перемещаться по его строкам, используя метод next. Если строк больше не осталось, этот метод возвращает FALSE.

Для каждого столбца именованного итератора существует собственный метод доступа с тем же именем, что и столбец.

Позиционные итераторы

Позиционный итератор во многом похож на именованный, но имеются два важных отличия:

- В объявлении позиционного итератора не указывается *имя_столбца*. Столбцы, извлеченные командой SELECT, помещают свои значения на те же позиции в столбцы итератора.
- Для извлечения строки из позиционного итератора в последовательность переменных базового языка применяется такой синтаксис.:

```
#sql {
  FETCH :имя_итератора INTO :переменная_базового языка [, :переменная_базового_языка ...]
};
```

Для позиционного итератора, который реализует интерфейс sqlj.runtime.Scrollable, инструкция FETCH может принимать следующие формы:

FETCH PREVIOUS | PRIOR

FETCH FIRST

FETCH LAST

FETCH ABSOLUTE:(*row_number*)

FETCH RELATIVE:(*relative_number*)

Метод end_fetch для позиционного итератора возвращает логическое значение, указывающее, была ли извлечена последняя строка.

В версии Oracle9*i* пользователи получили возможность неявно объявлять позиционный итератор, просто делая его переменной базового языка, которая будет получать результаты команды SELECT. Извлекать данные из итератора можно при помощи вызова FETCH CURRENT.

PL/SQL в SQLJ

Для вызова процедуры PL/SQL из SQLJ применяется следующий синтаксис:

```
#sql { CALL имя_процедуры([список_параметров])};
```

А так вызывается функция PL/SQL из SQLJ:

```
#sql переменная базового языка = { VALUES (имя_функции([список_параметров])) }
```

Если функция PL/SQL возвращает REF CURSOR, то *переменная_базового_языка* должна быть итератором, который соответствует резальтирующему множеству.

В SQLJ для получения данных в переменные базового языка можно также применять обсуждавшуюся ранее команду SET. Кроме того, можно включать анонимные блоки кода PL/SQL в вызов SQLJ.

Объекты базы данных и JPublish

Объекты Java можно применять как для записи данных в объекты БД, так и для извлечения данных из объекта БД при помощи инструкции INTO или при помощи итератора, имеющего такую же структуру, что и объект БД.

Утилита JPublish предназначена для создания объектных классов Java с теми же атрибутами и методами, что и объектные классы в БД Oracle. Утилиту можно запускать из командной строки или из JDeveloper (средства разработки на Java для Oracle). За более подробной информацией обратитесь к документации Oracle.

Методы SQLJ

SQLJ реализуется посредством библиотеки oracle.*sqlj.runtime.Oracle*, которую необходимо включать в любой код Java, использующий SQLJ. Имейте в виду, что некоторые из описанных далее методов соответствуют подклассам данного основного класса.

---

connect

```
Oracle.connect(URL_бд,
    имя_пользователя,
    пароль
    [, автофиксация]);
```

Устанавливает соединение с БД Oracle.

Переменные

URL_бд

Имеет вид: *имя_драйвера*@*местоположение_бд*, где *имя_драйвера* – это jdbc:oracle:*драйвер*, а *драйвер* может иметь следующие значения:

- thin для тонкого драйвера JDBC
- oci (Oracle9*i*), oci8 (Oracle8*i*) или oci7 (Oracle7) для драйвера OCI JDBC

Местоположение_бд задается как *имя_хоста*:*порт*:*бд_SID* для тонкого драйвера JDBC или как:

```
(description=(address=(host=имя хоста)(protocol=tcp)(port=порт))
(connect_data=(sid=бд_SID)))
```

для любого JDBC-драйвера. Весь адрес *URL_бд* целиком заключается в двойные кавычки.

имя_пользователя

Имя пользователя, заключенное в двойные кавычки.

пароль

Пароль пользователя, заключенный в двойные кавычки.

автофиксация

Необязательное логическое значение; указывает, следует ли выполнять автофиксацию для команд SQL в рамках данного соединения.

close

```
Oracle.close([ CLOSE_CONNECTION | KEEP_CONNECTION]);
```

Разрывает соединение с БД Oracle.

Переменные

CLOSE_CONNECTION

Закрывает базовое соединение JDBC.

KEEP_CONNECTION

Оставляет базовое соединение JDBC открытым.

Данные вызовы закрывают активное соединение (по умолчанию). Закрыть какое-то определенное соединение позволяет такой вызов:

```
контекст_соединения.close()
```

getConnection

```
DefaultContext контекст_соединения = Oracle.getConnection( URL_бд,
    имя_пользователя,
    пароль
    [, автофиксация]);
```

Создает и возвращает контекст соединения. Если планируется работа с несколькими соединениями, используйте данную функцию для создания нескольких контекстов и переменные базового языка типа DefaultContext для их хранения.

Функция также позволяет передавать базовое соединение в объект соединения JDBC, который затем можно будет использовать для команд JDBC. Можно передать JDBC-соединение соединению SQLJ, создав объект соединения SQLJ, который будет получать значение от вызова JDBC getConnection().

Переменные

Описания переменных приведены в разделе для вызова context().

setDefaultRowPrefetch

```
((OracleConnection) контекст_соединения.getConnection()).setDefaultRowPrefetch(целое);
```

Указывает количество строк, которое будет извлекаться при каждом обращении к серверу.

Переменные

(OracleConnection)

Приводит соединение JDBC к классу *OracleConnection*, который содержит метод setDefaultRowPrefetch().

Если данная функция применяется в виде DefaultContext.setDefaultContext.getConnection(), то можно задать количество строк упреждающей выборки для соединения по умолчанию.

setFetchSize

Заменяет метод setDefaultRowPrefetch() в версии Oracle 8.1.7 и выше.

getDefaultRowPrefetch

```
((OracleConnection) контекст_соединения.getConnection()).getDefaultRowPrefetch();
```

Получает количество строк, выбираемых при каждом обращении к серверу.

Переменные

(OracleConnection)

Приводит соединение JDBC к классу *OracleConnection*, который содержит метод getDefaultRowPrefetch().

Если данная функция применяется в виде DefaultContext.getDefaultContext.getConnection(), то можно получить количество строк упреждающей выборки для соединения по умолчанию.

getFetchSize

Заменяет getDefaultRowPrefetch() в версии Oracle 8.1.7 и выше.

setDefaultContext

```
DefaultContext.setDefaultContext(контекст_соединения);
```

Может применяться для присвоения контекста по умолчанию всем последующим командам SQL.

Переменные

контекст_соединения

Указатель контекста, созданный при помощи getConnection().

getExecutionContext

```
ExecutionContext контекст_исполнения =
    DefaultContext.getDefaultContext.getExecutionContext();
```

Возвращает контекст исполнения для соединения экземпляру ExecutionContext. Контекст исполнения должен был получен прежде, чем будут использоваться многие из описанных далее методов, такие как getMaxRows() и setMaxRows().

Переменные

контекст_исполнения

Экземпляр ExecutionContext для хранения указателя контекста.

getMaxRows

```
контекст_исполнения.getMaxRows();
```

Возвращает максимальное количество строк, которое может быть возвращено итератору.

Переменные

контекст_исполнения

Экземпляр ExecutionContext для хранения указателя контекста.

---

setMaxRows

```
контекст_исполнения.setMaxRows(integer);
```

Устанавливает максимальное количество строк, которое может быть возвращено итератору.

Переменные

контекст_исполнения

Экземпляр ExecutionContext для хранения указателя контекста.

---

getQueryTimeout

```
контекст_исполнения.getQueryTimeout();
```

Возвращает тайм-аут для запроса для контекста исполнения.

Переменные

контекст_исполнения

Экземпляр ExecutionContext для хранения указателя контекста.

---

setQueryTimeout

```
контекст_исполнения.setQueryTimeout(integer);
```

Устанавливает таум-аут для запроса для контекста исполнения.

Переменные

контекст_исполнения

Экземпляр ExecutionContext для хранения указателя контекста.

---

getResultSet

```
ResultSet результирующее_множество_JDBC = итератор.getResultSet()
```

Результирующее множество JDBC получает данные итератора.

Можно передать итератору результирующее множество JDBC посредством такой конструкции:

```
#sql итератор = {CAST результирующее_множество};
```

---

getSQLState

```
sql_warning.getSQLState();
```

Возвращает строку с текстом предупреждения для экземпляра класса SQLWarning, извлеченного методом getWarnings.

getUpdateCount

```
контекст_исполнения.getUpdateCount();
```

Возвращает количество строк, измененных последней командой SQL, для контекста исполнения.

Переменные

контекст_исполнения

Экземпляр ExecutionContext для хранения указателя контекста.

getWarnings

```
SQLWarning sql_warning=контекст_исполнения.getWarnings()
```

Возвращает первое предупреждение из контекста исполнения в экземпляр SQLWarning.

Переменные

контекст_исполнения

Экземпляр ExecutionContext для хранения указателя контекста.

setBatching

```
контекст_исполнения.setBatching(TRUE | FALSE);
```

Включает/выключает пакетную обработку для контекста исполнения.

isBatching

```
Boolean результат контекст_исполнения.isBatching();
```

Возвращает TRUE, если для контекста исполнения включена пакетная обработка, иначе – FALSE.

executeBatch

```
int[] результат контекст_исполнения.executeBatch();
```

Инициирует выполнение пакета команд, даже если количество накопленных в нем команд меньше установленного по умолчанию.

Переменные

результат

Массив целых чисел, каждое из которых указывает на результат выполнения отдельной команды пакета. Если число больше или равно 0, значит, команда выполнена успешно и число указывает количество затронутых выполнением команды записей. Если возвращено значение –2, значит, команда выполнена успешно,

но количество затронутых командой записей неизвестно. Остальные значения указывают на ошибку во время выполнения команды.

setBatchLimits

```
контекст_исполнения.setBatchLimits(целое | AUTO_BATCH | UNLIMITED);
```

Устанавливает количество накапливаемых операций SQL, по достижении которого пакет отправляется на обработку.

Ключевые слова

AUTO_BATCH

Позволяет SQLJ определить наиболее подходящее значение количества операций для пакетного выполнения.

JDBC

JDBC – это стандартный низкоуровневый Java API для баз данных. Этот раздел будет посвящен реализации стандартного набора интерфейсов JDBC в Oracle.

Oracle реализует стандартный интерфейс JDBC при помощи ряда пакетов, а именно (для версии Oracle9*i*):

oracle.jdbc

Состоит из интерфейсов, расширяющих интерфейсы java.sql.Connection, java.sql.MetaData, java.sql.PreparedStatement и java.sql.Savepoint, а также классов для их реализации. Пакет появился в Oracle9*i*. В версии Oracle8*i* некоторые из включенных в него интерфейсов были реализованы как классы в различных других пакетах (см. табл. 11.2). В последующих разделах будут рассмотрены интерфейсы и классы как для версии Oracle8*i*, так и для Oracle9*i*.

oracle.jdbc.driver

В Oracle9*i* отвечает за реализацию интерфейсов библиотеки oracle.jdbc. Классы пакета лишь реализуют интерфейсы, поэтому мы не будем подробно останавливаться на нем в этой главе. В версии Oracle8*i* в библиотеке *oracle.jdbc* интерфейсы не были определены.

oracle.jdbc.pool

Применяется для управления пулом подключений Oracle. Данная возможность востребована лишь небольшим количеством разработчиков, поэтому пакет не будет рассматриваться в этой главе.

*oracle.jdbc.xa**

Появившиеся в Oracle9*i* пакеты oracle.jdbc.xa и oracle.jdbc.xa.client интересны лишь разработчикам, применяющим протокол распределенных транзакций XA, и не будут рассматриваться в этой главе.

oracle.sql

Состоит из интерфейсов и классов для работы с пользовательскими типами данных и типами данных Oracle SQL.

В последующих разделах будут описаны специфические интерфейсы и классы Oracle двух наиболее широко распространенных библиотек Oracle JDBC: oracle.jdbc и oracle.sql. Информацию о стандартных интерфейсах JDBC можно найти в документации по JDBC.

Как уже говорилось, реализация интерфейса JDBC изменилась в версии Oracle9*i*. Далее мы будем говорить о ней как об основной, отмечая при необходимости особенности реализации для Oracle8*i*.

В табл. 11.2 приводится соответствие классов Oracle8*i* их аналогам в версии Oracle9*i* для тех случаев, когда существуют различия в их реализации или иерархии. Если класс или интерфейс не приводится в списке, это означает, что в обеих версиях он имеет одно и то же имя и занимает то же место в иерархической структуре. Названия интерфейсов выделены курсивом.

Таблица 11.2. Классы/интерфейсы Oracle8i и Oracle9i

Класс/интерфейс Oracle8*i*	**Класс/интерфейс Oracle9*i***
CharacterSet [oracle.sql]	Не реализован
ClientDataSupport [oracle.jdbc.driver]	Не реализован
Const [oracle.jdbc.driver]	Не реализован
Mutable [oracle.sql]	Не реализован
OracleCallableStatement [oracle.jdbc.driver]	*OracleCallableStatement* [oracle.jdbc]
OracleConnection [oracle.jdbc.driver]	*OracleConnection* [oracle.jdbc]
OracleDriver [oracle.jdbc.driver]	OracleDriver [oracle.jdbc]
OracleResultSet [oracle.jdbc.internal]	*OracleResultSet* [oracle.jdbc]
OracleResultSetCache [oracle.jdbc.internal]	*OracleResultSetCache* [oracle.jdbc]
OracleResultSetMetaData [oracle.jdbc.internal]	*OracleResultSetMetaData* [oracle.jdbc]
OracleTypes [oracle.jdbc.driver]	OracleTypes [oracle.jdbc]
StructMetaData [oracle.jdbc.driver]	*StructMetaData* [oracle.sql]
Не реализован	INTERVALYM[oracle.sql]
Не реализован	OracleOCIConnectionPool [oracle.jdbc.pool]
Не реализован	OracleXAConnectionCache [oracle.jdbc.pool]

Имейте в виду, что пакет oracle.jdbc.xa.client появился только в Oracle9*i*, так что для него не существует аналога в Oracle8*i*.

oracle.jdbc

Пакет oracle.jdbc включает в себя 10 специальных интерфейсов Oracle и 4 специальных класса Oracle. Интерфейсы содержат основные расширения стандартного синтаксиса JDBC, специфичные для Oracle. Классы реализуют java.sql.driver, java.lang.object и java.sql.DatabaseMetaData со специфическими расширениями Oracle. Существует класс для реализации интерфейса соединения Oracle.

Интерфейсы

Описанные здесь интерфейсы входят в состав пакета oracle.jdbc.

OracleCallableStatement

Служит расширением интерфейсов oracle.jdbc.OraclePreparedStatement и java.sql.CallableStatement в Oracle9*i*. Служит расширением oracle.jdbc.driver.OraclePreparedStatement и реализует интерфейс oracle.jdbc.internal.OracleCallableStatement в Oracle8*i*.

Унаследованные поля

Поля, унаследованные от интерфейса oracle.jdbc.PreparedStatement в Oracle9*i*:

FORM CHAR
FORM NCHAR

Поля, унаследованные от интерфейса oracle.jdbc.Statement в Oracle9*i*:

EXPLICIT
IMPLICIT
NEW

Поля, унаследованные от интерфейса oracle.jdbc.driver.OracleStatement в Oracle8*i*:

auto_rollback
closed
dbstmt
defines
EXPLICIT
IMPLICIT
NEW
wait_option

Методы

addBatch( )

Добавляет набор параметров для пакетной операции. Только для Oracle8*i*.

clearParameters()

Только для Oracle8*i*.

close()

Только для Oracle8*i*.

getAnyDataEmbeddedObject(int *parameterIndex*)

Извлекает данные встроенного объекта AnyData в объект java.lang.Object. Появился в Oracle9*i*.

getARRAY(int *parameterIndex*)

Извлекает данные в объект oracle.sql.ARRAY.

getAsciiStream(int parameterIndex)

Извлекает данные в объект java.io.InputStream.

getBFILE(int *parameterIndex*)

Извлекает данные в объект oracle.sql.BFILE.

getBigDecimal(int *parameterIndex*)

Извлекает данные в java.math.BigDecimal. Только для Oracle8*i*.

getBigDecimal(int *parameterIndex*, int *scale*)

Извлекает данные в java.math.BigDecimal. Только для Oracle8*i*.

getBinaryStream(int *parameterIndex*)

Извлекает данные в объект java.io.InputStream.

getBlob(int *parameterIndex*)

Извлекает данные в oracle.jdbc2.Blob. Только для Oracle8*i*.

getBLOB(int *parameterIndex*)

Извлекает данные в объект oracle.sql.BLOB.

getByte(int *parameterIndex*)

Извлекает данные в байт. Только для Oracle8*i*.

getBytes(int *parameterIndex*)

Извлекает данные в байтовый массив. Только для Oracle8*i*.

getCHAR(int *parameterIndex*)

Извлекает данные в объект oracle.sql.CHAR.

getCharacterStream(int *parameterIndex*)

Извлекает данные в объект java.io.Reader. Появился в Oracle9*i*.

getClob(int *parameterIndex*)

Извлекает данные в объект oracle.jdbc2.Clob. Только для Oracle8*i*.

getCLOB(int *parameterIndex*)

Извлекает данные в объект oracle.sql.CLOB.

getCursor(int *parameterIndex*)

Извлекает данные в объект java.sql.ResultSet.

getCustomDatum(int *parameterIndex*, CustomDatumFactory *factory*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

getDate(int *parameterIndex*)

Извлекает данные в объект java.sql.Date. Только для Oracle8*i*.

getDATE(int *parameterIndex*)

Извлекает данные в объект oracle.sql.DATE.

getDouble(int *parameterIndex*)

Извлекает данные в объект oracle.sql.Double. Только для Oracle8*i*.

getFloat(int *parameterIndex*)

Извлекает данные в объект oracle.sql.Float. Только для Oracle8*i*.

getInt(int *parameterIndex*)

Извлекает данные в объект oracle.sql.Int. Только для Oracle8*i*.

getINTERVALYM(int *parameterIndex*)

Извлекает данные в объект oracle.sql.INTERVALYM. Появился в Oracle9*i*.

getLong(int *parameterIndex*)

Извлекает данные в объект oracle.sql.long. Только для Oracle8*i*.

getNUMBER(int *parameterIndex*)

Извлекает данные в объект oracle.sql.NUMBER.

getObject(int *parameterIndex*)

Извлекает данные в java.lang.object. Только для Oracle8*i*.

getObject(int *parameterIndex*, *java.util.Dictionary map*)

Извлекает данные в java.lang.object. Только для Oracle8*i*.

getOPAQUE(int *parameterIndex*)

Извлекает данные в объект oracle.sql.OPAQUE. Появился в Oracle9*i*.

getOracleObject(int *parameterIndex*)

Извлекает данные в объект oracle.sql.Datum.

getOraclePlsqlIndexTable(int *paramIndex*)

Извлекает индексную таблицу PL/SQL в oracle.sql.Datum. Только для OCI драйвера Oracle.

getORAData(int *parameterIndex*, ORADataFactory *factory*)

Извлекает данные в объект java.lang.Object. Появился в Oracle9*i*.

getPlsqlIndexTable(int *paramIndex*)

Извлекает индексную таблицу PL/SQL в объект java.lang.Object. Только для OCI драйвера Oracle.

getPlsqlIndexTable(int *paramIndex*, java.lang.Class *primitiveType*)

Извлекает индексную таблицу PL/SQL в объект java.lang.Object. Только для OCI драйвера Oracle.

getRAW(int *parameterIndex*)

Извлекает данные в объект oracle.sql.RAW.

getRef(int *parameterIndex*)

Извлекает данные в объект oracle.jdbc2.Ref. Только для Oracle8*i*.

getREF(int *parameterIndex*)

Извлекает данные в oracle.sql.REF.

getROWID(int *parameterIndex*)

Извлекает данные в объект oracle.sql.ROWID.

getShort(int *parameterIndex*)

Извлекает данные в объект oracle.sql.short. Только для Oracle8*i*.

getString(int *parameterIndex*)

Извлекает данные в объект java.lang.string. Только для Oracle8*i*.

getSTRUCT(int *parameterIndex*)

Извлекает данные в объект oracle.sql.STRUCT.

getTime(int *paramIdx*)

Извлекает данные в объект java.sql.Time. Только для Oracle8*i*.

getTime(int *paramIdx*, java.util.Calendar *cal*)

Извлекает данные в объект java.sql.Time. Только для Oracle8*i*.

getTimestamp(int *paramIdx*)

Извлекает данные в объект java.sql.Timestamp. Только для Oracle8*i*.

getTimestamp(int *paramIdx*, java.util.Calendar *cal*)

Извлекает данные в объект java.sql.Timestamp. Только для Oracle8*i*.

getTIMESTAMP(int *paramIdx*)

Извлекает данные в объект oracle.sql.TIMESTAMP. Появился в Oracle9*i*.

getTIMESTAMPLTZ(int *paramIdx*)

Извлекает данные в объект oracle.sql.TIMESTAMPLTZ. Появился в Oracle9*i*.

getTIMESTAMPTZ(int *paramIdx*)

Извлекает данные в объект oracle.sql.TIMESTAMPTZ. Появился в Oracle9*i*.

getUnicodeStream(int *parameterIndex*)

Извлекает данные в объект java.io.InputStream.

registerIndexTableOutParameter(int *paramIndex*, int *maxLen*, int *elemSqlType*, int *elemMaxLen*)

Только для OCI драйвера Oracle.

registerOutParameter(int *paramIndex*, int *sqlType*)

Регистрирует выходные параметры. Только для Oracle8*i*.

registerOutParameter(int *paramIndex*, int *sqlType*, int *scale*)

Регистрирует выходные параметры. Только для Oracle8*i*.

registerOutParameter(int *paramIndex*, int *sqlType*, java.lang.String *sqlName*)

Регистрирует объекты столбцов как выходные параметры. Только для Oracle8*i*.

registerOutParameter(int *paramIndex*, int *sqlType*, int *scale*, int *maxLength*)

Специальная Oracle-версия метода registerOutParameter для регистрации столбцов типов CHAR, VARCHAR, LONG, RAW и LONG RAW.

registerOutParameterBytes(int *paramIndex*, int *sqlType*, int *scale*, int *maxLength*)

Специальная Oracle-версия метода registerOutParameter для регистрации столбцов типов CHAR, VARCHAR, LONG, RAW и LONG RAW. Появился в Oracle9*i*.

registerOutParameterChars(int *paramIndex*, int *sqlType*, int *scale*, int *maxLength*)

Специальная Oracle-версия метода registerOutParameter для регистрации столбцов типов CHAR, VARCHAR, LONG, RAW и LONG RAW. Появился в Oracle9*i*.

sendBatch()

Отправляет наборы параметров в случае применения Oracle-модели группового обновления и возвращает количество затронутых запросом строк.

setExecuteBatch(int *nrows*)

Устанавливает размер группы в случае применения Oracle-модели группового обновления.

wasNULL()

Возвращает логическое значение. Только для Oracle8*i*.

Унаследованные методы

Наследует все методы интерфейса java.sql.CallableStatement в Oracle9*i*, а также методы интерфейсов oracle.jdbc.OraclePreparedStatement и oracle.jdbc.OracleStatement.

OracleConnection

Расширяет интерфейс java.sql.Connection.

Конструктор для данного класса выглядит следующим образом (в Oracle8*i*):

```
OracleConnection(oracle.jdbc.dbaccess.DBAccess access, java.lang.String ur, java.lang.String
us, java.lang.String p, java.lang.String db, java.util.Properties info)
```

Поля

ASCII_TO_CHAR	Статическое целое, только для Oracle8*i*
CACHE_SIZE_NOT_SET	Статическое целое, появилось в Oracle9*i*
CHAR_TO_ASCII	Статическое целое, только для Oracle8*i*
CHAR_TO_JAVACHAR	Статическое целое, только для Oracle8*i*
CHAR_TO_UNICODE	Статическое целое, только для Oracle8*i*
conversion	Oracle.dbc.dbaccess.DBConversion, только для Oracle8*i*
DATABASE_CLOSED	Статическое целое, появилось в Oracle9*i*
DATABASE_NOTOK	Статическое целое, появилось в Oracle9*i*
DATABASE_OK	Статическое целое, появилось в Oracle9*i*
DATABASE_TIMEOUT	Статическое целое, появилось в Oracle9*i*
dataSizeBytes	Статическое java.lang.String, появилось в Oracle9*i*
dataSizeChars	Статическое java.lang.String, появилось в Oracle9*i*
dataSizeUnitsPropertyName	Статическое java.lang.String, появилось в Oracle9*i*
db_access	Oracle.dbc.dbaccess.DBAccess, только для Oracle8*i*
DEBUG	Статическое логическое, только для Oracle8*i*
JAVACHAR_TO_CHAR	Статическое целое, только для Oracle8*i*
lob_dbaccess	Oracle.sql.LobDBACCESSImpl, только для Oracle8*i*
NONE	Статическое целое, только для Oracle8*i*
RAW_TO_ASCII	Статическое целое, только для Oracle8*i*
RAW_TO_JAVACHAR	Статическое целое, только для Oracle8*i*
RAW_TO_UNICODE	Статическое целое, только для Oracle8*i*
UNICODE_TO_CHAR	Статическое целое, только для Oracle8*i*
UsingXA	Логическое, только для Oracle8*i*
XA_wants_error	Логическое, только для Oracle8*i*

Унаследованные поля

Следующие поля унаследованы от java.sql.Connection в версии Oracle9*i*:

TRANSACTION_NONE
TRANSACTION_READ_COMMITTED
TRANSACTION_READ_UNCOMMITTED
TRANSACTION_REPEATABLE_READ
TRANSACTION_SERIALIZABLE

Методы

_getPC()

Возвращает базовое физическое соединение для логического соединения java.sql.Connection. Появился в Oracle9*i*.

archive(int *mode*, int *aseq*, java.lang.String *acstext*)

Исключен из числа рекомендуемых к применению в Oracle9*i* и будет удален в следующей версии.

assertComplete()

Oracle больше не поддерживает Ultra. Исключен из числа рекомендуемых к использованию и будет удален в следующей версии.

clearWarnings()

Очищает предупреждения для объекта соединения. Только для Oracle8*i*.

close()

Откатывает текущую транзакцию. Только для Oracle8*i*.

commit()

Фиксирует изменения для объекта соединения. Только для Oracle8*i*.

createBfileDBAccess()

Возвращает oracle.sql.BfileDBAccess. Только для Oracle8*i*.

createBlobDBAccess()

Возвращает oracle.sql.BlobDBAccess. Только для Oracle8*i*.

createClobDBAccess()

Возвращает oracle.sql.ClobDBAccess. Только для Oracle8*i*.

createStatement()

Возвращает java.sql.Statement. Только для Oracle8*i*.

createStatement(int *resultSetType*, int *resultSetConcurrency*)

JDBC 2.0. Возвращает объект java.sql.Statement, который будет формировать объекты ResultSet с указанным типом и параллелизмом. Только для Oracle8*i*.

createStatementWithKey(java.lang.String *key*)

Возвращает команду java.sql.Statement. Если команда с указанным ключом существует в кэше, то команда возвращается в том виде, в котором она находилась при закрытии и кэшировании с данным ключем. Только для Oracle8*i*.

getAutoClose()

Возвращает логическое значение. Драйвер всегда находится в режиме автоматического закрытия.

getAutoCommit()

Возвращает логическое значение. Только для Oracle8*i*.

getCallWithKey(java.lang.String *key*)

Просматривает явный кэш в поиске совпадения ключа и возвращает java.sql.CallableStatement. Появился в Oracle9*i*.

getClientData(java.lang.Object *key*)

Возвращает java.lang.Object. Только для Oracle8*i*.

getCreateStatementAsRefCursor()

Извлекает текущее логическое значение флага createStatementAsRefCursor, который может быть установлен методом setCreateStatementAsRefCursor. Появился в Oracle9*i*.

getDatabaseProductionVersion()

Возвращает java.lang.String. Только для Oracle8*i*.

getDbCsId()

Возвращает короткое целое – идентификатор набора символов. Только для Oracle8*i*.

getDefaultAutoRefetch()

Возвращает логическое значение. Только для Oracle8*i*.

getDefaultExecuteBatch()

Возвращает целое число, представляющее общее значение размера группы обновления для данного соединения.

getDefaultRowPrefetch()

Возвращает целое число, представляющее размер выборки с опережением для всех команд, относящихся к данному соединению и созданных после установки данного значения.

getDescriptor(java.lang.String *sql_name*)

Возвращает объект Descriptor, соответствующий указанному типу SQL.

getExplicitCachingEnabled()

Возвращает True, если явный кэш в данный момент разрешен, и False в противном случае. Появился в Oracle9*i*.

getImplicitCachingEnabled()

Возвращает True, если неявный кэш в данный момент разрешен, и False в противном случае. Появился в Oracle9*i*.

getIncludeSynonyms()

Возвращает логическое значение, указывающее, следует ли включать информацию о синонимах в DatabaseMetaData.getColumns.

getJavaObject(java.lang.String *sql_name*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

getJdbcCsId()

Возвращает короткое целое – идентификатор набора символов. Только для Oracle8*i*.

getMetaData()

Возвращает java.sql.DatabaseMetaData. Только для Oracle8*i*.

getProperties()

Определяет свойства соединения и возвращает их в виде java.util.Properties. Появился в Oracle9*i*.

getRemarksReporting()

Возвращает логическое значение, указывающее, будет ли вызов getTables или getColumns интерфейса DatabaseMetaData сообщать о столбце REMARKS.

getReportRemarks()

Возвращает логическое значение. Только для Oracle8*i*.

getRestrictGetTables()

Возвращает статус ограничения в виде логического значения для возвращаемых данных DatabaseMetaData.getTables.

getSessionTimeZone()

Возвращает региональное имя часового пояса сеанса Oracle в виде java.lang. String. Появился в Oracle9*i*.

getSQLType(java.lang.Object *obj*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

getStatementCacheSize()

Возвращает целое число – текущий размер кэша приложения. Появился в Oracle9*i*.

getStatementWithKey(java.lang.String *key*)

Просматривает явный кэш в поиске совпадения ключа и возвращает java.sql.PreparedStatement. Появился в Oracle9*i*.

getStmtCacheSize()

Исключен из числа рекомендуемых к применению в Oracle9*i*. Используйте вместо него метод getStatementCacheSize().

getStructAttrCsId()

Возвращает в виде короткого целого Oracle-идентификатор набора символов, используемого в атрибутах STRUCT.

getSynchronousMode()

Исключен из числа рекомендуемых к применению в Oracle9*i* Release 2. Появился в Oracle9*i*.

getTypeMap()

Возвращает java.util.Dictionary. Только для Oracle8*i*.

getUserName()

Возвращает *имя_пользователя* текущего соединения в виде java.lang.String.

getUsingXAFlag()

Исключен из числа рекомендуемых к применению.

getVersionNumber()

Возвращает короткое целое. Только для Oracle8*i*.

getWarnings()

Возвращает java.sql.SQLWarning. Только для Oracle8*i*.

getXAErrorFlag()

Возвращает флаг ошибки XA (в классе OracleXAException для Oracle9*i*).

holdLine(OracleStatement *stmt*)

Только для Oracle8*i*.

initUserName()

Только для Oracle8*i*.

isClosed()

Возвращает логическое значение. Только для Oracle8*i*.

isLogicalConnection()

Возвращает логическое значение, которое указывает, является ли соединение логическим. Появился в Oracle9*i*.

isReadOnly()

Возвращает логическое значение. Только для Oracle8*i*.

nativeSQL(java.lang.String *sql*)

Возвращает java.lang.String. Только для Oracle8*i*.

needLine()

Только для Oracle8*i*.

openJoltConnection(java.lang.String *apiName*, short *major*, short *minor*)

Появился в Oracle9*i*. Исключен из числа рекомендуемых к применению в Oracle9*i* Release 2.

oracleReleaseSavepoint(OracleSavepoint *savepoint*)

Удаляет указанный объект OracleSavepoint из текущей транзакции. Появился в Oracle9*i*.

oracleRollback(OracleSavepoint *savepoint*)

Отменяет все изменения, выполненные после установки указанного объекта OracleSavepoint. Появился в Oracle9*i*.

oracleSetSavepoint()

Создает точку сохранения без имени в текущей транзакции и возвращает новый объект OracleSavepoint, который ее представляет. Появился в Oracle9*i*.

oracleSetSavepoint(java.lang.String *name*)

Создает точку сохранения с указанным именем в текущей транзакции и возвращает новый объект OracleSavepoint, который ее представляет. Появился в Oracle9*i*.

pingDatabase(int *timeOut*)

Посылает базе данных запрос проверки доступности и возвращает целое число. Появился в Oracle9*i*.

prepareCall(java.lang.String *sql*)

Возвращает java.sql.CallableStatement. Только для Oracle8*i*.

prepareCall(java.lang.String *sql*, int *resultSetType*, int *resultSetConcurrency*)

Возвращает java.sql.CallableStatement. Только для Oracle8*i*.

prepareCallWithKey(java.lang.String *key*)

Исключен из числа рекомендуемых к применению в Oracle9*i*. Метод аналогичен prepareCall, но есть одно отличие. Если команда CallableStatement с указанным ключом существует в кэше, то команда возвращается в том виде, в котором она находилась при закрытии и кэшировании с данным ключом. Объект, возвращенный из кэша по ключу, будет иметь статус "KEYED". Если такая команда CallableStatement не найдена, то возвращается NULL. Ключ *key* не может содержать NULL.

prepareStatement(java.lang.String *sql*)

Возвращает java.sql.PreparedStatement. Только для Oracle8*i*.

prepareStatement(java.lang.String *sql*, int *resultSetType*, int *resultSetConcurrency*)

Возвращает java.sql.PreparedStatement. Только для Oracle8*i*.

prepareStatementWithKey(java.lang.String *key*)

Исключен из числа рекомендуемых к применению в Oracle9*i* (см. пояснения для prepareCallWithKey).

purgeExplicitCache()

Удаляет все существующие команды из явного кэша, очищая его. Появился в Oracle9*i*.

purgeImplicitCache()

Удаляет все существующие команды из неявного кэша, очищая его. Появился в Oracle9*i*.

putDescriptor(java.lang.String *sql_name*, java.lang.Object *desc*)

Сохраняет дескриптор объекта для дальнейшего использования.

registerApiDescription(java.lang.String *apiName*, short *major*, short *minor*, java.lang.String *className*)

Появился в Oracle9*i*. Исключен из числа рекомендуемых к применению в Oracle9*i* Release 2.

registerSQLType(java.lang.String *sql_name*, java.lang.Class *java_class*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

registerSQLType(java.lang.String *sql_name*, java.lang.String *java_class_name*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

registerTAFCallback(OracleOCIFailover *cbk*, java.lang.Object *obj*)

Регистрирует экземпляр функции обратного вызова TAF, который будет вызываться при сбое приложения. Появился в Oracle9*i*.

releaseLine()

Только для Oracle8*i*.

removeClientData(java.lang.Object *key*)

Возвращает java.lang.Object. Только для Oracle8*i*.

rollback()

Удаляет изменения, начиная с последней фиксации или отката, и освобождает блокировки БД. Только для Oracle8*i*.

setAutoClose(boolean *autoClose*)

Режим автозакрытия всегда включен.

setAutoCommit(boolean *autoCommit*)

Только для Oracle8*i*.

setClientData(java.lang.Object *key*, java.lang.Object *value*)

Возвращает java.lang.Object. Только для Oracle8*i*.

setCreateStatementAsRefCursor(boolean *value*)

Если передаваемое в параметре значение равно True, то любые новые команды для данного соединения будут создаваться как REF CURSOR. Появился в Oracle9*i*.

setDefaultAutoRefetch(boolean *autoRefresh*)

Задает значение по умолчанию для автоматического повторного извлечения строк в процессе обновления. Только для Oracle8*i*.

setDefaultExecuteBatch(int *batch*)

Задает значение по умолчанию для размера группы в случае применения Oracle-модели группового обновления (значение по умолчанию равно 1).

setDefaultRowPrefetch(int *value*)

Задает значение упреждающей выборки строк для всех команд, связанных с данным соединением и созданных после установки данного значения. Происходит сопоставление параметра упреждающей выборки строк с указанным объектом команды.

setExplicitCachingEnabled(boolean *cache*)

Разрешает и запрещает использование явного кэша. Появился в Oracle9*i*.

setImplicitCachingEnabled(boolean *cache*)

Разрешает и запрещает использование неявного кэша. Появился в Oracle9*i*.

setIncludeSynonyms(boolean *synonyms*)

Разрешает и запрещает извлечение информации о синонимах в DatabaseMetaData.

setReadOnly(boolean *readOnly*)

Только для Oracle8*i*.

setRemarksReporting(boolean *reportRemarks*)

Включает и выключает вывод сведений из столбцов REMARKS вызовами getTables и getColumns интерфейса DatabaseMetaData. Если требуется включать сведения из столбцов REMARKS, то обращения к DatabaseMetaData.getTables и getColumns будут чрезвычайно медленными, т. к. они требуют дорогостоящего внешнего соединения. Поэтому по умолчанию драйвер JDBC не сообщает о столбцах REMARKS.

setRestrictGetTables(boolean *restrict*)

Включает и выключает ограничение на возвращаемые данные в методе DatabaseMetaData.getTables, возвращающем информацию о доступных таблицах, представлениях и синонимах.

setSessionTimeZone(java.lang.String *regionName*)

Задает часовой пояс сеанса. Появился в Oracle9*i*.

setStatementCacheSize(int *size*)

Задает размер кэша приложения (который будет использоваться как для явного, так и для неявного кэширования). Появился в Oracle9*i*.

setStmtCacheSize(int *size*)

Исключен из числа рекомендуемых к применению в Oracle9*i*. Используйте вместо него setStatementCacheSize().

setStmtCacheSize(int *size*, boolean *clearMetaData*)

Исключен из числа рекомендуемых к применению в Oracle9*i*. Используйте вместо него setStatementCacheSize().

setSynchronousMode(boolean *isSynchronous*)

Появился в Oracle9*i*. Исключен из числа рекомендуемых к применению в Oracle9*i* Release 2.

setTypeMap(java.util.Dictionary *map*)

Устанавливает map как карту типов для соединения. Только для Oracle8*i*.

setUsingXAFlag(boolean *value*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

setWrapper(OracleConnection *wrapper*)

Задает объект-обертку. Появился в Oracle9*i*.

setXAErrorFlag(boolean *value*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

shutdown(int *mode*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

startup(java.lang.String *startup_str*, int *mode*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

trace(java.lang.String *s*)

Только для Oracle8*i*.

unwrap()

Возвращает объект-обертку в виде OracleConnection (или NULL). Появился в Oracle9*i*.

Унаследованные методы

Наследует все методы интерфейса java.sql.Connection.

Наследует все методы java.lang.Object, за исключением clone, equals и finalize.

OracleJdbc2SQLInput

Расширяет интерфейс java.lang.Object. Реализует интерфейс oracle.jdbc2.SQLInput. Только для Oracle8*i*.

Методы

readArray()

Возвращает oracle.jdbc2.Array из потока.

readARRAY()

Возвращает ARRAY, следующий атрибут в потоке, в виде oracle.sql.ARRAY.

readAsciiStream()

Возвращает java.io.InputStream как следующий атрибут в потоке.

readBFILE()

Возвращает BFILE как следующий атрибут в потоке.

readBigDecimal()

Возвращает java.math.BigDecimal как следующий атрибут в потоке.

readBinaryStream()

Возвращает java.io.InputStream как следующий атрибут в потоке.

readBlob()

Возвращает oracle.jdbc2.Blob как следующий атрибут в потоке.

readBLOB()

Возвращает BLOB как следующий атрибут в потоке.

readBoolean()

Возвращает логическое значение как следующий атрибут в потоке.

readByte()

Возвращает байт как следующий атрибут в потоке.

readBytes()

Возвращает массив байтов как следующий атрибут в потоке.

readCHAR()

Возвращает oracle.sql.CHAR как следующий атрибут в потоке.

readCharacterStream()

Возвращает java.io.Reader как следующий атрибут в потоке.

readClob()
Возвращает oracle.jdbc2.Clob как следующий атрибут в потоке.
readCLOB()
Возвращает CLOB как следующий атрибут в потоке.
readDATE()
Возвращает значение типа DATE как следующий атрибут в потоке.
readDouble()
Возвращает значение типа Double как следующий атрибут в потоке.
readFloat()
Возвращает вещественное число как следующий атрибут в потоке.
readInt()
Возвращает целое число как следующий атрибут в потоке.
readLong()
Возвращает длинное целое как следующий атрибут в потоке.
readNUMBER()
Возвращает NUMBER как следующий атрибут в потоке.
readObject()
Возвращает java.lang.Object как элемент данных в начале потока.
readOracleObject()
Возвращает java.lang.Object как следующий атрибут в потоке.
readRAW()
Возвращает RAW как следующий атрибут в потоке.
readRef()
Возвращает oracle.jdbc2.Ref как следующий атрибут в потоке.
readREF()
Возвращает REF как следующий атрибут в потоке.
readROWID()
Возвращает ROWID как следующий атрибут в потоке.
readShort()
Возвращает короткое целое как следующий атрибут в потоке.
readString()
Возвращает java.lang.String как следующий атрибут в потоке.
readStruct()
Возвращает oracle.jdbc2.Struct как следующий атрибут в потоке.
readSTRUCT()
Возвращает STRUCT как следующий атрибут в потоке.
readTime()
Возвращает java.sql.Time как следующий атрибут в потоке.
readTimestamp()
Возвращает java.sql.Timestamp как следующий атрибут в потоке.

wasNull()

Возвращает логическое значение.

Унаследованные методы

Наследует все методы интерфейса java.lang.Object, кроме clone и finalize.

OracleOCIFailover

Обрабатывает сбой на уровне OCI. Появился в Oracle9*i*.

Поля

FO_ABORT	Статическое целое
FO_BEGIN	Статическое целое
FO_END	Статическое целое
FO_ERROR	Статическое целое
FO_EVENT_UNKNOWN	Статическое целое
FO_NONE	Статическое целое
FO_REAUTH	Статическое целое
FO_RETRY	Статическое целое
FO_SELECT	Статическое целое
FO_SESSION	Статическое целое
FO_TYPE_UNKNOWN	Статическое целое

Метод

callbackFn(java.sql.Connection *conn*, java.lang.Object *ctxt*, int *type*, int *event*)

Возвращает целое число.

OracleParameterMetaData

Расширяет интерфейс java.sql.ParameterMetaData. Появился в Oracle9*i*.

Поля

ParameterModeIn	Статическое целое
ParameterModeInOut	Статическое целое
ParameterModeOut	Статическое целое
ParameterModeUnknown	Статическое целое
ParameterNoNulls	Статическое целое
ParameterNullable	Статическое целое
ParameterNullableUnknown	Статическое целое

Методы

getParameterClassName(int *param*)

Возвращает полное уточненное имя класса Java (в виде java.lang.String), экземпляры которого должны быть переданы методу PreparedStatement.setObject.

getParameterCount()

Возвращает целое число – количество параметров в объекте PreparedStatement, для которого данный объект OracleParameterMetaData содержит информацию.

getParameterMode(int *param*)

Возвращает целое число – режим указанного параметра.

getParameterType(int *param*)

Возвращает целое число – SQL-тип указанного параметра.

getParameterTypeName(int *param*)

Возвращает java.lang.String – имя специфичного для БД имени типа указанного параметра.

getPrecision(int *param*)

Возвращает целое число – количество разрядов указанного параметра.

getScale(int *param*)

Возвращает целое число – количество разрядов дробной части указанного параметра.

isNullable(int *param*)

Возвращает целое число, указывающее, разрешены ли в данном параметре значения NULL.

isSigned(int *param*)

Возвращает логическое значение, указывающее, могут ли значения указанного параметра быть числами со знаком.

OraclePreparedStatement

Расширяет интрефейс java.sql.PreparedStatementдля Oracle9*i*; расширяет OracleStatement в Oracle8*i*. В Oracle8*i* реализует oracle.jdbc.internal.OraclePreparedStatement.

Поля

FORM_CHAR	Статическое целое, появилось в Oracle9*i*
FORM_NCHAR	Статическое целое, появилось в Oracle9*i*

Унаследованные поля

Следующие поля унаследованы от oracle.jdbc.OracleStatement:

auto_rollback	Только для Oracle8*i*
closed	Только для Oracle8*i*
dbstmt	Только для Oracle8*i*
defines	Только для Oracle8*i*
EXPLICIT	
IMPLICIT	
NEW	
wait_option	Только для Oracle8*i*

Методы

closeWithKey(java.lang.String *key*)

Базовый курсор не закрывается, дескриптор команды кэшируется с ключом *key*, при этом состояние, данные и метаданные не очищаются. Команда может быть впоследствии извлечена по данному ключу. Только для Oracle8*i*.

defineParameterType(int *param_index*, int *type*, int *max_size*)

Определяет тип, который будет применяться при связывании параметра, и максимальный размер (в символах) данных, которые будут связаны.

defineParameterTypeBytes(int *param_index*, int *type*, int *max_size*)

Определяет тип, который будет применяться при связывании параметра, и максимальный размер (в байтах) данных, которые будут связаны. Появился в Oracle9*i*.

defineParameterTypeChars(int *param_index*, int *type*, int *max_size*)

Определяет тип, который будет применяться при связывании параметра, и максимальный размер (в символах) данных, которые будут связаны. Появился в Oracle9*i*.

getExecuteBatch()

Возвращает размер группы для данной команды в случае применения Oracle-модели группового обновления (по умолчанию устанавливается объектом соединения).

OracleGetParameterMetaData()

Извлекает количество, типы и свойства параметров объекта PreparedStatement в виде OracleParameterMetaData.

sendBatch()

Отправляет любую существующую группу (при использовании Oracle-модели группового обновления) и возвращает целое число.

setARRAY(int *paramIndex*, ARRAY *arr*)

Связывает указанный параметр с массивом oracle.sql.ARRAY.

setBfile(int *paramIndex*, BFILE *file*)

Связывает указанный параметр со значением oracle.sql.BFILE.

setBFILE(int *paramIndex*, BFILE *file*)

Связывает указанный параметр со значением oracle.sql.BFILE.

setBLOB(int *paramIndex*, BLOB *lob*)

Связывает указанный параметр со значением oracle.sql.BLOB.

setCHAR(int *paramIndex*, CHAR *ch*)

Связывает указанный параметр со значением oracle.sql.CHAR.

setCheckBindTypes(boolean *flag*)

Включает/выключает проверки типа при связывании.

setCLOB(int *paramIndex*, CLOB *lob*)

Связывает указанный параметр со значением oracle.sql.CLOB.

setCursor(int *paramIndex*, java.sql.ResultSet *rs*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

setCustomDatum(int *paramIndex*, CustomDatum *x*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

setDATE(int *paramIndex*, DATE *date*)

Связывает указанный параметр со значением oracle.sql.DATE.

setDisableStmtCaching(boolean *cache*)

Означает, что данный объект не следует кэшировать, даже если для соответствующего объекта соединения кэширование разрешено.

setExecuteBatch(int *batchValue*)

Задает размер группы для данной команды при использовании Oracle-модели группового обновления (по умолчанию устанавливается объектом соединения).

setFixedCHAR(int *paramIndex*, java.lang.String *x*)

Приводит указанный параметр к типу String и выполняет сравнение без дополнения заполнителями с SQL CHAR.

setFormOfUse(int *paramIndex*, short *formOfUse*)

Определяет, связываются ли данные для типа данных SQL NCHAR.

setINTERVALYM(int *paramIndex*, INTERVALYM *x*)

Связывает указанный параметр со значением oracle.sql.INTERVALYM.

setNUMBER(int *paramIndex*, NUMBER *num*)

Связывает указанный параметр со значением oracle.sql.NUMBER.

setOPAQUE(int *paramIndex*, OPAQUE *val*)

Связывает указанный параметр с oracle.sql.OPAQUE.

setOracleObject(int *paramIndex*, Datum *x*)

Связывает указанный параметр со значением oracle.sql.Datum.

setORAData(int *paramIndex*, ORAData *x*)

Связывает указанный параметр со значением oracle.sql.ORAData.

setPlsqlIndexTable(int *paramIndex*, java.lang.Object *arrayData*, int *maxLen*, int *curLen*, int *elemSqlType*, int *elemMaxLen*)

Связывает входной (IN) параметр с ассоциативным массивом PL/SQL.

setRAW(int *paramIndex*, RAW *raw*)

Связывает указанный параметр со значением oracle.sql.RAW.

setREF(int *paramIndex*, REF *ref*)

Связывает указанный параметр со значением oracle.sql.REF.

setRefType(int *paramIndex*, REF *ref*)

Связывает указанный параметр со значением oracle.sql.REF.

setROWID(int *paramIndex*, ROWID *rowid*)

Связывает указанный параметр со значением oracle.sql.ROWID.

setSTRUCT(int *paramIndex*, STRUCT *struct*)

Связывает указанный параметр со значением oracle.sql.STRUCT.

setStructDescriptor(int *paramIndex*, StructDescriptor *desc*)

Устанавливет тип связывания указанного параметра из oracle.sql.StructDescriptor.

setTIMESTAMP(int *paramIndex*, TIMESTAMP *x*)

Связывает указанный параметр со значением oracle.sql.TIMESTAMP.

setTIMESTAMPLTZ(int *paramIndex*, TIMESTAMPLTZ *x*)

Связывает указанный параметр со значением oracle.sql.TIMESTAMPLTZ.

setTIMESTAMPTZ(int *paramIndex*, TIMESTAMPTZ *x*)

Связывает указанный параметр со значением oracle.sql.TIMESTAMPTZ.

Унаследованные методы

Наследует все методы интерфейса java.sql.PreparedStatement, за исключением executeQuery.

Наследует все методы интерфейса oracle.jdbc.OracleStatement.

OracleResultSet

Расширяет интерфейс java.sql.ResultSet.

Унаследованные поля

Следующие поля унаследованы от java.sql.ResultSet:

CONCUR_READ_ONLY
CONCUR_UPDATABLE
FETCH_FORWARD
FETCH_REVERSE
FETCH_UNKNOWN
TYPE_FORWARD_ONLY
TYPE_SCROLL_INSENSITIVE
TYPE_SCROLL_SENSITIVE

Методы

getARRAY(int *columnIndex*)

Возвращает ARRAY из столбца с индексом *columnIndex*.

getARRAY(java.lang.String *columnName*)

Возвращает ARRAY из столбца с именем *columnName*.

getBfile(int *columnIndex*)

Возвращает BFILE из столбца с индексом *columnIndex*.

getBFILE(int *columnIndex*)

Возвращает BFILE из столбца с индексом *columnIndex*.

getBfile(java.lang.String *columnName*)

Возвращает BFILE из столбца с именем *columnName*.

getBFILE(java.lang.String *columnName*)

Возвращает BFILE из столбца с именем *columnName*.

getBLOB(int *columnIndex*)

Возвращает BLOB из столбца с индексом *columnIndex*.

getBLOB(java.lang.String *columnName*)

Возвращает BLOB из столбца с именем *columnName*.

getCHAR(int *columnIndex*)

Возвращает CHAR из столбца с индексом *columnIndex*.

getCHAR(java.lang.String *columnName*)

Возвращает CHAR из столбца с именем *columnName*.

getCLOB(int *columnIndex*)

Возвращает CLOB из столбца с индексом *columnIndex*.

getCLOB(java.lang.String *columnName*)

Возвращает CLOB из столбца с именем *columnName*.

getCursor(int *columnIndex*)

Возвращает java.sql.ResultSet из столбца с индексом *columnIndex*.

getCursor(java.lang.String *columnName*)

Возвращает java.sql.ResultSet из столбца с именем *columnName*.

getCustomDatum(int *columnIndex*, CustomDatumFactory *factory*)

Возвращает CustomDatum. Исключен из числа рекомендуемых к применению в Oracle9*i*.

getCustomDatum(java.lang.String *columnName*, CustomDatumFactory *factory*)

Возвращает CustomDatum. Исключен из числа рекомендуемых к применению в Oracle9*i*.

getDATE(int *columnIndex*)

Возвращает DATE из столбца с индексом *columnIndex*.

getDATE(java.lang.String *columnName*)

Возвращает DATE из столбца с именем *columnName*.

getINTERVALYM(int *columnIndex*)

Возвращает значение oracle.sql.INTERVALYM из столбца с индексом *columnIndex*.

getINTERVALYM(java.lang.String *columnName*)

Возвращает значение oracle.sql.INTERVALYM из столбца с именем *columnName*.

getNUMBER(int *columnIndex*)

Возвращает NUMBER из столбца с индексом *columnIndex*.

getNUMBER(java.lang.String *columnName*)

Возвращает NUMBER из столбца с именем *columnName*.

getOPAQUE(int *columnIndex*)

Возвращает OPAQUE из столбца с индексом *columnIndex*.

getOPAQUE(java.lang.String *columnName*)

Возвращает OPAQUE из столбца с именем *columnName*.

getOracleObject(int *columnIndex*)

Возвращает Datum из столбца с индексом *columnIndex*.

getOracleObject(java.lang.String columnName)

Возвращает Datum из столбца с именем *columnName*.

getORAData(int *columnIndex*, ORADataFactory *factory*)

Возвращает ORAData из столбца с индексом *columnIndex*.

getORAData(java.lang.String *columnName*, ORADataFactory *factory*)

Возвращает ORAData из столбца с именем *columnName*.

getRAW(int *columnIndex*)

Возвращает RAW из столбца с индексом *columnIndex*.

getRAW(java.lang.String *columnName*)

Возвращает RAW из столбца с именем *columnName*.

getREF(int *columnIndex*)

Возвращает REF из столбца с индексом *columnIndex*.

getREF(java.lang.String *columnName*)

Возвращает REF из столбца с именем *columnName*.

getROWID(int *columnIndex*)

Возвращает ROWID из столбца с индексом *columnIndex*.

getROWID(java.lang.String *columnName*)

Возвращает ROWID из столбца с именем *columnName*.

getSTRUCT(int *columnIndex*)

Возвращает STRUCT из столбца с индексом *columnIndex*.

getSTRUCT(java.lang.String *columnName*)

Возвращает STRUCT из столбца с именем *columnName*.

getTIMESTAMP(int *columnIndex*)

Возвращает значение oracle.sql.TIMESTAMP из столбца с индексом *columnIndex*.

getTIMESTAMP(java.lang.String *colName*)

Возвращает oracle.sql.TIMESTAMP из столбца с именем *colName*.

getTIMESTAMPLTZ(int *columnIndex*)

Возвращает значение oracle.sql.TIMESTAMPLTZ из столбца с индексом *columnIndex*.

getTIMESTAMPLTZ(java.lang.String *colName*)

Возвращает значение oracle.sql.TIMESTAMPLTZ из столбца с именем *colName*.

getTIMESTAMPTZ(int *columnIndex*)

Возвращает значение oracle.sql.TIMESTAMPTZ из столбца с индексом *columnIndex*.

getTIMESTAMPTZ(java.lang.String *colName*)

Возвращает значение oracle.sql.TIMESTAMPTZ из столбца с именем *colName*.

updateArray(int *columnIndex*, java.sql.Array *x*)

Обновляет столбец с индексом *columnIndex* значением типа java.sql.Array.

updateARRAY(int *columnIndex*, ARRAY *x*)

Обновляет столбец с индексом *columnIndex* значением типа oracle.sql.ARRAY.

updateArray(java.lang.String *columnName*, java.sql.Array *x*)

Обновляет столбец с именем *columnName* значением типа java.sql.Array.

updateARRAY(java.lang.String *columnName*, ARRAY *x*)

Обновляет столбец с именем *columnName* значением типа oracle.sql.ARRAY.

updateBfile(int *columnIndex*, BFILE *x*)

Обновляет столбец с индексом *columnIndex* значением типа BFILE.

updateBFILE(int *columnIndex*, BFILE *x*)

Обновляет столбец с индексом *columnIndex* значением типа BFILE.

updateBfile(java.lang.String *columnName*, BFILE *x*)

Обновляет столбец с именем *columnName* значением типа BFILE.

updateBFILE(java.lang.String *columnName*, BFILE *x*)

Обновляет столбец с именем *columnName* значением типа BFILE.

updateBlob(int *columnIndex*, java.sql.Blob *x*)

Обновляет столбец с индексом *columnIndex* значением типа java.sql.Blob.

updateBLOB(int *columnIndex*, BLOB *x*)

Обновляет столбец с индексом *columnIndex* значением типа BLOB.

updateBlob(java.lang.String *columnName*, java.sql.Blob *x*)

Обновляет столбец с именем *columnName* значением типа java.sql.Blob.

updateBLOB(java.lang.String *columnName*, BLOB *x*)

Обновляет столбец с именем *columnName* значением типа BLOB.

updateCHAR(int *columnIndex*, CHAR *x*)

Обновляет столбец с индексом *columnIndex* значением типа CHAR.

updateCHAR(java.lang.String *columnName*, CHAR *x*)

Обновляет столбец с именем *columnName* значением типа CHAR.

updateClob(int *columnIndex*, java.sql.Clob *x*)

Обновляет столбец с индексом *columnIndex* значением типа java.sql.Clob.

updateCLOB(int *columnIndex*, CLOB *x*)

Обновляет столбец с индексом *columnIndex* значением типа CLOB.

updateClob(java.lang.String *columnName*, java.sql.Clob *x*)

Обновляет столбец с именем *columnName* значением типа java.sql.Clob.

updateCLOB(java.lang.String *columnName*, CLOB *x*)

Обновляет столбец с именем *columnName* значением типа CLOB.

updateCustomDatum(int *columnIndex*, CustomDatum *x*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

updateCustomDatum(java.lang.String *columnName*, CustomDatum *x*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

updateDATE(int *columnIndex*, DATE *x*)

Обновляет столбец с индексом *columnIndex* значением типа DATE.

updateDATE(java.lang.String *columnName*, DATE *x*)

Обновляет столбец с именем *columnName* значением типа DATE.

updateNUMBER(int *columnIndex*, NUMBER *x*)

Обновляет столбец с индексом *columnIndex* значением типа NUMBER.

updateNUMBER(java.lang.String *columnName*, NUMBER *x*)

Обновляет столбец с именем *columnName* значением типа NUMBER.

updateOracleObject(int *columnIndex*, Datum *x*)

Обновляет столбец с индексом *columnIndex* значением типа OracleObject.

updateOracleObject(java.lang.String *columnName*, Datum *x*)

Обновляет столбец с именем *columnName* значением типа OracleObject.

updateORAData(int *columnIndex*, ORAData *x*)

Обновляет столбец с индексом *columnIndex* значением типа OraData.

updateORAData(java.lang.String *columnName*, ORAData *x*)

Обновляет столбец с именем *columnName* значением типа OraData.

updateRAW(int *columnIndex*, RAW *x*)

Обновляет столбец с индексом *columnIndex* значением типа RAW.

updateRAW(java.lang.String *columnName*, RAW *x*)

Обновляет столбец с именем *columnName* значением типа RAW.

updateRef(int *columnIndex*, java.sql.Ref *x*)

Обновляет столбец с индексом *columnIndex* значением типа java.sql.Ref.

updateREF(int *columnIndex*, REF *x*)

Обновляет столбец с индексом *columnIndex* значением типа REF.

updateRef(java.lang.String *columnName*, java.sql.Ref *x*)

Обновляет столбец с именем *columnName* значением типа java.sql.Ref.

updateREF(java.lang.String *columnName*, REF *x*)

Обновляет столбец с именем *columnName* значением типа REF.

updateROWID(int *columnIndex*, ROWID *x*)

Обновляет столбец с индексом *columnIndex* значением типа ROWID.

updateROWID(java.lang.String *columnName*, ROWID *x*)

Обновляет столбец с именем *columnName* значением типа ROWID.

updateSTRUCT(int *columnIndex*, STRUCT *x*)

Обновляет столбец с индексом *columnIndex* значением типа STRUCT.

updateSTRUCT(java.lang.String *columnName*, STRUCT *x*)

Обновляет столбец с именем *columnName* значением типа STRUCT.

Унаследованные методы

Наследует все методы java.sql.ResultSet, за исключением insertRow.

OracleResultSetCache

Обеспечивает интерфейс для кэша результирующего множества Oracle.

Методы

clear()

Удаляет все данные из кэша результирующего множества.

close()

Закрывает кэш результирующего множества.

get(int *i*, int *j*)

Возвращает данные, хранящиеся в *i*-ой строке и *j*-ом столбце в виде java.lang.Object.

put(int *i*, int *j*, java.lang.Object *value*)

Возвращает данные *i*-ой строки и *j*-ого столбца в java.lang.Object.

remove(int *i*)

Удаляет *i*-ю строку из кэша результирующего множества.

remove(int *i*, int *j*)

Возвращает данные, хранящиеся в *i*-й строке и *j*-м столбце кэша результирующего множества.

OracleResultSetMetaData

Расширяет интерфейс java.sql.ResultSetMetaData.

Унаследованные поля

Следующие поля унаследованы от java.sql.ResultSetMetaData:

columnNoNulls
columnNullable
columnNullableUnknown

Метод

isNCHAR(int *index*)

Возвращает логическое значение, указывающее, относится ли индексная запись к типу NCHAR, NVARCHAR или NCLOB.

Унаследованные методы

Наследует все методы от java.sql.ResultSetMetaData.

OracleSavepoint

Расширяет интерфейс java.sql.Savepoint.

Методы

getSavepointId()

Возвращает сформированный идентификатор (целое число) точки сохранения, представленной данным объектом OracleSavepoint.

getSavepointName()

Возвращает в виде java.lang.String имя точки сохранения, представленной данным объектом OracleSavepoint. Появился в Oracle9*i*.

OracleStatement

Расширяет интерфейс java.sql.Statement. OracleCallableStatement и OraclePreparedStatement представляют собой его подынтерфейсы. OracleStatement представляет собой класс oracle.jdbc.driver в Oracle8*i* с некоторыми дополнительными методами (за подробной информацией обращайтесь к документации Oracle).

Поля

EXPLICIT Статическое целое

IMPLICIT	Статическое целое
NEW	Статическое целое

Методы

clearDefines()

Очищает ранее определенные типы для определяемых столбцов данной команды.

closeWithKey(java.lang.String *key*)

Базовый курсор не закрывается, дескриптор команды кэшируется с ключом, при этом состояние, данные и метаданные не очищаются. Команда может быть впоследствии извлечена по данному ключу. Только для Oracle8*i*.

creationState()

Исключен из числа рекомендуемых к применению в Oracle9*i*.

defineColumnType(int *column_index*, int *type*)

Определяет тип, задаваемый при извлечении данных из столбца определенной таблицы БД с индексом *column_index*.

defineColumnType(int *column_index*, int *type*, int *max_size*)

Определяет тип, задаваемый при извлечении данных из столбца определенной таблицы БД с индексом *column_index* и максимальным размером *max_size*.

defineColumnType(int *column_index*, int *typeCode*, java.lang.String *typeName*)

Определяет тип *typeCode*, задаваемый при извлечении данных типа *typeName* из столбца определенной таблицы БД с индексом *column_index*.

defineColumnTypeBytes(int *column_index*, int *type*, int *max_size*)

Определяет тип, задаваемый при извлечении данных из столбца определенной таблицы БД с индексом *column_index* и максимальным размером *max_size* в байтах.

defineColumnTypeChars(int *column_index*, int *type*, int *max_size*)

Определяет тип, задаваемый при извлечении данных из столбца определенной таблицы БД с индексом *column_index* и максимальным размером *max_size* в символах.

getRowPrefetch()

Возвращает значение размера упреждающей выборки строк для результирующих множеств, созданных этой командой.

isNCHAR(int *index*)

Возвращает логическое значение, указывающее, относится ли индексная запись к типу NCHAR, NVARCHAR или же NCLOB.

setResultSetCache(OracleResultSetCache *cache*)

Задает конкретную реализацию кэша для позиционируемых результирующих множеств с клиентской стороны.

setRowPrefetch(int *value*)

Задает значение размера упреждающей выборки строк для всех результирующих множеств, созданных данной командой.

Унаследованные методы

Наследует все методы от java.sql.Statement.

StructMetaData

Расширяет интерфейс OracleResultSetMetaData.

Унаследованные поля

Следующие поля унаследованы от java.sql.ResultSetMetaData:

columnNoNulls
columnNullable
columnNullableUnknown

Методы

getAttributeJavaName(int *column*)

Возвращает внешнее имя атрибута JAVA_STRUCT в виде java.lang.String.

getLocalColumnCount()

Возвращает количество локальных атрибутов виде целого числа.

getOracleColumnClassName(int *column*)

Возвращает в виде java.lang.String полное уточненное имя класса Datum, экземпляры которого формируются, если для извлечения значения из столбца вызывается метод OracleResultSet.getOracleObject.

isInherited(int *column*)

Возвращает логическое значение, которое указывает, унаследован ли атрибут от типа более высокого уровня.

Унаследованные методы

Наследует все методы от OracleResultSetMetaData.

Наследует все методы от java.sql.ResultSetMetaData.

Классы

В пакет oracle.jdbc входят следующие классы.

OracleConnectionWrapper

Расширяет java.lang.Object и реализует OracleConnection. Появился в Oracle9*i*.

Конструктор для данного класса выглядит следующим образом:

```
OracleConnectionWrapper(OracleConnection toBeWrapped)
```

Методы

_getPC()

Возвращает объект java.sql.Connection.

archive(int *mode*, int *aseq*, java.lang.String *acstext*)

Исключен из числа рекомендуемых к применению и будет удален в следующей версии.

assertComplete()

Oracle больше не поддерживает Ultra. Исключен из числа рекомендуемых к применению и будет удален в следующей версии.

clearWarnings()

Очищает предупреждения для соединения.

close()

Закрывает соединение.

commit()

Фиксирует соединение.

createStatement()

Возвращает объект java.sql.Statement.

createStatement(int *resultSetType*, int *resultSetConcurrency*)

Возвращает объект java.sql.Statement.

createStatement(int *resultSetType*, int *resultSetConcurrency*, int *resultSetHoldability*)

Возвращает объект java.sql.Statement.

getAutoClose()

Возвращает логическое значение, указывающее режим автоматического закрытия.

getAutoCommit()

Возвращает логическое значение, указывающее режим автоматической фиксации.

getCallWithKey(java.lang.String *key*)

Возвращает объект java.sql.CallableStatement, соответствующий ключу *key*.

getCatalog()

Возвращает имя каталога в виде java.lang.String.

getCreateStatementAsRefCursor()

Возвращает логическое значение, указывающее текущее состояние CreateStatementAsRefCursor.

getDefaultExecuteBatch()

Возвращает целое число – количество операций в группе по умолчанию.

getDefaultRowPrefetch()

Возвращает целое число – количество предварительно извлекаемых строк по умолчанию.

getDescriptor(java.lang.String *sql_name*)

Возвращает объект java.lang.Object, имеющий SQL-тип *sql_name*.

getExplicitCachingEnabled()

Возвращает логическое значение, указывающее текущее состояние явного кэша.

getHoldability()

Возвращает целое число, указывающее статус удержания курсора.

getImplicitCachingEnabled()

Возвращает логическое значение, указывающее текущее состояние неявного кэша.

getIncludeSynonyms()

Возвращает логическое значение, указывающее, следует ли включать информацию о синонимах.

getJavaObject(java.lang.String *sql_name*)

Возвращает объект java.lang.Object для *sql_name*.

getMetaData()

Возвращает объект java.sql.DatabaseMetaData.

getProperties()

Возвращает объект java.util.Properties.

getRemarksReporting()

Возвращает логическое значение, указывающее, следует ли возвращать комментарии в составе метаданных.

getRestrictGetTables()

Возвращает логическое значение, указывающее текущее состояние ограничения.

getSessionTimeZone()

Возвращает значение часового пояса сеанса в виде java.lang.String.

getSQLType(java.lang.Object *obj*)

Возвращает тип объекта *obj* в java.lang.String.

getStatementCacheSize()

Возвращает целое число, указывающее размер кэша команд.

getStatementWithKey(java.lang.String *key*)

Возвращает команду java.sql.PreparedStatement с ключом *key* из явного кэша.

getStmtCacheSize()

Возвращает целое число, указывающее размер кэша команд.

getStructAttrCsId()

Возвращает короткое целое, идентифицирующее в Oracle набор символов для атрибутов STRUCT.

getSynchronousMode()

Возвращает логическое значение, указывающее, включен ли синхронный режим.

getTransactionIsolation()

Возвращает целое число – уровень изоляции транзакции.

getTypeMap()

Возвращает объект java.util.Map.

getUserName()

Возвращает имя пользователя в java.lang.String.

getUsingXAFlag()

Возвращает логическое значение, указывающее статус флага UsingXA.

getWarnings()

Возвращает объект java.sql.SQLWarning.

getXAErrorFlag()

Возвращает логическое значение, указывающее статус флага XAError.

isClosed()

Возвращает логическое значение, указывающее статус соединения.

isLogicalConnection()

Возвращает логическое значение, указывающее, является ли соединение логическим.

isReadOnly()

Возвращает логическое значение, указывающее, доступно ли соединение только для чтения.

nativeSQL(java.lang.String *sql*)

Возвращает *sql* во внутреннем формате как java.lang.String.

openJoltConnection(java.lang.String *apiName*, short *major*, short *minor*)

Возвращает объект java.lang.Object. Исключен из числа рекомендуемых к применению в Oracle9*i*.

oracleReleaseSavepoint(OracleSavepoint *savepoint*)

Освобождает точку сохранения *savepoint*.

oracleRollback(OracleSavepoint *savepoint*)

Откатывает операции до точки сохранения *savepoint*.

oracleSetSavepoint()

Создает и возвращает неименованную точку сохранения OracleSavepoint.

oracleSetSavepoint(java.lang.String *name*)

Создает и возвращает точку сохранения OracleSavepoint с именем *name*.

pingDatabase(int *timeOut*)

Возвращает целое число – признак доступности БД.

prepareCall(java.lang.String *sql*)

Возвращает объект java.sql.CallableStatement.

prepareCall(java.lang.String *sql*, int *resultSetType*, int *resultSetConcurrency*)

Возвращает объект java.sql.CallableStatement.

prepareCall(java.lang.String *sql*, int *resultSetType*, int *resultSetConcurrency*,int *resultSetHoldability*)

Возвращает объект java.sql.CallableStatement.

prepareCallWithKey(java.lang.String *key*)

Возвращает объект java.sql.CallableStatement.

prepareStatement(java.lang.String *sql*)

Возвращает объект java.sql.PreparedStatement.

prepareStatement(java.lang.String *sql*, int *autoGeneratedKeys*)

Возвращает объект java.sql.PreparedStatement.

prepareStatement(java.lang.String *sql*, int[] *columnIndexes*)

Возвращает объект java.sql.PreparedStatement.

prepareStatement(java.lang.String *sql*, int *resultSetType*, int *resultSetConcurrency*)

Возвращает объект java.sql.PreparedStatement.

prepareStatement(java.lang.String *sql*, int resultSetType, int *resultSetConcurrency*, int *resultSetHoldability*)

Возвращает объект java.sql.PreparedStatement.

prepareStatement(java.lang.String *sql*, java.lang.String[] *columnNames*)

Возвращает объект java.sql.PreparedStatement.

prepareStatementWithKey(java.lang.String *key*)

Возвращает объект java.sql.PreparedStatement.

purgeExplicitCache()

Очищает явный кэш.

purgeImplicitCache()

Очищает неявный кэш.

putDescriptor(java.lang.String *sql_name*, java.lang.Object *desc*)

Сохраняет дексриптор объекта.

registerApiDescription(java.lang.String *apiName*, short *major*, short *minor*, java.lang.String *className*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

registerSQLType(java.lang.String *sql_name*, java.lang.Class *java_class*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

registerSQLType(java.lang.String *sql_name*, java.lang.String *java_class_name*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

registerTAFCallback(OracleOCIFailover *cbk*, java.lang.Object *obj*)

Регистрирует экземпляр функции обратного вызова TAF в контексте *obj*.

releaseSavepoint(java.sql.Savepoint *savepoint*)

Освобождает точку сохранения *savepoint*.

rollback()

Откатывает операции.

rollback(java.sql.Savepoint *savepoint*)

Откатывает операции до точки сохранения *savepoint*.

setAutoClose(boolean *autoClose*)

Устанавливает режим автоматического закрытия в значение *autoClose*; автоматическое закрытие для соединения всегда включено.

setAutoCommit(boolean *autoCommit*)

Устанавливает режим автоматической фиксации в значение *autoCommit*.

setCatalog(java.lang.String *catalog*)

Задает каталог.

setCreateStatementAsRefCursor(boolean *value*)

Устанавливает флаг CreateStatementAsRefCursor.

setDefaultExecuteBatch(int *batch*)

Устанавливает количество операций по умолчанию в групповой операции.

setDefaultRowPrefetch(int *value*)

Устанавливает количество строк по умолчанию для предварительньой выборки.

setExplicitCachingEnabled(boolean *cache*)

Устанавливает флаг ExplicitCachingEnabled.

setHoldability(int *holdability*)

Задает режим удержания курсора.

setImplicitCachingEnabled(boolean *cache*)

Устанавливает флаг ImplicitCachingEnabled.

setIncludeSynonyms(boolean *synonyms*)

Устанавливает флаг IncludeSynonyms.

setReadOnly(boolean *readOnly*)

Устанавливает флаг ReadOnly.

setRemarksReporting(boolean *reportRemarks*)

Устанавливает флаг RemarksReporting.

setRestrictGetTables(boolean *restrict*)

Устанавливает флаг RestrictGetTables.

setSavepoint()

Создает и возвращает объект java.sql.Savepoint.

setSavepoint(java.lang.String *name*)

Создает и возвращает объект java.sql.Savepoint с именем *name*.

setSessionTimeZone(java.lang.String *regionName*)

Устанавливает часовой пояс сеанса SessionTimeZone в *regionName*.

setStatementCacheSize(int *size*)

Устанавливает размер кэша StatementCacheSize в значение *size*.

setStmtCacheSize(int *size*)

Исключен из числа рекомендуемых к применению в Oracle9*i*. Вместо него рекомендован setStatementCacheSize.

setStmtCacheSize(int *size*, boolean *clearMetaData*)

Исключен из числа рекомендуемых к применению в Oracle9*i*. Вместо него рекомендован метод setStatementCacheSize.

setSynchronousMode(boolean *isSynchronous*)

Устанавливает флаг SynchronousMode.

setTransactionIsolation(int *level*)

Устанавливает уровень изоляции транзакции.

setTypeMap(java.util.Map *map*)

Задает карту типов.

setUsingXAFlag(boolean *value*)

Устанавливает флаг UsingXA. Исключен из числа рекомендуемых к применению в Oracle9*i*.

setWrapper(OracleConnection *wrapper*)

Назначает обертку *wrapper* для данного соединения.

setXAErrorFlag(boolean *value*)

Устанавливает флаг XAError. Исключен из числа рекомендуемых к применению в Oracle9*i*.

shutdown(int *mode*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

startup(java.lang.String *startup_str*, int *mode*)

Исключен из числа рекомендуемых к применению в Oracle9*i*.

unwrap()

Возвращает соединение OracleConnection, распакованное на один уровень.

Унаследованные методы

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

OracleDatabaseMetaData

Расширяет класс java.lang.Object и реализует интерфейс JDBC 1.0 DatabaseMetaData как java.sql.DatabaseMetaData.

Конструктор класса выглядит следующим образом:

```
OracleDatabaseMetaData(OracleConnection conn)
```

Методы

allProceduresAreCallable()

Возвращает логическое значение, указывающее, все ли процедуры, возвращаемые методом getProcedures, могут быть вызваны текущим пользователем.

allTablesAreSelectable()

Возвращает логическое значение, указывающее, все ли таблицы, возвращаемые методом getTable(), доступны текущему пользователю для запросов SELECT.

dataDefinitionCausesTransactionCommit()

Возвращает логическое значение, указывающее, вызывает ли команда DDL внутри транзакции ее фиксацию.

dataDefinitionIgnoredInTransactions()

Возвращает логическое значение, указывающее, игнорируется ли команда DDL внутри транзакции.

deletesAreDetected(int *type*)

Возвращает логическое значение. Реализует JDBC 2.0 DatabaseMetaData.deletesAreDetected.

doesMaxRowSizeIncludeBlobs()

Возвращает логическое значение, указывающее, содержит ли строка объекты LONGVARCHAR или LONGVARBINARY.

getAttributes(java.lang.String *catalog*, java.lang.String *schemaPattern*, java.lang.String *typeNamePattern*, java.lang.String *attributeNamePattern*)

Возвращает объект java.sql.ResultSet, содержащий описание указанного атрибута указанного типа для пользовательского типа, который доступен в указанной схеме и каталоге. Для JDBC 3.0. Появился в Oracle9*i*.

getBestRowIdentifier(java.lang.String *catalog*, java.lang.String *schema*, java.lang.String *table*, int *scope*, boolean *nullable*)

Возвращает объект java.sqlResultSet, уникально идентифицирующий строку.

getCatalogs()

Возвращает объект java.sqlResultSet с каталогами.

getCatalogSeparator()

Возвращает объект java.lang.String с разделителем каталогов.

getCatalogTerm()

Возвращает в java.lang.String терм поставщика БД для каталога.

getColumnPrivileges(java.lang.String *catalog*, java.lang.String *schemaPattern*, java.lang.String *tableNamePattern*, java.lang.String *columnNamePattern*)

Возвращает объект java.sql.ResultSet с описанием прав доступа к столбцам таблицы.

getColumns(java.lang.String *catalog*, java.lang.String *schemaPattern*, java.lang.String *tableNamePattern*, java.lang.String *columnNamePattern*)

Возвращает объект java.sql.ResultSet с описанием столбцов таблицы, доступных в каталоге.

getConnection()

Возвращает объект java.sql.Connection. Реализует JDBC 2.0 DatabaseMetaData.getConnection.

getCrossReference(java.lang.String *primaryCatalog*, java.lang.String *primarySchema*, java.lang.String *primaryTable*, java.lang.String *foreignCatalog*, java.lang.String *foreignSchema*, java.lang.String *foreignTable*)

Возвращает объект java.sql.ResultSet с описанием столбцов внешних ключей в таблице *foreignTable*, ссылающихся на столбцы первичных ключей таблицы *primaryTable*.

getDatabaseMajorVersion()

Возвращает целое число – основной номер версии БД. Для JDBC 3.0. Появился в Oracle9*i*.

getDatabaseMinorVersion()

Возвращает целое число – дополнительный номер версии БД. Для JDBC 3.0. Появился в Oracle9*i*.

getDatabaseProductName()

Возвращает java.lang.String с именем продукта БД.

getDatabaseProductVersion()

Возвращает java.lang.String с версией продукта БД.

getDefaultTransactionIsolation()

Возвращает целое число – уровень изоляции транзакций по умолчанию для БД.

getDriverMajorVersion()

Возвращает целое число – основной номер версии драйвера JDBC.

getDriverMajorVersionInfo()

Возвращает целое число – основной номер версии драйвера JDBC. Статический метод.

getDriverMinorVersion()

Возвращает целое число – дополнительный номер версии драйвера JDBC.

getDriverMinorVersionInfo()

Возвращает целое число – дополнительный номер версии драйвера JDBC. Статический метод.

getDriverName()

Возвращает java.lang.String с именем данного драйвера JDBC.

getDriverNameInfo()

Возвращает java.lang.String с именем данного драйвера JDBC. Статический метод.

getDriverVersion()

Возвращает java.lang.String с версией данного драйвера JDBC.

getDriverVersionInfo()

Возвращает java.lang.String с версией данного драйвера JDBC. Статический метод.

getExportedKeys(java.lang.String *catalog*, java.lang.String *schema*, java.lang.String *table*)

Возвращает объект java.sql.ResultSet с описанием столбцов внешних ключей, ссылающихся на столбцы первичных ключей таблицы.

getExtraNameCharacters()

Возвращает в java.lang.String все «дополнительные» символы, которые могут употребляться в именах идентификаторов без заключения в кавычки.

getIdentifierQuoteString()

Возвращает в java.lang.String символы, выступающие в качестве кавычек для идентификаторов SQL, или пробел, если заключение идентификаторов в кавычки не поддерживается.

getImportedKeys(java.lang.String *catalog*, java.lang.String *schema*, java.lang.String *table*)

Возвращает объект java.sql.ResultSet с описанием столбцов первичных ключей, на которые ссылаются столбцы внешних ключей таблицы.

getIndexInfo(java.lang.String *catalog*, java.lang.String *schema*, java.lang.String *table*, boolean *unique*, boolean *approximate*)

Возвращает объект java.sql.ResultSet с описанием индексов таблицы и статистик.

getJDBCMajorVersion()

Возвращает целое число – основной номер версии JDBC данного драйвера. Для JDBC 3.0. Появился в Oracle9*i*.

getJDBCMinorVersion()

Возвращает целое число – дополнительный номер версии JDBC данного драйвера. Для JDBC 3.0. Появился в Oracle9*i*.

getLobPrecision()

Возвращает java.lang.String. Статический метод.

getMaxBinaryLiteralLength()

Возвращает целое число – максимально допустимое количество шестнадцатеричных символов во встроенном двоичном литерале.

getMaxCatalogNameLength()

Возвращает целое число – максимальную длину имени каталога.

getMaxCharLiteralLength()

Возвращает целое число – максимальную длину символьного литерала.

getMaxColumnNameLength()

Возвращает целое число – ограничение на длину имени столбца.

getMaxColumnsInGroupBy()

Возвращает целое число – максимальное количество столбцов в инструкции GROUP BY.

getMaxColumnsInIndex()

Возвращает целое число – максимальное количество столбцов индекса.

getMaxColumnsInOrderBy()

Возвращает целое число – максимальное количество столбцов в инструкции ORDER BY.

getMaxColumnsInSelect()

Возвращает целое число – максимальное количество столбцов в списке выборки.

getMaxColumnsInTable()

Возвращает целое число – максимальное количество столбцов в таблице.

getMaxConnections()

Возвращает целое число – максимальное количество активных соединений с данной БД.

getMaxCursorNameLength()

Возвращает целое число – максимальную длину имени курсора.

getMaxIndexLength()

Возвращает целое число – максимальную длину индекса в байтах.

getMaxProcedureNameLength()

Возвращает целое число – максимальную длину имени процедуры.

getMaxRowSize()

Возвращает целое число – максимальную длину отдельной строки.

getMaxSchemaNameLength()

Возвращает целое число – максимальную длину имени схемы.

getMaxStatementLength()

Возвращает целое число – максимальную длину команды SQL.

getMaxStatements()

Возвращает целое число – максимальное количество активных команд, одновременно открытых для БД.

getMaxTableNameLength()

Возвращает целое число – максимальную длину имени таблицы.

getMaxTablesInSelect()

Возвращает целое число – максимальное количество таблиц в команде SELECT.

getMaxUserNameLength()

Возвращает целое число – максимальную длину имени пользователя.

getNumericFunctions()

Возвращает объект java.lang.String, содержащий список имен арифметических функций, разделенных запятыми.

getPrimaryKeys(java.lang.String *catalog*, java.lang.String *schema*, java.lang.String *table*)

Возвращает объект java.sql.ResultSet с описанием столбцов первичных ключей таблицы.

getProcedureColumns(java.lang.String *catalog*, java.lang.String *schemaPattern*, java.lang.String *procedureNamePattern*, java.lang.String *columnNamePattern*)

Возвращает объект java.sql.ResultSet с описанием параметров хранимых процедур каталога и столбцов результатов.

getProcedures(java.lang.String *catalog*, java.lang.String *schemaPattern*, java.lang.String *procedureNamePattern*)

Возвращает объект java.sql.ResultSet с описанием хранимых процедур, доступных в каталоге.

getProcedureTerm()

Возвращает в java.lang.String терм поставщика БД для процедуры.

getResultSetHoldability()

Возвращает целое число – режим удержания курсора по умолчанию для данного объекта ResultSet. Для JDBC 3.0.

getSchemas()

Возвращает объект java.sql.ResultSet с именами схем, доступных в данной БД.

getSchemaTerm()

Возвращает в java.lang.String терм поставщика БД для схемы.

getSearchStringEscape()

Возвращает в java.lang.String строку, которая может использоваться для экранирования символов «_» и «% » в параметрах поиска по каталогу.

getSQLKeywords()

Возвращает в java.lang.String полный список всех ключевых слов SQL (разделенных запятыми), не являющихся ключевыми словами SQL92.

getSQLStateType()

Возвращает целое число, указывающее, каким из стандартов определяются состояния SQLSTATE, возвращаемые методом SQLException.getSQLState: X/Open, Open Group SQL CLI или SQL99. Для JDBC 3.0. Появился в Oracle9*i*.

getStringFunctions()

Возвращает объект java.lang.String, содержащий список имен строковых функций, разделенных запятыми.

getSuperTables(java.lang.String *catalog*, java.lang.String *schemaPattern*,java.lang.String *tableNamePattern*)

Возвращает объект java.sql.ResultSet с описанием иерархий таблиц, определенных для некоторой базы данных. Для JDBC 3.0. Появился в Oracle9*i*.

getSuperTypes(java.lang.String *catalog*, java.lang.String *schemaPattern*, java.lang.String *typeNamePattern*)

Возвращает объект java.sql.ResultSet с описанием иерархий пользовательских типов, определенных в некоторой схеме данной БД. Для JDBC 3.0. Появился в Oracle9*i*.

getSystemFunctions()

Возвращает объект java.lang.String, содержащий список имен системных функций, разделенных запятыми.

getTablePrivileges(java.lang.String *catalog*, java.lang.String *schemaPattern*, java.lang.String *tableNamePattern*)

Возвращает объект java.sql.ResultSet с описанием прав доступа для каждой таблицы каталога.

getTables(java.lang.String *catalog*, java.lang.String *schemaPattern*, java.lang.String *tableNamePattern*, java.lang.String[] *types*)

Возвращает объект java.sql.ResultSet с описанием таблиц каталога.

getTableTypes()

Возвращает объект java.sql.ResultSet с типами таблиц, доступных в базе данных.

getTimeDateFunctions()

Возвращает в java.lang.String список имен функций для работы с датой/временем, разделенных запятыми.

getTypeInfo()

Возвращает объект java.sql.ResultSet с описанием всех стандартных типов SQL, которые поддерживаются данной БД.

getUDTs(java.lang.String *catalog*, java.lang.String *schemaPattern*, java.lang.String *typeNamePattern*, int[] *types*)

Возвращает объект java.sql.ResultSet. Реализует JDBC 2.0 DatabaseMetaData::getUDTs.

getURL()

Возвращает в java.lang.String адрес URL данной БД.

getUserName()

Возвращает в java.lang.String имя текущего пользователя в том виде, в котором оно известно БД.

getVersionColumns(java.lang.String *catalog*, java.lang.String *schema*, java.lang.String *table*)

Возвращает объект java.sql.ResultSet с описанием столбцов таблицы, которые автоматически обновляются при обновление любого значения в строке.

insertsAreDetected(int *type*)

Возвращает логическое значение, указывающее может ли БД обнаружить вставку посредством вызова ResultSet.rowInserted. Реализует JDBC 2.0 DatabaseMetaData.insertsAreDetected.

isCatalogAtStart()

Возвращает логическое значение, указывающее, появляется ли имя каталога в начале уточненного имени таблицы.

isReadOnly()

Возвращает логическое значение, указывающее, находится ли база данных в режиме доступа только для чтения.

nullPlusNonNullIsNull()

Возвращает логическое значение, указывающее, получается ли в результате конкатенации значений NULL и не-NULL значение NULL.

nullsAreSortedAtEnd()

Возвращает логическое значение, указывающее, располагаются ли значения NULL в конце упорядоченного списка вне зависимости от порядка сортировки.

nullsAreSortedAtStart()

Возвращает логическое значение, указывающее, располагаются ли значения NULL в начале упорядоченного списка вне зависимости от порядка сортировки.

nullsAreSortedHigh()

Возвращает логическое значение, которое указывает, считаются ли значения NULL при сортировке наибольшими.

nullsAreSortedLow()

Возвращает логическое значение, которое указывает, считаются ли значения NULL при сортировке наименьшими.

othersDeletesAreVisible(int *type*)

Возвращает логическое значение, реализует JDBC 2.0 DatabaseMetaData.othersDeletesAreVisible.

othersInsertsAreVisible(int type)

Возвращает логическое значение, реализует JDBC 2.0 DatabaseMetaData.othersInsertsAreVisible.

othersUpdatesAreVisible(int *type*)

Возвращает логическое значение, реализует JDBC 2.0 DatabaseMetaData.othersUpdatesAreVisible.

ownDeletesAreVisible(int *type*)

Возвращает логическое значение, реализует JDBC 2.0 DatabaseMetaData.ownDeletesAreVisible.

ownInsertsAreVisible(int *type*)

Возвращает логическое значение, реализует JDBC 2.0 DatabaseMetaData.ownInsertsAreVisible.

ownUpdatesAreVisible(int *type*)

Возвращает логическое значение, реализует JDBC 2.0 DatabaseMetaData.ownUpdatesAreVisible.

storesLowerCaseIdentifiers()

Возвращает логическое значение, указывающее, воспринимает ли база данных идентификаторы SQL, записанные в разных регистрах без кавычек, как нечувствительные к регистру и сохраняет их в нижнем регистре.

storesLowerCaseQuotedIdentifiers()

Возвращает логическое значение, указывающее, воспринимает ли база данных идентификаторы SQL, записанные в разных регистрах в кавычках, как нечувствительные к регистру и сохраняет их в нижнем регистре.

storesMixedCaseIdentifiers()

Возвращает логическое значение, указывающее, воспринимает ли база данных идентификаторы SQL, записанные в разных регистрах без кавычек, как нечувствительные к регистру и сохраняет их в разных регистрах.

storesMixedCaseQuotedIdentifiers()

Возвращает логическое значение, указывающее, воспринимает ли база данных идентификаторы SQL, записанные в разных регистрах в кавычках, как нечувствительные к регистру и сохраняет их в разных регистрах.

storesUpperCaseIdentifiers()

Возвращает логическое значение, указывающее, воспринимает ли база данных идентификаторы SQL, записанные в разных регистрах без кавычек, как нечувствительные к регистру и сохраняет их в верхнем регистре.

storesUpperCaseQuotedIdentifiers()

Возвращает логическое значение, указывающее, воспринимает ли база данных идентификаторы SQL, записанные в разных регистрах в кавычках, как нечувствительные к регистру и сохраняет их в верхнем регистре.

supportsAlterTableWithAddColumn()

Возвращает логическое значение, указывающее, поддерживается ли команда ALTER TABLE с добавлением столбца.

supportsAlterTableWithDropColumn()

Возвращает логическое значение, указывающее, поддерживается ли команда ALTER TABLE с удалением столбца.

supportsANSI92EntryLevelSQL()

Возвращает логическое значение, указывающее, поддерживается ли для SQL грамматика ANSI92 начального уровня.

supportsANSI92FullSQL()

Возвращает логическое значение, указывающее, поддерживается ли для SQL полная грамматика ANSI92.

supportsANSI92IntermediateSQL()

Возвращает логическое значение, указывающее, поддерживается ли для SQL промежуточная грамматика ANSI92.

supportsBatchUpdates()

Возвращает логическое значение, указывающее, поддерживаются ли пакетные обновления. Реализует JDBC 2.0 DatabaseMetaData::supportBatchUpdates.

supportsCatalogsInDataManipulation()

Возвращает логическое значение, определяющее, можно ли указывать имя каталога в командах манипулирования данными.

supportsCatalogsInIndexDefinitions()

Возвращает логическое значение, указывающее, можно ли задавать имя каталога в командах определения индексов.

supportsCatalogsInPrivilegeDefinitions()

Возвращает логическое значение, указывающее, можно ли задавать имя каталога в командах определения привилегий.

supportsCatalogsInProcedureCalls()

Возвращает логическое значение, указывающее, можно ли задавать имя каталога в командах вызова процедур.

supportsCatalogsInTableDefinitions()

Возвращает логическое значение, указывающее, можно ли задавать имя каталога в командах определения таблиц.

supportsColumnAliasing()

Возвращает логическое значение, указывающее, поддерживаются ли псевдонимы столбцов.

supportsConvert()

Возвращает логическое значение, указывающее, поддерживается ли функция CONVERT для типов SQL.

supportsConvert(int *fromType*, int *toType*)

Возвращает логическое значение, указывающее, поддерживается ли функция CONVERT для указанных типов SQL.

supportsCoreSQLGrammar()

Возвращает логическое значение, указывающее, поддерживается ли основная SQL-грамматика ODBC.

supportsCorrelatedSubqueries()

Возвращает логическое значение, указывающее, поддерживаются ли связанные подзапросы.

supportsDataDefinitionAndDataManipulationTransactions()

Возвращает логическое значение, указывающее, поддерживаются ли в рамках транзакции команды определения данных и команды манипулирования данными.

supportsDataManipulationTransactionsOnly()

Возвращает логическое значение, указывающее, поддерживаются ли внутри транзакции только команды манипулирования данными.

supportsDifferentTableCorrelationNames()

Возвращает логическое значение, указывающее, поддерживаются ли связанные имена таблиц и должны ли они отличаться от имен таблиц.

supportsExpressionsInOrderBy()

Возвращает логическое значение, указывающее, поддерживаются ли выражения в списках ORDER BY.

supportsExtendedSQLGrammar()

Возвращает логическое значение, указывающее, поддерживается ли расширенная SQL-грамматика ODBC.

supportsFullOuterJoins()

Возвращает логическое значение, указывающее, поддерживаются ли полные вложенные внешние соединения.

supportsGetGeneratedKeys()

Возвращает логическое значение, указывающее, можно ли извлекать автоматически сформированные ключи после выполнения команды.

supportsGroupBy()

Возвращает логическое значение, указывающее, поддерживается ли какая-либо форма инструкции GROUP BY.

supportsGroupByBeyondSelect()

Возвращает логическое значение, указывающее, использует ли инструкция GROUP BY дополнительные столбцы, не входящие в список SELECT, при условии, что она содержит все столбцы из списка SELECT.

supportsGroupByUnrelated()

Возвращает логическое значение, указывающее, использует ли инструкция GROUP BY столбцы, не входящие в список SELECT.

supportsIntegrityEnhancementFacility()

Возвращает логическое значение, указывающее, поддерживается ли свойство SQL Integrity Enhancement Facility.

supportsLikeEscapeClause()

Возвращает логическое значение, указывающее, поддерживается ли символ экранирования в инструкциях LIKE.

supportsLimitedOuterJoins()

Возвращает логическое значение, указывающее, реализована ли ограниченная поддержка внешних соединений.

supportsMinimumSQLGrammar()

Возвращает логическое значение, указывающее, поддерживается ли минимальная SQL-грамматика ODBC.

supportsMixedCaseIdentifiers()

Возвращает логическое значение, указывающее, воспринимает ли база данных идентификаторы SQL, записанные в разных регистрах без кавычек, как чувствительные к регистру и сохраняет ли их в разных регистрах.

supportsMixedCaseQuotedIdentifiers()

Возвращает логическое значение, указывающее, воспринимает ли база данных идентификаторы SQL, записанные в разных регистрах в кавычках, как чувствительные к регистру и сохраняет ли их в разных регистрах.

supportsMultipleOpenResults()

Возвращает логическое значение, указывающее, поддерживается ли одновременно несколько объектов ResultSets для одного объекта CallableStatement. Для JDBC 3.0. Появился в Oracle9*i*.

supportsMultipleResultSets()

Возвращает логическое значение, указывающее, поддерживается ли одновременно несколько объектов ResultSets для одного исполнения.

supportsMultipleTransactions()

Возвращает логическое значение, указывающее, может ли одновременно быть открыто несколько транзакций.

supportsNamedParameters()

Возвращает логическое значение, указывающее, поддерживает ли база данных именованные параметры для вызываемых команд. Для JDBC 3.0. Появился в Oracle9*i*.

supportsNonNullableColumns()

Возвращает логическое значение, указывающее, можно ли определять столбцы как не допускающие NULL.

supportsOpenCursorsAcrossCommit()

Возвращает логическое значение, указывающее, остаются ли курсоры открытыми при выполнении фиксаций.

supportsOpenCursorsAcrossRollback()

Возвращает логическое значение, указывающее, остаются ли курсоры открытыми при выполнении откатов.

supportsOpenStatementsAcrossCommit()

Возвращает логическое значение, указывающее, остаются ли команды открытыми при выполнении фиксаций.

supportsOpenStatementsAcrossRollback()

Возвращает логическое значение, указывающее, остаются ли команды открытыми при выполнении откатов.

supportsOrderByUnrelated()

Возвращает логическое значение, указывающее, использует ли инструкция ORDER BY столбцы, которые отсутствуют в списке SELECT.

supportsOuterJoins()

Возвращает логическое значение, указывающее, поддерживается ли какая-либо форма внешнего соединения.

supportsPositionedDelete()

Возвращает логическое значение, указывающее, поддерживается ли позиционная команда DELETE.

supportsPositionedUpdate()

Возвращает логическое значение, указывающее, поддерживается ли позиционная команда UPDATE.

supportsResultSetConcurrency(int *type*, int *concurrency*)

Возвращает логическое значение, указывающее, поддерживается ли параллелизм указанного типа. Реализует JDBC 2.0 DatabaseMetaData.supportsResultSetConcurrency.

supportsResultSetHoldability(int *holdability*)

Возвращает логическое значение, указывающее, поддерживается ли указанный тип удержания курсора. Для JDBC 3.0. Появился в Oracle9*i*.

supportsResultSetType(int *type*)

Возвращает логическое значение, указывающее, поддерживаются ли результирующие множества указанного типа. Реализует JDBC 2.0 DatabaseMetaData.supportsResultSetType.

supportsSavepoints()

Возвращает логическое значение, указывающее, поддерживаются ли точки сохранения. Для JDBC 3.0. Появился в Oracle9*i*.

supportsSchemasInDataManipulation()

Возвращает логическое значение, указывающее, разрешено ли употребление имени схемы в команде манипулирования данными.

supportsSchemasInIndexDefinitions()

Возвращает логическое значение, указывающее, разрешено ли употребление имени схемы в команде определения индекса.

supportsSchemasInPrivilegeDefinitions()

Возвращает логическое значение, указывающее, разрешено ли употребление имени схемы в команде определения привилегии.

supportsSchemasInProcedureCalls()

Возвращает логическое значение, указывающее, разрешено ли употребление имени схемы в команде вызова процедуры.

supportsSchemasInTableDefinitions()

Возвращает логическое значение, указывающее, разрешено ли употребление имени схемы в команде определения таблицы.

supportsSelectForUpdate()

Возвращает логическое значение, указывающее, поддерживается ли SELECT for UPDATE.

supportsStoredProcedures()

Возвращает логическое значение, указывающее, поддерживаются ли вызовы процедур, использующие синтаксис с экранированием.

supportsSubqueriesInComparisons()

Возвращает логическое значение, указывающее, поддерживаются ли подзапросы в выражениях сравнения.

supportsSubqueriesInExists()

Возвращает логическое значение, указывающее, поддерживаются ли подзапросы в выражениях EXISTS.

supportsSubqueriesInIns()

Возвращает логическое значение, указывающее, поддерживаются ли подзапросы в выражениях IN.

supportsSubqueriesInQuantifieds()

Возвращает логическое значение, указывающее, поддерживаются ли подзапросы в квантифицируемых выражениях.

supportsTableCorrelationNames()

Возвращает логическое значение, указывающее, поддерживаются ли связанные имена таблиц.

supportsTransactionIsolationLevel(int *level*)

Возвращает логическое значение, указывающее, поддерживает ли БД указанный уровень изоляции транзакций.

supportsTransactions()

Возвращает логическое значение, указывающее, поддерживаются ли транзакции.

supportsUnion()

Возвращает логическое значение, указывающее, поддерживается ли инструкция SQL UNION.

supportsUnionAll()

Возвращает логическое значение, указывающее, поддерживается ли инструкция SQL UNION ALL.

updatesAreDetected(int type)

Возвращает логическое значение, указывающее, может ли база данных обнаружить изменение посредством вызова ResultSet.rowUpdated. Реализует JDBC 2.0 DatabaseMetaData.updatesAreDetected.

usesLocalFilePerTable()

Возвращает логическое значение, указывающее, использует ли БД файл для каждой таблицы.

usesLocalFiles()

Возвращает логическое значение, указывающее, сохраняет ли БД таблицы в локальном файле.

Унаследованные методы

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

OracleDriver

Расширяет класс oracle.jdbc.driver.OracleDriver и реализует интерфейс java.sql.Driver. Конструктор класса выглядит следующим образом:

```
OracleDriver()
```

Унаследованные поля

Следующие поля унаследованы от класса oracle.jdbc.driver.OracleDriver:

at_sign_character
database_string
dll_string
dms_parent_name_string
dms_parent_type_string
logon_as_internal_str
password_string
process_escapes_string
protocol_string
slash_character
user_string

Унаследованные методы

Наследует все методы от oracle.jdbc.driver.OracleDriver.

Наследует все методы от java.lang.Object, за исключением clone и finalize.

OracleTypes

Расширяет класс java.lang.Object.

Конструктор класса выглядит следующим образом:

```
OracleTypes()
```

Поля

ARRAY	Статическое целое
BFILE	Статическое целое
BIGINT	Статическое целое
BINARY	Статическое целое
BIT	Статическое целое
BLOB	Статическое целое

BOOLEAN	Статическое целое, появилось в Oracle9*i*
CHAR	Статическое целое
CLOB	Статическое целое
CURSOR	Статическое целое
DATALINK	Статическое целое
DATE	Статическое целое
DECIMAL	Статическое целое
DOUBLE	Статическое целое
FIXED_CHAR	Статическое целое
FLOAT	Статическое целое
INTEGER	Статическое целое
INTERVALDS	Статическое целое, появилось в Oracle9*i*
INTERVALYM	Статическое целое, появилось в Oracle9*i*
JAVA_OBJECT	Статическое целое, появилось в Oracle9*i*
JAVA_STRUCT	Статическое целое, появилось в Oracle9*i*
LONGVARBINARY	Статическое целое
LONGVARCHAR	Статическое целое
NULL	Статическое целое
NUMBER	Статическое целое
NUMERIC	Статическое целое
OPAQUE	Статическое целое, появилось в Oracle9*i*
OTHER	Статическое целое, указывает на то, что SQL-тип является специфичным для БД и отображается на объект Java, к которому можно обратиться при помощи методов getObject и setObject
PLSQL_INDEX_TABLE	Статическое целое
RAW	Статическое целое
REAL	Статическое целое
REF	Статическое целое
ROWID	Статическое целое
SMALLINT	Статическое целое
STRUCT	Статическое целое
TIME	Статическое целое
TIMESTAMP	Статическое целое
TIMESTAMPLTZ	Статическое целое, появилось в Oracle9*i*
TIMESTAMPNS	Статическое целое, появилось в Oracle9*i*, не рекомендуется для использования начиная с версии Oracle 9.2.0
TIMESTAMPTZ	Статическое целое, появилось в Oracle9*i*
TINYINT	Статическое целое
VARBINARY	Статическое целое
VARCHAR	Статическое целое

Унаследованные методы

Наследует все методы от java.lang.Object, за исключением clone и finalize.

---

oracle.sql

Этот пакет предоставляет классы-обертки Java для необработанных данных SQL, позволяющие работать с данными без потери информации. Классы для структурированных типов SQL, таких как объекты и массивы, также содержат дополнительную информацию о структуре и ее преобразовании в другие формы. Все эти классы расширяют oracle.sql.Datum.

Пакет oracle.sql содержит четыре Oracle-интерфейса в Oracle9*i* и три – в Oracle8*i*.

Интерфейсы

В пакет oracle.sql входят следующие интерфейсы.

CustomDatum

Применяется для получения объекта Datum. Начиная с Oracle9*i* не рекомендуется к применению и заменен на ORAData.

Метод

toDatum(OracleConnection *c*)

Возвращает объект Datum, будучи вызванным setOracleObject.

CustomDatumFactory

Создает объект Object из SQL-типа. Начиная с Oracle9*i* не рекомендуется к применению и заменен на ORADataFactory.

Метод

create(Datum *d*, int *sqlType*)

Создает из *sqlType* объект и возвращает его в виде CustomDatum. Datum инициализируется данными *d*.

Mutable

Применяется для перемещения одного типа объекта в другой.Только для Oracle8*i*.

Метод

copy(CustomDatum *cd*)

Копирует содержимое *cd* в объект.

ORAData

Интерфейс для настраиваемых пользовательских типов.Заменил интерфейс CustomDatum, исключенный из числа рекомендуемых к применению в Oracle9*i*.

Метод

toDatum(java.sql.Connection *c*)

Извлекает объект oracle.sql.Datum из соединения *c*.

ORADataFactory

Интерфейс для фабрики классов ORAData. Заменил интерфейс CustomDatumFactory, исключенный из числа рекомендуемых к применению в Oracle9*i*.

Метод

create(Datum *d*, int *sqlType*)

Создает из *sqlType* объект и возвращает его в виде CustomDatum. Datum инициализируется данными *d*.

Классы

В пакет oracle.sql входят следующие классы.

ARRAY

Расширяет класс oracle.sql.DatumWithConnection. Реализует java.sql.Array.

Класс имеет такой конструктор:

```
ARRAY(ArrayDescriptor type, java.sql.Connection conn,
java.lang.Object elements)
```

Поля

ACCESS_FORWARD	Статическое целое
ACCESS_REVERSE	Статическое целое
ACCESS_UNKNOWN	Статическое целое

Методы

getAccessDirection()

Возвращает целое число – текущее состояние автоиндексации.

getArray()

Возвращает содержимое SQL-массива, определяемого объектом, в виде java.lang.Object.

getArray(java.util.Map *map*)

Возвращает содержимое SQL-массива, определяемого объектом, в виде java.lang.Object.

getArray(long *index*, int *count*)

Возвращает в виде java.lang.Object массив, содержащий часть SQL-массива, которая начинается с указанного индекса *index* и содержит не более *count* последовательных элементов массива SQL.

getArray(long *index*, int *count*, java.util.Map *map*)

Возвращает в виде java.lang.Object массив, содержащий часть SQL-массива, которая начинается с указанного индекса и содержит не более *count* последовательных элементов массива SQL.

getAutoBuffering()

Возвращает логическое значение.

getAutoIndexing()

Возвращает логическое значение.

getBaseType()

Возвращает целое число – код типа элементов массива (из java.sql.Types).

getBaseTypeName()

Возвращает в java.lang.String имя SQL-типа элементов массива, который определяется данным объектом Array.

getConnection()

Возвращает OracleConnection.

getDescriptor()

Возвращает ArrayDescriptor.

getDoubleArray()

Возвращает массив double[].

getDoubleArray(long *index*, int *count*)

Возвращает массив double[], начиная с индекса *index* длиной *count* записей.

getFloatArray()

Возвращает массив float[]

getFloatArray(long *index*, int *count*)

Возвращает массив float[], начиная с индекса *index* длиной *count* записей.

getIntArray()

Возвращает массив int[].

getIntArray(long *index*, int *count*)

Возвращает массив int[], начиная с индекса *index* длиной *count* записей.

getJavaSqlConnection()

Возвращает java.sql.Connection.

getLongArray()

Возвращает массив long[].

getLongArray(long *index*, int *count*)

Возвращает массив long[], начиная с индекса *index* длиной *count* записей.

getOracleArray()

Возвращает массив Datum[].

getOracleArray(long *index*, int *count*)

Возвращает массив Datum[], начиная с индекса *index* длиною *count* записей.

getResultSet()

Возвращает объект java.sql.ResultSet, содержащий элементы массива.

getResultSet(java.util.Map *map*)

Возвращает объект java.sql.ResultSet, содержащий элементы массива, который определяется данным объектом Array и использует карту *map* для отображения элементов массива.

getResultSet(long *index*, int *count*)

Возвращает объект java.sql.ResultSet, содержащий элементы подмассива, который начинается с указанного индекса и содержит не более *count* последовательных элементов.

getResultSet(long *index*, int *count*, java.util.Map *map*)

Возвращает объект java.sql.ResultSet, содержащий элементы подмассива, который начинается с указанного индекса и содержит не более *count* последовательных элементов.

getShortArray()

Возвращает массив short[].

getShortArray(long *index*, int *count*)

Возвращает массив int[], начиная с индекса *index* длиной *count* записей.

getSQLTypeName()

Возвращает имя типа в java.lang.String.

isConvertibleTo(java.lang.Class *jClass*)

Возвращает логическое значение.

length()

Возвращает целое число – размер массива.

setAutoBuffering(boolean *enable*)

Устанавливает флаг AutoBuffering в *enable*.

setAutoIndexing(boolean *enable*)

Устанавливает флаг AutoIndexing в *enable*.

setAutoIndexing(boolean *enable*, int *direction*)

Устанавливает флаг AutoIndexing в *enable* для направления *direction*.

toJdbc()

Возвращает объект java.lang.Object в виде JDBC-представления объекта Datum.

Унаследованные методы

Следующие методы унаследованы от oracle.sql.DatumWithConnection в версии Oracle9*i*:

assertNotNull

getOracleConnection

Наследует все методы от oracle.sql.Datum, кроме isConvertibleTo и toJdbc.

Также наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

ArrayDescriptor

Расширяет класс TypeDescriptor. Реализует java.io.Serializable.

Поля

CACHE_ALL
CACHE_LAST
CACHE_NONE
TYPE_NESTED_TABLE
TYPE_VARRAY

Унаследованные поля

От класса oracle.sql.TypeDescriptor унаследовано следующее поле:

DEBUG_SERIALIZATION

Методы

createDescriptor(java.lang.String *name*, java.sql.Connection *conn*)

Фабрика класса Descriptor. Возвращает ArrayDescriptor для типа *name*. Статический метод.

descType()

Возвращает в java.lang.String описание типа коллекции.

getArrayType()

Возвращает целое число – тип массива.

getBaseName()

Возвращает java.lang.String с полным уточненным именем типа, если элементы являются именованными типами, или возвращает имя типа, которое используется базой данных.

getBaseType()

Возвращает целое число – код типа элемента.

getMaxLength()

Возвращает длинное целое – максимальное количество элементов, которое может хранить объект Array.

getTypeCode()

Возвращает целое число. Появился в Oracle9*i*.

toResultSet(ARRAY *array*, long *index*, int *count*, java.util.Map *map*, boolean *saveLocalCopy*)

Возвращает объект java.sql.ResultSet.

toResultSetFromImage(ARRAY *array*, long *index*, int *count*, java.util.Map *map*)

Возвращает объект java.sql.ResultSet.

Унаследованные методы

Наследует все методы от oracle.sql.TypeDescriptor. Также наследует все методы от java.lang.Object, за исключением clone, equals и finalize; в версии Oracle8*i* также наследует equals.

BFILE

Расширяет oracle.sql.DatumWithConnection в версии Oracle9*i*, oracle.sql.Datum – в Oracle8*i*.

Поля

MODE_READONLY	Статическое целое. Появилось в Oracle 9*i*.
MODE_READWRITE	Статическое целое. Появилось в Oracle 9*i*.

Методы

asciiStreamValue()

Возвращает java.io.InputStream, который является представлением ASCII-потока для объекта Datum.

close()

Закрывает ранее открытый внешний объект LOB. Появился в Oracle9*i*.

closeFile()

Закрывает файл.

fileExists()

Возвращает логическое значение, определяющее, указывает ли указатель данного BFILE на файл, реально существующий в файловой системе сервера.

getBinaryStream()

Возвращает java.io.InputStream для всего BFILE.

getBinaryStream(long *pos*)

Возвращает java.io.InputStream в виде потока, начинающегося с позиции *pos*. Появился в Oracle9*i*.

getBytes(long *pos*, int *length*)

Возвращает массив byte[] как копию содержимого BFILE, начиная с позиции *pos* длиной *length*.

getBytes(long *pos*, int *length*, byte[] *buf*)

Возвращает целое число; копирует содержимое BFILE, начиная с позиции *pos* длиной *length* в *buf*.

getConnection()

Возвращает OracleConnection.

getDirAlias()

Возвращает java.lang.String, псевдоним каталога BFILE.

getJavaSqlConnection()

Возвращает java.sql.Connection. Появился в Oracle9*i*.

getName()

Возвращает в java.lang.String имя файла BFILE.

isConvertibleTo(java.lang.Class *jClass*)

Возвращает логическое значение, указывающее, можно ли преобразовать данный объект Data в указанный тип данных Java.

isFileOpen()

Возвращает логическое значение, указывающее, был ли BFILE открыт при помощи данного BFILE.

isOpen()

Возвращает логическое значение, указывающее, открыт ли внешний объект LOB. Появился в Oracle9*i*.

length()

Возвращает длинное целое – размер BFILE в байтах.

open()

Открывает внешний LOB только для чтения. Появился в Oracle9*i*.

open(int *mode*)

Открывает внешний LOB в указанном режиме. Появился в Oracle9*i*.

openFile()

Открывает файл FILE.

position(BFILE *pattern*, long *start*)

Возвращает длинное целое, определяющее байтовую позицию, с которой начинается указанный образец *pattern* (отсчет начинает с позиции *start*).

position(byte[] *pattern*, long *start*)

Возвращает длинное целое, определяющее байтовую позицию, с которой начинается указанный образец *pattern* (отсчет начинает с позиции *start*).

toJdbc()

Возвращает java.lang.Object с типом объекта Java по умолчанию.

Унаследованные методы

Наследует все методы от oracle.sql.TypeDescriptor.

Наследует все методы от oracle.sql.Datum, за исключением asciiStreamValue и toJDBC.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

BLOB

Расширяет oracle.sql.DatumWithConnection в версии Oracle9*i*, oracle.sql.Datum – в версии Oracle8*i*. Реализует java.sql.Blob в Oracle 9*i* и oracle.jdbc2.Blob – в Oracle8*i*.

Поля

DURATION_CALL	Статическое целое. Появилось в Oracle 9*i*
DURATION_SESSION	Статическое целое. Появилось в Oracle 9*i*
MAX_CHUNK_SIZE	Статическое целое
MODE_READONLY	Статическое целое. Появилось в Oracle 9*i*
MODE_READWRITE	Статическое целое. Появилось в Oracle 9*i*

Методы

close()

Закрывает ранее открытый BLOB. Появился в Oracle9*i*.

createTemporary(java.sql.Connection *conn*, Boolean *cache*, int *duration*)

Возвращает временный BLOB. Статический метод. Появился в Oracle9*i*.

empty_lob()

Возвращает BLOB в виде пустого большого объекта. Статический метод.

freeTemporary()

Освобождает содержимое и указатель временного объекта BLOB. Появился в Oracle9*i*.

freeTemporary(BLOB *temp_lob*)

Освобождает содержимое и указатель указанного временного объекта BLOB. Статический метод. Появился в Oracle9*i*.

getBinaryOutputStream()

Возвращает поток java.io.OutputStream.

getBinaryOutputStream(long *pos*)

Возвращает поток java.io.OutputStream, начиная с позиции *pos*. Появился в Oracle9*i*.

getBinaryStream()

Возвращает поток java.io.InputStream в качестве интерфейса BLOB.

getBinaryStream(long *pos*)

Возвращает поток java.io.InputStream, начиная с позиции *pos*. Появился в Oracle9*i*.

getBufferSize()

Возвращает целое число – размер буфера объекта.

getBytes(long *pos*, int *length*)

Возвращает массив byte[] в качестве интерфейса BLOB.

getBytes(long *pos*, int *length*, byte[] *buf*)

Копирует содержимое BLOB, начиная с позиции *pos* длиной *length* в буфер *buf*.

getChunkSize()

Возвращает целое число.

getConnection()

Возвращает OracleConnection.

getJavaSqlConnection()

Возвращает java.sql.Connection. Появился в Oracle9*i*.

isConvertibleTo(java.lang.Class *jClass*)

Возвращает логическое значение, указывающее на возможность преобразования к заданному типу.

isEmptyLob()

Возвращает логическое значение, определяющее, указывает ли указатель LOB на пустой BLOB.

isOpen()

Возвращает логическое значение, указывающее, открыт ли BLOB. Появился в Oracle9*i*.

isTemporary()

Возвращает логическое значение, определяющее, указывает ли указатель LOB на временный BLOB. Появился в Oracle9*i*.

isTemporary(BLOB *lob*)

Возвращает логическое значение, определяющее, указывает ли указатель LOB на данный временный BLOB. Статический метод. Появился в Oracle9*i*.

length()

Возвращает длинное целое – длину BLOB.

open(int *mode*)

Открывает BLOB в указанном режиме.

position(java.sql.Blob *pattern*, long *start*)

Возвращает длинное целое – позицию образца *pattern*, при этом отсчет начинается с позиции *start*. В Oracle8*i* образец имеет тип oracle.jdbc2.Blob.

position(byte[] *pattern*, long *start*)

Возвращает длинное целое – позицию образца, при этом отсчет начинается с позиции *start*.

putBytes(long *pos*, byte[] *bytes*)

Возвращает целое число – количество байтов, записанных в позицию *pos* в BLOB из массива *bytes*.

putBytes(long *pos*, byte[] *bytes*, int *length*)

Возвращает целое число – количество фактически записанных из массива *bytes* длиной *length* байтов в позицию *pos* BLOB.

setBinaryStream(long *pos*)

Возвращает объект java.io.OutputStream, который может использоваться для записи значения в BLOB, представленный данным объектом. Для JDBC 3.0. Появился в Oracle9*i*.

setBytes(long *pos*, byte[] *bytes*)

Возвращает целое число – количество фактически записанных байтов; записывает указанный массив байтов в BLOB, представленный данным объектом, начиная с позиции *pos*. Для JDBC 3.0. Появился в Oracle9*i*.

setBytes(long *pos*, byte[] *bytes*, int *offset*, int *len*)

Возвращает целое число – количество фактически записанных байтов и записывает полностью или частично указанный массив байтов в BLOB, представленный данным объектом, начиная с позиции *pos* длиною *len*. JDBC 3.0. Появился в Oracle9*i* Release 2.

toJdbc()

Возвращает объект java.lang.Object с типом, принятым по умолчанию в Java.

trim(long *newlen*)

Обрезает значение BLOB до размера *newlen*. Появился в Oracle9*i*.

truncate(long *len*)

Усекает значение BLOB до длины *len* байт. Для JDBC 3.0. Появился в Oracle9*i* Release 2.

Унаследованные методы

Следующие методы унаследованы от oracle.sql.DatumWithConnection в Oracle9*i*:

assertNotNull

getOracleConnection

Наследует все методы от oracle.sql.Datum, за исключением isConvertibleTo и toJdbc.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

CHAR

Расширяет oracle.sql.Datum.

Конструктор класса имеет вид:

```
CHAR(java.lang.Object obj, oracle.sql.CharacterSet charSet)
```

для создания объекта типа CHAR из объекта *obj* и

```
CHAR(java.lang.String str, oracle.sql.CharacterSet charSet)
```

для создания CHAR из строки *str*.

Поле

DEFAULT_CHARSET	Статический oracle.sql.CharacterSet

Методы

asciiStreamValue()

Возвращает java.io.InputStream – представление объекта Datum.

bigDecimalValue()

Возвращает объект данных в виде java.math.BigDecimal.

binaryStreamValue()

Возвращает объект данных в виде java.io.InputStream.

booleanValue()

Возвращает объект данных в виде логического значения.

byteValue()

Возвращает объект данных в виде байтового значения.

characterStreamValue()

Возвращает объект данных в виде java.io.Reader.

dateValue()

Возвращает объект данных в виде java.sql.Date.

doubleValue()

Возвращает объект данных в виде значения типа double.

floatValue()

Возвращает объект данных в виде значения типа float.

getString()

Возвращает объект данных в виде java.lang.String.

intValue()

Возвращает объект данных в виде целого числа.

isConvertibleTo(java.lang.Class *jClass*)

Возвращает логическое значение, указывающее, можно ли преобразовать этот объект данных в *jClass*.

longValue()

Возвращает объект данных в виде значения типа long.

stringValue()

Возвращает объект данных в виде java.lang.String.

timestampValue()

Возвращает объект данных в виде java.sql.Timestamp.

timeValue()

Возвращает объект данных в виде java.sql.Time.

toJdbc()

Возвращает объект данных в виде значения с типом, принятым для него в Java по умолчанию.

toString()

Возвращает объект данных в виде java.lang.String.

Унаследованные методы

Следующие методы унаследованы от oracle.sql.Datum:

equals
getBytes
getLength
getStream
makeJdbcArray
setBytes
setShareBytes
shareBytes

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

CharacterSet

Расширяет java.lang.Object. Только для Oracle8*i*.

Поля

Данный класс имеет поле для поддержки каждого набора символов. Подробную информацию можно найти в документации Oracle.

Методы

convert(CharacterSet, byte[], int, int)

Возвращает абстрактный байтовый массив.

convert(java.lang.String)

Возвращает абстрактный байтовый массив.

convertUnshared(CharacterSet, byte[], int, int)

Возвращает байтовый массив.

convertWithReplacement(java.lang.String)

Возвращает абстрактный байтовый массив.

equals(java.lang.Object)

Возвращает логическое значение.

getOracleId()
    Возвращает целое число.
getRatioTo(CharacterSet)
    Возвращает целое число.
hashCode()
    Возвращает целое число.
isConvertibleFrom(CharacterSet)
    Возвращает логическое значение.
isLossyFrom(CharacterSet)
    Возвращает абстрактное логическое значение.
make(int)
    Возвращает CharacterSet. Статический метод.
toString()
    Возвращает java.lang.String.
toString(byte[], int, int)
    Возвращает java.lang.String.
toStringWithReplacement(byte[], int, int)
    Возвращает абстрактную строку java.lang.String.

Унаследованные методы

Перечисленные ниже методы унаследованы от java.lang.Object:

getClass
notify
notifyAll
wait

CLOB

Расширяет класс oracle.sql.DatumWithConnection. Реализует java.sql.Clob в Oracle9*i*, oracle. jdbc2.Clob – в Oracle8*i*.

Конструктор класса имеет следующий вид:

```
CLOB(OracleConnection conn, byte[] lob_descriptor, short csform)
```

Поля

DURATION_CALL	Статическое целое. Появилось в Oracle 9*i*
DURATION_SESSION	Статическое целое. Появилось в Oracle 9*i*
MAX_CHUNK_SIZE	Статическое целое
MODE_READONLY	Статическое целое. Появилось в Oracle 9*i*
MODE_READWRITE	Статическое целое. Появилось в Oracle 9*i*

Методы

close()
    Закрывает ранее открытый CLOB. Появился в Oracle9*i*.

createTemporary(java.sql.Connection *conn*, boolean *cache*, int *duration*)

Возвращает временный BLOB. Статический метод. Появился в Oracle9*i*.

empty_lob()

Возвращает LCOB в виде пустого большого объекта. Статический метод.

freeTemporary()

Освобождает содержимое и указатель временного объекта CLOB. Появился в Oracle9*i*.

freeTemporary(CLOB *temp_lob*)

Освобождает содержимое и указатель указанного временного объекта CLOB. Статический метод. Появился в Oracle9*i*.

getAsciiOutputStream()

Возвращает java.io.OutputStream.

getAsciiOutputStream(long *pos*)

Возвращает поток java.io.OutputStream, начиная с позиции *pos*. Появился в Oracle9*i*.

getAsciiStream()

Возвращает поток java.io.InputStream в качестве интерфейса CLOB.

getAsciiStream(long *pos*)

Возвращает поток java.io.InputStream, начиная с позиции *pos*. Появился в Oracle9*i*.

getBufferSize()

Возвращает целое число – размер буфера.

getCharacterOutputStream()

Возвращает java.io.Writer для записи в объект.

getCharacterOutputStream(long *pos*)

Возвращает поток java.io.Writer для записи в объект, начиная с позиции *pos*. Появился в Oracle9*i*.

getCharacterStream()

Возвращает поток java.io.Reader для чтения объекта.

getCharacterStream(long *pos*)

Возвращает поток java.io.Reader для чтения объекта, начиная с позиции *pos*. Появился в Oracle9*i*.

getChars(long pos, int length, char[] buffer)

Возвращает целое число – количество символов, прочитанных, начиная с позиции *pos* в буфер *buf* в количестве *length*. Появился в Oracle9*i*.

getChunkSize()

Возвращает целое число.

getConnection()

Возвращает OracleConnection.

getJavaSqlConnection()

Возвращает java.sql.Connection. Появился в Oracle9*i*.

getSubString(long *pos*, int *length*)

Возвращает в java.lang.String строку длиной *length*, начиная с позиции *pos*.

isConvertibleTo(java.lang.Class *jClass*)

Возвращает логическое значение, указывающее на возможность преобразования.

isEmptyLob()

Возвращает логическое значение, определяющее, указывает ли указатель на пустой CLOB.

isNCLOB()

Возвращает логическое значение. Появился в Oracle9*i*.

isOpen()

Возвращает логическое значение, указывающее, открыт ли CLOB. Появился в Oracle9*i*.

isTemporary()

Возвращает логическое значение, определяющее, указывает ли указатель на временный CLOB. Появился в Oracle9*i*.

isTemporary(CLOB *lob*)

Возвращает логическое значение, определяющее, является ли данный CLOB временным. Статический метод. Появился в Oracle9*i*.

length()

Возвращает длинное целое – длину CLOB.

open(int *mode*)

Открывает CLOB в указанном режиме.

position(java.lang.String *pattern*, long *start*)

Возвращает длинное целое – позицию образца *pattern*, при этом отсчет начинается с позиции *start*.

position(java.sql.Clob *pattern*, long *start*)

Возвращает длинное целое – позицию образца *pattern*, при этом отсчет начинается с позиции *start*.

putChars(long *pos*, char[] *chars*)

Записывает символы из буфера *chars* в CLOB, начиная с позиции *pos*, и возвращает целое число – количество реально записанных символов.

putChars(long *pos*, char[] *chars*, int *length*)

Записывает *length* символов из буфера *chars* в CLOB, начиная с позиции *pos*, и возвращает целое число – количество реально записанных символов. Появился в Oracle9*i*.

putString(long *pos*, java.lang.String *str*)

Записывает символы из строки *str* в CLOB, начиная с позиции *pos*, и возвращает целое число – количество реально записанных символов.

setAsciiStream(long *pos*)

Возвращает поток java.io.OutputStream, который может использоваться для записи в данный объект CLOB, начиная с позиции *pos*. Для JDBC 3.0. Появился в Oracle9*i*.

setBytes(long *pos*, java.lang.String *str*, int *offset*, int *len*)

Записывает *len* символов из строки *str*, расположенных по смещению *offset* в CLOB, начиная с позиции *pos*, и возвращает целое число – количество реально записанных символов. Для JDBC 3.0. Появился в Oracle9*i* Release 2.

setCharacterStream(long *pos*)

Возвращает java.io.Writer, который может использоваться для записи символов Unicode в данный объект CLOB. Для JDBC 3.0. Появился в Oracle9*i*.

setString(long *pos*, java.lang.String *str*)

Записывает символы из строки *str* в CLOB, начиная с позиции *pos*, и возвращает целое число – количество реально записанных символов. JDBC 3.0. Появился в Oracle9*i*.

setString(long *pos*, java.lang.String *string*, int *offset*, int *length*)

Записывает *len* символов из строки *string*, расположенных по смещению *offset* в CLOB, начиная с позиции *pos*, и возвращает целое число – количество реально записанных символов. Появился в Oracle9*i*.

toJdbc()

Возвращает объект данных в виде значения с типом, принятым для него в Java по умолчанию.

trim(long *newlen*)

Обрезает значение CLOB до размера *newlen*. Появился в Oracle9*i*.

truncate(long *len*)

Усекает значение CLOB до длины *len* байтов. Для JDBC 3.0. Появился в Oracle9*i* Release 2.

Унаследованные методы

Методы, перечисленные ниже, унаследованы от oracle.sql.DatumWithConnection в Oracle9*i*:

assertNotNull

getOracleConnection

Наследует все методы от oracle.sql.Datum, за исключением isConvertibleTo и toJdbc.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

DATE

Расширяет oracle.sql.Datum.

Класс имеет множество конструкторов, позволяющих создавать даты из данных различных типов:

```
DATE()
DATE(byte[] date)
DATE(java.sql.Date date)
DATE(java.sql.Date date, java.util.Calendar cal)
DATE(java.lang.Object obj)
DATE(java.lang.Object obj, java.util.Calendar cal)
DATE(java.lang.String str)
DATE(java.lang.String str, java.util.Calendar cal) - Появился в Oracle9i.
DATE(java.sql.Time time)
DATE(java.sql.Time time, java.util.Calendar cal)
DATE(java.sql.Timestamp timestamp)
DATE(java.sql.Timestamp timestamp, java.util.Calendar cal)
```

Поля

BDA	Статическое целое
BDAL	Статическое целое
BDT	Статическое целое
BHR	Статическое целое
BHRL	Статическое целое
BMN	Статическое целое
BMNL	Статическое целое
BMO	Статическое целое
BMOL	Статическое целое
BSC	Статическое целое
BSCL	Статическое целое
BYR	Статическое целое
BYRL	Статическое целое
HRZERO	Статическое целое
MIZERO	Статическое целое
MSD	Статическое целое
SEZERO	Статическое целое
YR0	Статическое целое

Методы

addJulianDays(int *julianDay*, int *julianSec*)

Возвращает дату DATE, которая наступит через *julianDay* дней и *julianSec* секунд.

addMonths(int *months*)

Возвращает дату DATE, которая наступит через *months* месяцев.

checkValidity(byte[] *date*)

Возвращает целое число, равное 0, если дата корректна, или же комбинацию полей, описанных ранее, для обозначения недействительной составляющей даты. Статический метод.

compareTo(DATE *date*)

Возвращает -1, если DATE меньше *date*, 0 – если даты равны, и 1 – если DATE больше, чем *date*.

dateValue()

Возвращает внутреннюю дату Oracle, преобразованную в java.sql.Date.

dateValue(java.util.Calendar *cal*)

Возвращает внутреннюю дату Oracle и календарь *cal*, преобразованные в java.sql.Date.

diffInJulianDays(DATE *date*, int[] *julianDay*, int[] *julianSec*)

Вычисляет разницу между датами Юлианского календаря в днях.

diffInMonths(DATE *date*)

Возвращает число, равное разнице между двумя датами в месяцах.

fromJulianDays(int *julianDay*, int *julianSec*)

Возвращает дату DATE, полученную преобразованием из Юлианских дней и секунд в тип Oracle DATE. Статический метод.

fromText(java.lang.String *datestr*, java.lang.String *fmt*, java.lang.String *lang*)

Возвращает дату DATE, полученную преобразованием из строки в объект DATE. Статический метод.

getCurrentDate()

Возвращает текущие дату и время в виде DATE. Статический метод.

isConvertibleTo(java.lang.Class *cls*)

Возвращает логическое значение, указывающее, можно ли преобразовать объект к данному классу.

lastDayOfMonth()

Возвращает объект DATE, инициализированный последним днем месяца.

makeJdbcArray(int *arraySize*)

Возвращает данные, представленные как массив JDBC.

numberToJulianDays(NUMBER *num*, int[] *julianDay*, int[] *julianSec*)

Преобразует число *num* типа Oracle NUMBER в Юлианские дни и секунды.

parseFormat(java.lang.String *fmt*, java.lang.String *lang*)

Возвращает в byte[] результат преобразования строки *fmt* в токены для использования методом toText(). Появился в Oracle9*i*.

round(java.lang.String *prec*)

Возвращает DATE с округлением даты до указанной точности *prec*.

setDayOfWeek(int *day*)

Возвращает объект DATE, инициализированный датой, которая на неделю предшествует указанному дню.

stringValue()

Возвращает результат преобразования внутренней даты Oracle в строку Java.

timestampValue()

Возвращает результат преобразования внутренней даты Oracle в Java Timestamp.

timestampValue(java.util.Calendar *cal*)

Возвращает результат преобразования внутренней даты Oracle и календаря *cal* в Java Timestamp.

timeValue()

Возвращает результат преобразования внутренней даты Oracle в Java Time.

timeValue(java.util.Calendar *cal*)

Возвращает результат преобразования внутренней даты Oracle и календаря *cal* в Java Time.

toBytes()

Возвращает массив byte[]. Статический метод.

toBytes(java.sql.Date *date*)

Возвращает в массив byte[] результат преобразования даты *date* в дату Oracle. Статический метод.

toBytes(java.sql.Date *date*, java.util.Calendar *cal*)

Возвращает в массив byte[] результат преобразования даты *date* в дату Oracle для календаря *cal*. Статический метод.

toBytes(java.lang.String *str*)

Возвращает в массив byte[] результат преобразования строки *str* в дату Oracle. Статический метод.

toBytes(java.lang.String *str*, java.util.Calendar *cal*)

Возвращает в массив byte[] результат преобразования строки *str* в дату Oracle для календаря *cal*. Появился в Oracle9*i*. Статический метод.

toBytes(java.sql.Time *time*)

Возвращает в массив byte[] результат преобразования времени *time* в дату Oracle. Статический метод.

toBytes(java.sql.Time *time*, java.util.Calendar *cal*)

Возвращает в массив byte[] результат преобразования времени *time* в дату Oracle для календаря *cal*. Статический метод.

toBytes(java.sql.Timestamp *timestamp*)

Возвращает в массив byte[] результат преобразования метки времени *timestamp* в дату Oracle. Статический метод.

toBytes(java.sql.Timestamp *timestamp*, java.util.Calendar *cal*)

Возвращает в массив byte[] результат преобразования метки времени *timestamp* в дату Oracle для календаря *cal*. Статический метод.

toDate(byte[] *date*)

Возвращает в java.sql.Date результат преобразования даты Oracle *date*. Статический метод.

toDate(byte[] *date*, java.util.Calendar *cal*)

Возвращает в java.sql.Date результат преобразования даты Oracle *date* для календаря *cal*. Статический метод.

toJdbc()

Возвращает в java.lang.Object Java-представление объекта Datum.

toJulianDays(int[] *julianDay*, int[] *julianSec*)

Преобразует указанную дату в Юлианские дни и часы.

toNumber()

Возвращает значение типа NUMBER, преобразуя дату в значение типа Oracle NUMBER.

toString(byte[] *date*)

Возвращает java.lang.String как результат преобразования даты *date*. Статический метод.

toText(byte[] *pfmt*, java.lang.String *lang*)

Возвращает java.lang.String как результат преобразования даты согласно формату *pfmt* на языке *lang*. Появился в Oracle9*i*.

toText(java.lang.String *fmt*, java.lang.String *lang*)

Возвращает java.lang.String как результат преобразования даты согласно формату *fmt* на языке *lang*.

toTime(byte[] *date*)

Возвращает java.sql.Time как результат преобразования даты Oracle DATE. Статический метод.

toTime(byte[] *date*, java.util.Calendar *cal*)

Возвращает java.sql.Time как результат преобразования даты Oracle DATE для календаря *cal*. Статический метод.

toTimestamp(byte[] *date*)

Возвращает java.sql.Timestamp как результат преобразования даты Oracle DATE. Статический метод.

toTimestamp(byte[] *date*, java.util.Calendar *cal*)

Возвращает java.sql.Timestamp как результат преобразования даты Oracle DATE для календаря *cal*. Статический метод.

truncate(java.lang.String *prec*)

Возвращает объект DATE с датой, усеченной с точностью до *prec*.

Унаследованные методы

Следующие методы унаследованы от oracle.sql.Datum:

asciiStreamValue
bigDecimalValue
binaryStreamValue
BooleanValue
byteValue
characterStreamValue
doubleValue
equals
floatValue
getBytes
getLength
getStream
intValue
longValue
setBytes
setShareBytes
shareBytes

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

Datum

Расширяет класс java.sql.Object. Datum является корневым классом для следующих подклассов: CHAR, DATE, DatumWithConnection, INTERVALYM, NUMBER, RAW, ROWID, TIMESTAMP, TIMESTAMPLTZ и TIMESTAMPTZ в Oracle9*i*; в версии Oracle8*i* Datum корневой класс для ARRAY, BFILE, BLOB, CHAR, CLOB, DATE, NUMBER, RAW, REF, ROWID и STRUCT.

Конструкторами класса являются:

```
Datum()
```

который создает пустой объект Datum и

```
Datum(byte[] newData)
```

который создает новый объект с данными *newData*.

Методы

asciiStreamValue()

Возвращает java.io.InputStream, который является представлением ASCII-потока для объекта Datum.

bigDecimalValue()

Возвращает представление объекта Datum в виде java.math.BigDecimal.

binaryStreamValue()

Возвращает представление объекта Datum в виде java.io.InputStream.

booleanValue()

Возвращает логическое представление объекта Datum.

byteValue()

Возвращает байтовое представление объекта Datum.

characterStreamValue()

Возвращает представление объекта Datum в виде символьного потока.

dateValue()

Возвращает представление объекта Datum в виде java.sql.Date.

doubleValue()

Возвращает представление объекта Datum в виде значения типа Double.

equals(java.lang.Object *obj*)

Возвращает логическое значение – результат проверки на равенство с объектом Datum.

floatValue()

Возвращает представление объекта Datum в виде значения типа float.

getBytes()

Возвращает байтовый массив, содержащий копию данных БД.

getLength()

Возвращает длинное целое – длину объекта Datum.

getStream()

Возвращает поток java.io.InputStream, пригодный для чтения необработанных данных.

intValue()

Возвращает представление объекта Datum в виде целого числа.

isConvertibleTo(java.lang.Class *cls*)

Возвращает абстрактное логическое значение, указывающее, может ли объект Datum быть преобразован в *cls*.

longValue()

Возвращает представление объекта Datum в виде значения типа long.

makeJdbcArray(int *arraySize*)

Возвращает абстрактное представление объекта Datum в виде java.lang.Object.

setBytes(byte[] *array*)

Устанавливает значение объекта Datum при помощи байтового массива.

setShareBytes(byte[] *array*)

Устанавливает значение объекта Datum при помощи байтового массива.

shareBytes()

Возвращает массив byte[].

stringValue()

Возвращает представление объекта Datum в виде java.lang.String.

timestampValue()

Возвращает представление объекта Datum в виде java.sql.Timestamp.

timeValue()

Возвращает представление объекта Datum java.sql.Time.

toJdbc()

Возвращает the JDBC-представление объекта Datum в виде абстрактного объекта java.lang.Object.

Унаследованные методы

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

DatumWithConnection

Расширяет класс Datum. Класс DatumWithConnection является родителем для подклассов ARRAY, BFILE, BLOB, CLOB, OPAQUE, REF и STRUCT. Появился в Oracle9*i*.

Конструкторы класса:

```
DatumWithConnection()
```

и

```
DatumWithConnection(byte[] elements)
```

Методы

assertNotNull(java.sql.Connection *conn*)

Статический метод.

assertNotNull(TypeDescriptor *desc*)

Статический метод.

getConnection()

Возвращает OracleConnection. Не рекомендуется для использования начиная с версии 9.0.0. Вместо него рекомендованы методы getJavaSqlConnection(), getInternalConnection() или getOracleConnection().

getJavaSqlConnection()

Возвращает объект java.sql.Connection, соответствующий приемнику.

getOracleConnection()

Возвращает объект OracleConnection, соответствующий приемнику.

Унаследованные методы

Наследует все методы от oracle.sql.Datum.

INTERVALYM

Расширяет класс Datum. Применяется для измерения интервалов времени. Появился в Oracle9*i*.

Конструкторами класса являются:

```
INTERVALYM()
INTERVALYM(byte[] intervalYM)
INTERVALYM(java.lang.String str)
```

при этом экземпляр инициализируется значениями 0:0, *intervalYM* или *str*.

Методы

isConvertibleTo(java.lang.Class *cls*)

Возвращает логическое значение, указывающее, можно ли преобразовать объект в класс *cls*.

makeJdbcArray(int *arraySize*)

Возвращает объект java.lang.Object, содержащий представление объекта Datum в массиве размера *arraySize*.

toBytes()

Возвращает объект INTERVALYM в виде байтового массива.

toBytes(java.lang.String *str*)

Возвращает в байтовом массиве результат преобразования строки *str* в тип INTERVALYM. Статический метод.

toJdbc()

Возвращает объект java.lang.Object с JDBC-представлением объекта INTERVALYM.

toString()

Возвращает в java.lang.String результат преобразования INTERVALYM в строку.

toString(byte[] *inparray*)

Возвращает в java.lang.String результат преобразования INTERVALYM, представленного массивом *inparray*, в строку.

Унаследованные методы

Наследует все методы от oracle.sql.Datum, за исключением isConvertibleTo, makeJdbcArray и toJdbc.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

JAVA_STRUCT

Расширяет класс oracle.sql.struct. Появился в Oracle9*i*.

Конструктор класса:

```
JAVA_STRUCT(StructDescriptor type, java.sql.Connection conn,
    java.lang.Object[] attributes)
```

Методы

toJdbc()

Возвращает java.lang.Object.

Унаследованные методы

Наследует все методы от oracle.sql.STRUCT, за исключением toJdbc.

Следующие методы унаследованы от oracle.sql.DatumWithConnection:

assertNotNull

getOracleConnection

Наследует все методы от oracle.sql.Datum, за исключением isConvertibleTo и toJdbc.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

NUMBER

Расширяет класс oracle.sql.Datum. Используется для преобразования числовых типов данных Oracle в числовые типы Java и наоборот.

Класс имеет следующие конструкторы:

```
NUMBER()
NUMBER(java.math.BigDecimal BigDecNum)
NUMBER(java.math.BigInteger BigIntNum)
NUMBER(boolean boolNum)
NUMBER(byte byteNum)
NUMBER(byte[] num)
NUMBER(double doubleNum)
NUMBER(float floatNum)
NUMBER(int intNum)
NUMBER(java.lang.Object obj)
NUMBER(java.lang.String StringNum, int scale)
NUMBER(long longNum)
NUMBER(short shortNum)
```

В каждом из конструкторов в качестве инициализирующего значения выступает переменная. В конструкторе на базе java.lang.String основание системы счисления преобразуемого числа задается аргументом *scale*.

Методы

abs()

Возвращает значение типа NUMBER, инициализированное абсолютным значением NUMBER.

acos()

Возвращает значение типа NUMBER, инициализированное арккосинусом значения NUMBER.

add(NUMBER *n*)

Возвращает значение типа NUMBER, инициализированное значением NUMBER плюс *n*.

asin()

Возвращает значение типа NUMBER, инициализированное арксинусом значения NUMBER.

atan()

Возвращает значение типа NUMBER, инициализированное арктангенсом значения NUMBER.

atan2(NUMBER *x*)

Возвращает значение типа NUMBER, инициализированное значением atan2(NUMBER/*x*).

bigDecimalValue()

Возвращает значение, преобразованное из внутреннего типа Oracle Number в java.math.BigDecimal.

bigIntegerValue()

Возвращает значение, преобразованное из внутреннего типа Oracle Number в java.math.BigInteger.

booleanValue()

Возвращает значение внутреннего типа Oracle Number, преобразованное к типу Boolean.

byteValue()

Возвращает значение внутреннего типа Oracle Number, преобразованное к типу byte.

ceil()

Возвращает значение типа NUMBER, инициализированное как значение NUMBER, округленное до ближайшего целого сверху.

compareTo(NUMBER *n*)

Возвращает –1,если значение NUMBER меньше *n*, 0, если значение NUMBER равно *n*, 1 – если значение NUMBER больше *n*.

cos()

Возвращает значение типа NUMBER, инициализированное косинусом значения NUMBER.

cosh()

Возвращает значение типа NUMBER, инициализированное гиперболическим косинусом значения NUMBER.

decrement()

Возвращает значение типа NUMBER, инициализированное значением NUMBER, уменьшенным на единицу.

div(NUMBER *n*)

Возвращает значение типа NUMBER, инициализированное частным, полученным в результате деления NUMBER на *n*.

doubleValue()

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу double.

e()

Возвращает значение типа NUMBER, инициализированное значением *e*. Статический метод.

exp()

Возвращает значение типа NUMBER, инициализированное значением *e*, возведенным в степень значения NUMBER.

floatingPointRound(int *precision*)

Возвращает значение типа NUMBER, инициализированное значением NUMBER, округленным до *precision* значащих десятичных разрядов.

floatValue()

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу float.

floor()

Возвращает значение типа NUMBER, инициализированное как ближайшее целое снизу для значения NUMBER.

formattedTextToNumber(java.lang.String *num*, java.lang.String *fmt*, java.lang.String *lang*)

Возвращает значение типа NUMBER, преобразованное из *num* по формату *fmt* для языка *lang*. Статический метод.

increment()

Возвращает значение типа NUMBER, инициализированное значением NUMBER, увеличенным на единицу.

inInitialization()

Возвращает логическое значение, указывающее состояние инициализации. Статический метод. Только для Oracle8*i*.

intValue()

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу integer.

isConvertibleTo(java.lang.Class *cls*)

Возвращает TRUE, если объект может быть преобразован в класс *cls*.

isInf()

Возвращает TRUE, если значение NUMBER равно плюс или минус бесконечности.

isInt()

Возвращает TRUE, если значение NUMBER является конечным целым числом.

isNegInf()

Возвращает TRUE, если значение NUMBER равно минус бесконечности.

isPosInf()

Возвращает TRUE, если значение NUMBER равно плюс бесконечности.

isValid(byte[] *num*)

Проверяет, действительно ли значение *num* является числом. Статический метод.

isZero()

Возвращает TRUE, если значение NUMBER равно 0.

ln()

Возвращает значение типа NUMBER, инициализированное натуральным логарифмом значения NUMBER.

ln10()

Возвращает значение типа NUMBER, инициализированное значением десятичного логарифма значения NUMBER.

log(NUMBER *base*)

Возвращает значение типа NUMBER, инициализированное логарифмом NUMBER по основанию *base*.

longValue()

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу long.

makeJdbcArray(int arraySize)

Возвращает в объекте java.lang.Object представление объекта Datum в виде массива размером *arraySize*.

mod(NUMBER *n*)

Возвращает значение типа NUMBER, инициализированное остатком от деления значения NUMBER на *n*.

mul(NUMBER *n*)

Возвращает значение типа NUMBER, инициализированное произведением значения NUMBER и *n*.

negate()

Возвращает значение типа NUMBER, инициализированное значением NUMBER с противоположным знаком.

negInf()

Возвращает значение типа NUMBER, инициализированное минус бесконечностью. Статический метод.

pi()

Возвращает значение типа NUMBER, инициализированное значением π. Статический метод.

posInf()

Возвращает значение типа NUMBER, инициализированное положительной бесконечностью. Статический метод.

pow(int *exp*)

Возвращает значение типа NUMBER, инициализированное значением NUMBER, возведенным в целую степень *exp*.

pow(NUMBER *exp*)

Возвращает значение типа NUMBER, инициализированное значением NUMBER, возведенным в степень *exp*.

round(int *decimal_place*)

Возвращает значение типа NUMBER, инициализированное значением NUMBER, округленным до десятичного разряда *decimal_place*.

scale(int *left*, int *right*, boolean[] *big*)

Возвращает значение типа NUMBER, инициализированное значением, которое определяется округлением слева и справа. Если слева окажется слишком много разрядов, то значение *big* равно TRUE.

shift(int *digits*)

Возвращает значение типа NUMBER, инициализированное значением NUMBER, сдвинутым на *digits* десятичных разрядов.

shortValue()

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу short.

sign()

Возвращает -1, если NUMBER отрицательно, 0 – если значение типа NUMBER равно 0, и положительное число – если NUMBER положительно.

sin()

Возвращает значение типа NUMBER, инициализированное синусом значения NUMBER.

sinh()

Возвращает значение типа NUMBER, инициализированное гиперболическим синусом значения NUMBER.

sqroot()

Возвращает значение типа NUMBER, инициализированное квадратным корнем значения NUMBER.

stringValue()

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу java.lang.String.

sub(NUMBER *n*)

Возвращает значение типа NUMBER, инициализированное разностью между NUMBER и *n*.

tan()

Возвращает значение типа NUMBER, инициализированное тангенсом значения NUMBER.

tanh()

Возвращает значение типа NUMBER, инициализированное гиперболическим тангенсом значения NUMBER.

textToPrecisionNumber(java.lang.String *num*, boolean *precflag*, int *preclen*, boolean *scaleflag*, int *scalelen*, java.lang.String *lang*)

Возвращает значение типа NUMBER, инициализированное значением *num*. Статический метод.

toBigDecimal(byte[] *num*)

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу java.math.BigDecimal.

toBigInteger(byte[] *num*)

Возвращает значение, преобразованное из внутреннего типа Oracle Number, к типу java.math.BigInteger.

toBoolean(byte[] *num*)

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу Boolean. Статический метод.

toByte(byte[] *num*)

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу byte. Статический метод.

toBytes()

Возвращает внутренний байтовый массив Oracle Number.

toBytes(boolean *boolNum*)

Преобразует *boolNum* к байтовому массиву Oracle Number. Статический метод.

toBytes(byte *byteNum*)

Преобразует *byteNum* к байтовому массиву Oracle Number. Статический метод.

toBytes(double *doubleNum*)

Преобразует *doubleNum* к байтовому массиву Oracle Number. Статический метод.

toBytes(float *floatNum*)

Преобразует *floatNum* к байтовому массиву Oracle Number. Статический метод.

toBytes(int *intNum*)

Преобразует *intNum* к байтовому массиву Oracle Number. Статический метод.

toBytes(java.lang.String *StringNum*, int *scale*)

Преобразует *StringNum* к байтовому массиву Oracle Number, аргумент *scale* задает основание системы счисления. Статический метод.

toBytes(java.math.BigDecimal *BigDecNum*)

Преобразует *BigDecNum* к байтовому массиву Oracle Number. Статический метод.

toBytes(java.math.BigInteger *BigIntNum*)

Преобразует *BigIntNum* к байтовому массиву Oracle Number. Статический метод.

toBytes(long *longNum*)

Преобразует *longNum* к байтовому массиву Oracle Number. Статический метод.

toBytes(short *shortNum*)

Преобразует *shortNum* к байтовому массиву Oracle Number. Статический метод.

toDouble(byte[] *num*)

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу double. Статический метод.

toFloat(byte[] *num*)

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу float. Статический метод.

toFormattedText(java.lang.String *fmt*, java.lang.String *lang*)

Возвращает в java.lang.String значение, преобразованное по формату *fmt* для языка *lang*.

toInt(byte[] *num*)

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу integer. Статический метод.

toJdbc()

Возвращает в java.lang.Object представление объекта Datum.

toLong(byte[] *num*)

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу long. Статический метод.

toShort(byte[] *num*)

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу short. Статический метод.

toString(byte[] *num*)

Возвращает значение, преобразованное из внутреннего типа Oracle Number к типу java.lang.String. Статический метод.

toText(int *outStringLength*, java.lang.String *lang*)

Возвращает java.lang.String с неформатированным представлением значения типа NUMBER.

truncate(int *decimal_place*)

Возвращает значение типа NUMBER, инициализированное значением NUMBER, усеченным до десятичного разряда *decimal_place*.

zero()

Возвращает значение типа NUMBER, инициализированное нулем. Статический метод.

Унаследованные методы

Методы, перечисленные ниже, унаследованы от oracle.sql.Datum:

asciiStreamValue
binaryStreamValue
characterStreamValue
dateValue
equals
getBytes
getLength
getStream
setBytes
setShareBytes
shareBytes
timestampValue
timeValue

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

OPAQUE

Расширяет класс DatumWithConnection. Появился в Oracle9*i*.

Конструктор класса:

```
OPAQUE(OpaqueDescriptor type, java.sql.Connection conn,
   java.lang.Object value)
```

Методы

getBytesValue()

Возвращает байтовый массив, представляющий атрибуты в том формате, который фактически используется в базе данных.

getConnection()

Возвращает OracleConnection.

getDescriptor()

Возвращает OpaqueDescriptor.

getJavaSqlConnection()

Возвращает java.sql.Connection.

getSQLTypeName()

Возвращает в java.lang.String имя структурированного типа SQL, который представляет данный объект Struct.

getValue()

Возвращает в java.lang.Object значение Opaque.

isConvertibleTo(java.lang.Class *jClass*)

Возвращает логическое значение TRUE, если объект Datum можно преобразовать к классу *jClass*.

toJdbc()

Возвращает представление java.lang.Object для объекта Datum.

Унаследованные методы

Следующие методы унаследованы от oracle.sql.DatumWithConnection:

assertNotNull

getOracleConnection

Наследует все методы от oracle.sql.Datum, за исключением isConvertibleTo и toJdbc.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

OpaqueDescriptor

Расширяет класс oracle.sql.TypeDescriptor. Реализует java.io.Serializable. Появился в Oracle9*i*.

Конструктор класса:

```
OpaqueDescriptor(oracle.sql.SQLName name, oracle.sql.OracleTypeOPAQUE type, java.sql.
Connection connection)
```

Унаследованные поля

Поле, унаследованное от класса oracle.sql.TypeDescriptor:

DEBUG_SERIALIZATION

Методы

createDescriptor(java.lang.String *name*, java.sql.Connection *conn*)

Возвращает OpaqueDescriptor. Статический метод.

descType()

Возвращает java.lang.String с описанием типа.

getMaxLength()

Возвращает значение типа LONG – максимальное количество байт, которое может хранить данный объект Opaque.

getTypeCode()

Возвращает целое число – код типа Opaque: OracleTypes.OPAQUE.

hasFixedSize()

Возвращает логическое значение, указывающее, имеет ли тип Opaque фиксированный размер.

hasUnboundedSize()

Возвращает логическое значение, указывающее, имеет ли тип Opaque неограниченный размер.

isModeledInC()

Возвращает логическое значение, указывающее, смоделирован ли тип Opaque в языке C.

isTrustedLibrary()

Возвращает логическое значение, указывающее, задана ли для типа Opaque библиотека Trusted Library, реализующая функции поддержки.

Унаследованные методы

Методы, перечисленные ниже, унаследованы от oracle.sql.TypeDescriptor:

getName

getSubtypeName

getConnection

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

OracleSQLOutput

Расширяет java.lang.Object. Реализует oracle.jdbc2.SQLOutput. Только для Oracle8*i*.

Методы

writeArray(ARRAY *x*)

Записывает массив *x* в поток.

writeArray(oracle.jdbc2.Array *x*)

Записывает массив *x* в поток.

writeAsciiStream(java.io.InputStream *x*)

Записывает следующий атрибут в поток в виде потока ASCII-символов *x*.

writeBfile(BFILE *x*)

Записывает следующий атрибут в поток в виде *x* типа BFILE.

writeBigDecimal(java.math.BigDecimal *x*)

Записывает следующий атрибут в поток в виде *x* типа java.math.BigDecimal.

writeBinaryStream(java.io.InputStream *x*)

Записывает следующий атрибут в поток в виде *x* типа java.math.InputStream.

writeBlob(BLOB *x*)

Записывает следующий атрибут в поток в виде *x* типа BLOB.

writeBlob(oracle.jdbc2.Blob *x*)

Записывает следующий атрибут в поток в виде *x* типа oracle.jdbc2.Blob.

writeBiilean(Boolean *x*)

Записывает следующий атрибут в поток в виде *x* типа Java Boolean.

writeByte(byte *x*)

Записывает следующий атрибут в поток в виде *x* типа Java byte.

writeBytes(byte[] *x*)

Записывает следующий атрибут в поток в виде байтового массива *x*.

writeCHAR(CHAR *x*)

Записывает следующий атрибут в поток в виде *x* типа CHAR.

writeCharacterStream(java.io.Reader *x*)

Записывает следующий атрибут в поток в виде *x* типа java.io.Reader.

writeClob(CLOB *x*)

Записывает следующий атрибут в поток в виде *x* типа CLOB.

writeClob(oracle.jdbc2.Clob *x*)

Записывает следующий атрибут в поток в виде *x* типа CLOB.

writeDATE(DATE *x*)

Записывает следующий атрибут в поток в виде *x* типа DATE.

writeDate(java.sql.Date *x*)

Записывает следующий атрибут в поток в виде *x* типа java.sql.Date

writeDouble(double *x*)

Записывает следующий атрибут в поток в виде *x* типа Double.

writeFloat(float *x*)

Записывает следующий атрибут в поток в виде *x* типа Float.

writeInt(int *x*)

Записывает следующий атрибут в поток в виде *x* типа integer.

writeLong(long *x*)

Записывает следующий атрибут в поток в виде *x* Java-типа long.

writeNUMBER(NUMBER *x*)

Записывает следующий атрибут в поток в виде *x* типа NUMBER.

writeObject(java.lang.Object *x*)

Записывает следующий атрибут в поток в виде *x* типа java.lang.Object.

writeObject(oracle.jdbc2.SQLData *x*)

Записывает следующий атрибут в поток в виде *x* типа oracle.jdbc2.SQLData.

writeOracleObject(Datum *x*)

Записывает следующий атрибут в поток в виде *x* типа Datum.

writeRAW(RAW *x*)

Записывает следующий атрибут в поток в виде *x* типа RAW.

writeRef(oracle.jdbc2.Ref *x*)

Записывает следующий атрибут в поток в виде *x* типа oracle.jdbc2.Ref.

writeRef(REF *x*)

Записывает следующий атрибут в поток в виде *x* типа REF.

writeROWID(ROWID *x*)

Записывает следующий атрибут в поток в виде *x* типа ROWID.

writeShort(short *x*)

Записывает следующий атрибут в поток в виде *x* типа short.

writeString(java.lang.String *x*)

Записывает следующий атрибут в поток в виде *x* типа java.lang.String.

writeStruct(oracle.jdbc2.Struct *x*)

Записывает следующий атрибут в поток в виде *x* типа oracle.jdbc2.Struct.

writeStruct(STRUCT *x*)

Записывает следующий атрибут в поток в виде *x* типа STRUCT.

writeTime(java.sql.Time *x*)

Записывает следующий атрибут в поток в виде *x* типа java.sql.Time.

writeTimestamp(java.sql.Timestamp *x*)

Записывает следующий атрибут в поток в виде *x* типа java.sql.Timestamp.

Унаследованные методы

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

RAW

Расширяет класс oracle.sql.Datum.

Конструкторы класса:

```
RAW(byte[] raw_bytes)
RAW(java.lang.Object val)
```

Второй конструктор не рекомендуется применять в версии Oracle9*i* Release 2.

Методы

hexString2Bytes(java.lang.String *hexString*)

Возвращает байтовый массив, представляющий собой строку, преобразованную из шестнадцатеричных разрядов *hexString*. Статический метод.

isConvertibleTo(java.lang.Class *jClass*)

Возвращает логическое значение, указывающее, можно ли преобразовать данный объект данных в класс *jClass*.

newRAW(java.lang.Object *obj*)

Возвращает значение типа RAW, созданное на основе объекта *obj*, действующее как конструктор RAW(Object) начиная с версии, следующей за Oracle JDBC 9.2. Статический метод.

oldRAW(java.lang.Object *obj*)

Возвращает значение типа RAW, созданное на основе объекта *obj*, действующее как конструктор RAW(Object) для версии Oracle JDBC 9.2 и ей предшествующих. Статический метод.

stringValue()

Возвращает java.lang.String как результат преобразования данного объекта.

toJdbc()

Возвращает java.lang.Object как результат преобразования данного объекта.

Унаследованные методы

Наследует все методы от oracle.sql.Datum, за исключением isConvertibleTo, stringValue и toJdbc.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

REF

Расширяет класс DatumWithConnection в Oracle9*i*, Datum – в Oracle8*i*. Реализует java.sql.Ref, java.io.Serializable и java.lang.Cloneable в Oracle9*i*, oracle.jdbc2.Ref – в Oracle8*i*.

Методы

clone()

Возвращает java.lang.Object. Появился в Oracle9*i*.

equals(java.lang.Object *obj*)

Возвращает логическое значение. Появился в Oracle9*i*.

getBaseTypeName()

Возвращает java.lang.String.

getConnection()

Возвращает OracleConnection.

getDescriptor()

Возвращает StructDescriptor.

getJavaSqlConnection()

Возвращает java.sql.Connection. Появился в Oracle9*i*.

getObject()

Возвращает объект java.lang.Object, на который ссылается данный объект Ref. Для JDBC 3.0. Появился в Oracle9*i*.

getObject(java.util.Map *map*)

Возвращает объект java.lang.Object. Появился в Oracle9*i*.

getSQLTypeName()

Возвращает объект java.lang.String. Появился в Oracle9*i*.

getSTRUCT()

Возвращает STRUCT.

getValue()

Возвращает объект java.lang.Object.

getValue(java.util.Map *map*)

Возвращает объект java.lang.Object.

hashCode()

Возвращает целое число. Появился в Oracle9*i*.

isConvertibleTo(java.lang.Class *jClass*)

Возвращает логическое значение, указывающее, можно ли преобразовать данный объект данных к классу *jClass*.

setObject(java.lang.Object *value*)

Присваивает значение структурного типа данному экземпляру объекта. Для JDBC 3.0. Появился в Oracle9*i*.

setValue(java.lang.Object *value*)

Устанавливает значение.

toJdbc()

Возвращает java.lang.Object как результат преобразования данного объекта данных к объекту принятого в Java по умолчанию типа.

Унаследованные методы

Методы, перечисленные ниже, унаследованы от DatumWithConnection в Oracle9*i*:

assertNotNull

assertNotNull

getOracleConnection

Наследует все методы от oracle.sql.Datum, за исключением isConvertibleTo и toJdbc.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

ROWID

Расширяет класс oracle.sql.Datum.

Методы

isConvertibleTo(java.lang.Class *jClass*)

Возвращает логическое значение, указывающее, может ли указанный объект данных быть преобразован к классу *jClass*.

stringValue()

Возвращает в java.lang.String результат преобразования из данного объекта.

toJdbc()

Возвращает объект java.lang.Object, полученный преобразованием из указанного объекта данных.

Унаследованные методы

Наследует все методы от oracle.sql.Datum, за исключением isConvertibleTo, stringValue и toJdbc.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

STRUCT

Расширяет класс oracle.sql.DatumWithConnection. Реализует java.sql.Struct. Является родительским классом для JAVA_STRUCT.

Конструктор класса:

```
STRUCT(StructDescriptor type, java.sql.Connection conn, java.lang.Object[] attributes)
```

Методы

getAttributes()

Возвращает массив java.lang.Object[] с атрибутами.

getAttributes(java.util.Map *map*)

Возвращает массив java.lang.Object[] с атрибутами.

getAutoBuffering()

Возвращает логическое значение.

getConnection()

Возвращает OracleConnection.

getDescriptor()

Возвращает StructDescriptor.

getJavaSqlConnection()

Возвращает java.sql.Connection.

getOracleAttributes()

Возвращает масив Datum[].

getSQLTypeName()

Возвращает java.lang.String с именем структурного типа SQL.

isConvertibleTo(java.lang.Class *jClass*)

Возвращает логическое значение, указывающее, может ли объект Datum быть преобразован в класс *jClass*.

setAutoBuffering(Boolean *enable*)

Возвращает логическое значение.

toJdbc()

Возвращает java.lang.Object.

Унаследованные методы

Методы, перечисленные ниже, унаследованы от oracle.sql.DatumWithConnection:

assertNotNull

getOracleConnection

Наследует все методы от oracle.sql.Datum, за исключением isConvertibleTo и toJdbc.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

StructDescriptor

Расширяет класс oracle.sql.TypeDescripton. Реализует java.io.Serializable.

Конструктор класса:

```
StructDescriptor(oracle.sql.SQLName name, oracle.sql.OracleTypeADT type,
java.sql.Connection connection)
```

Унаследованные поля

Поле, унаследованное от oracle.sql.TypeDescriptor:

DEBUG_SERIALIZATION

Методы

createDescriptor(java.lang.String *name*, java.sql.Connection *conn*)

Возвращает StructDescriptor. Статический метод.

descType()

Возвращает описание в виде java.lang.String.

getJavaClassName()

Возвращает java.lang.String с внешним именем типа JAVA_STRUCT.

getLanguage()

Возвращает java.lang.String.

getLength()

Возвращает целое число – количество полей объектного типа.

getLocalAttributeCount()

Возвращает целое число – количество атрибутов, определенных в подтипе.

getMetaData()

Возвращает java.sql.ResultSetMetaData.

getSubtypeNames()

Возвращает java.lang.String[] с именами непосредственных подтипов SQL.

getSupertypeName()

Возвращает java.lang.String с именем непостредственного супертипа SQL.

getTypeCode()

Возвращает целое число.

getTypeVersion()

Возвращает целое число.

isFinalType()

Возвращает логическое значение, указывающее, является ли объектный тип финальным.

isInstantiable()

Возвращает логическое значение, указывающее, может ли объектный тип иметь реализацию.

isJavaObject()

Возвращает логическое значение, определяющее объектный тип как JAVA_STRUCT или как STRUCT.

isSubtype()

Возвращает логическое значение.

Унаследованные методы

Перечисленные ниже методы наследуются от oracle.sql.TypeDescriptor:

getName

getSubtypeName

setConnection

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

TIMESTAMP

Расширяет класс oracle.sql.Datum.

Конструкторы класса:

```
TIMESTAMP()
TIMESTAMP(byte[] timestamp)
TIMESTAMP(DATE date)
TIMESTAMP(java.sql.Date date)
TIMESTAMP(java.lang.String str)
TIMESTAMP(java.sql.Time time)
TIMESTAMP(java.sql.Timestamp timestamp)
```

Переменные конструктора представляют собой исходные значения для создаваемых объектов.

Методы

isConvertibleTo(java.lang.Class *cls*)

Возвращает логическое значение, указывающее, можно ли преобразовывать объект в класс *cls*.

makeJdbcArray(int *arraySize*)

Возвращает java.lang.Object – представление Datum.

timestampValue()

Возвращает java.sql.Timestamp – преобразование внутреннего типа Oracle.

TimeZoneConvert(java.sql.Connection *conn*, TIMESTAMP *tstamp*, java.util.TimeZone *tz1*, java.util.TimeZone *tz2*)

Возвращает TIMESTAMP, преобразованный из часового пояса *tz1* в *tz2*. Статический метод.

toBytes()

Возвращает байтовый массив для преобразованного Oracle Timestamp.

toBytes(DATE *date*)

Возвращает байтовый массив для преобразованного Oracle DATE. Статический метод.

toBytes(java.lang.String *str*)

Возвращает байтовый массив для преобразованного Java String. Статический метод.

toBytes(java.sql.Date *date*)

Возвращает байтовый массив для преобразованного Java Date. Статический метод.

toBytes(java.sql.Time *time*)

Возвращает байтовый массив для преобразованного Java Time. Статический метод.

toBytes(java.sql.Timestamp *timestamp*)

Возвращает байтовый массив для преобразованного Java Timestamp. Статический метод.

toDate(byte[] *timestamp*)

Возвращает java.sql.Date, преобразованный из байтового массива, представляющего объект TIMESTAMP. Статический метод.

toDATE(byte[] *timestamp*)

Возвращает DATE, преобразованный из байтового массива, представляющего объект TIMESTAMP. Статический метод.

toJdbc()

Возвращает java.lang.Object – представление объекта Datum.

toString(byte[] *timestamp*)

Возвращает java.lang.String, преобразованный из типа TIMESTAMP. Статический метод.

toTime(byte[] *timestamp*)

Возвращает java.sql.Time, преобразованный из байтового массива, представляющего объект TIMESTAMP. Статический метод.

toTimestamp(byte[] *timestamp*)

Возвращает java.sql.Timestamp, преобразованный из байтового массива, представляющего объект TIMESTAMP. Статический метод.

Унаследованные методы

Наследует все методы от oracle.sql.Datum, за исключением isConvertibleTo, makeJdbcAray, timestampValue и toJdbc.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

TIMESTAMPLTZ

Расширяет класс oracle.sql.Datum.

Конструкторы данного класса выглядят следующим образом:

```
TIMESTAMPLTZ()
TIMESTAMPLTZ(byte[] timestampltz)
TIMESTAMPLTZ(java.sql.Connection conn, java.util.Calendar sess, java.sql.Date date)
TIMESTAMPLTZ(java.sql.Connection conn, java.util.Calendar sess, DATE date)
TIMESTAMPLTZ(java.sql.Connection conn, java.util.Calendar sess, java.lang.String str) --
Исключен из числа рекомендуемых к применению в Oracle9i
TIMESTAMPLTZ(java.sql.Connection conn, java.util.Calendar sess, java.sql.Time time)
TIMESTAMPLTZ(java.sql.Connection conn, java.util.Calendar sess, java.sql.Timestamp timestamp)
TIMESTAMPLTZ(java.sql.Connection conn, java.sql.Date date)
TIMESTAMPLTZ(java.sql.Connection conn, DATE date)
TIMESTAMPLTZ(java.sql.Connection conn, java.sql.Date date, java.util.Calendar dbtz) --
Исключен из числа рекомендуемых к применению в Oracle9i
TIMESTAMPLTZ(java.sql.Connection conn, DATE date, java.util.Calendar dbtz) -- Исключен из
числа рекомендуемых к применению в Oracle9i
TIMESTAMPLTZ(java.sql.Connection conn, java.lang.String str) -- Исключен из числа
рекомендуемых к применению в Oracle9i
TIMESTAMPLTZ(java.sql.Connection conn, java.lang.String str, java.util.Calendar dbtz) --
Исключен из числа рекомендуемых к применению в Oracle9i
TIMESTAMPLTZ(java.sql.Connection conn, java.sql.Time time)
TIMESTAMPLTZ(java.sql.Connection conn, java.sql.Time time, java.util.Calendar dbtz) --
Исключен из числа рекомендуемых к применению в Oracle9i
TIMESTAMPLTZ(java.sql.Connection conn, java.sql.Timestamp timestamp)
TIMESTAMPLTZ(java.sql.Connection conn, java.sql.Timestamp timestamp, java.util.Calendar dbtz)
-- Исключен из числа рекомендуемых к применению в Oracle9i
```

Методы

isConvertibleTo(java.lang.Class *cls*)

Возвращает логическое значение, указывающее, можно ли преобразовывать объект в класс *cls*.

makeJdbcArray(int *arraySize*)

Возвращает java.lang.Object как представление объекта Datum.

timestampValue(java.sql.Connection *conn*, java.util.Calendar *dbtz*)

Возвращает java.sql.Timestamp, преобразованный из внутреннего представления Oracle.

toBytes()

Возвращает байтовый массив для преобразованного объекта в Oracle TIMESTAMPLTZ.

toBytes(java.sql.Connection *conn*, DATE *date*, java.util.Calendar *dbtz*)

Статический метод. Исключен из числа рекомендуемых к применению в Oracle9*i*.

toBytes(java.sql.Connection *conn*, java.lang.String *str*, java.util.Calendar *dbtz*)

Возвращает байтовый массив с результатом преобразования объекта Java String в Oracle TIMESTAMPLTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.sql.Date *date*, java.util.Calendar *dbtz*)

Статический метод. Исключен из числа рекомендуемых к применению в Oracle9*i*.

toBytes(java.sql.Connection *conn*, java.sql.Time *time*, java.util.Calendar *dbtz*)

Статический метод. Исключен из числа рекомендуемых к применению в Oracle9*i*.

toBytes(java.sql.Connection *conn*, java.util.Calendar *cal*, DATE *date*)

Возвращает байтовый массив с результатом преобразования объекта Oracle DATE в Oracle TIMESTAMPLTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.util.Calendar *cal*, java.sql.Date *date*)

Возвращает байтовый массив с результатом преобразования Java Date в Oracle TIMESTAMPLTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.util.Calendar *cal*, java.sql.Time *time*)

Возвращает для преобразованного объекта Java Time в Oracle TIMESTAMPLTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.util.Calendar cal, java.sql.Timestamp *timestamp*)

Возвращает байтовый массив с результатом преобразования объекта Java Timestamp в Oracle TIMESTAMPLTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.util.Calendar *sess*, java.lang.String *str*)

Возвращает байтовый массив с результатом преобразования объекта Java String в Oracle TIMESTAMPLTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.sql.Timestamp *timestamp*, java.util.Calendar *dbtz*)

Статический метод. Исключен из числа рекомендуемых к применению в Oracle9*i*.

toDate(java.sql.Connection *conn*, byte[] *timestamp*)

Возвращает объект java.sql.Date, содержащий результат преобразования объекта TIMESTAMPLTZ. Статический метод.

toDATE(java.sql.Connection *conn*, byte[] *timestamp*)

Возвращает объект DATE, преобразованный из типа TIMESTAMPLTZ. Статический метод.

toDate(java.sql.Connection *conn*, byte[] *timestamp*, java.util.Calendar *dbtz*)

Возвращает объект java.sql.Date, содержащий результат преобразования объекта TIMESTAMPLTZ. Статический метод.

toDATE(java.sql.Connection *conn*, byte[] *timestamp*, java.util.Calendar dbtz)

Возвращает объект DATE, преобразованный из типа TIMESTAMPLTZ в Oracle Date. Статический метод.

toJdbc()

Возвращает java.lang.Object – представление объекта Datum.

toString(java.sql.Connection *conn*, byte[] *timestamp*)

Возвращает java.lang.String, преобразованный из типа TIMESTAMP. Статический метод.

toString(java.sql.Connection *conn*, byte[] *timestamp*, java.util.Calendar *dbtz*)

Статический метод. Исключен из числа рекомендуемых к применению в Oracle9*i*.

toTime(java.sql.Connection *conn*, byte[] *timestamp*)

Возвращает java.sql.Time, преобразованный из типа TIMESTAMPLTZ. Статический метод.

toTime(java.sql.Connection *conn*, byte[] *timestamp*, java.util.Calendar *dbtz*)

Возвращает java.sql.Time, преобразованный из типа TIMESTAMPLTZ. Статический метод.

toTimestamp(java.sql.Connection *conn*, byte[] *timestamptz*)

Возвращает java.sql.Timestamp, преобразованный из типа TIMESTAMP. Статический метод.

toTimestamp(java.sql.Connection *conn*, byte[] *timestamptz*, java.util.Calendar *dbtz*)

Возвращает java.sql.Timestamp, преобразованный из типа TIMESTAMP. Статический метод.

Унаследованные методы

Наследует все методы от oracle.sql.Datum, за исключением isConvertibleTo, makeJdbcAray и toJdbc.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

TIMESTAMPTZ

Расширяет класс oracle.sql.Datum.

Конструкторы данного класса выглядят следующим образом:

```
TIMESTAMPTZ()
TIMESTAMPTZ(byte[] timestamptz)
TIMESTAMPTZ(java.sql.Connection conn, java.sql.Date date)
TIMESTAMPTZ(java.sql.Connection conn, DATE date)
TIMESTAMPTZ(java.sql.Connection conn, java.sql.Date date, java.util.Calendar cal)
TIMESTAMPTZ(java.sql.Connection conn, java.lang.String str)
```

```
TIMESTAMPTZ(java.sql.Connection conn, java.lang.String str, java.util.Calendar cal)
TIMESTAMPTZ(java.sql.Connection conn, java.sql.Time time)
TIMESTAMPTZ(java.sql.Connection conn, java.sql.Time time, java.util.Calendar cal)
TIMESTAMPTZ(java.sql.Connection conn, java.sql.Timestamp timestamp)
TIMESTAMPTZ(java.sql.Connection conn, java.sql.Timestamp timestamp, java.util.Calendar cal)
```

Методы

isConvertibleTo(java.lang.Class *cls*)

Возвращает логическое значение, указывающее, можно ли преобразовать объект в *cls*.

makeJdbcArray(int *arraySize*)

Возвращает java.lang.Object – представление объекта Datum.

timestampValue(java.sql.Connection *conn*)

Возвращает java.sql.Timestamp, преобразованный из Oracle TIMESTAMPTZ.

toBytes()

Возвращает байтовый массив, преобразованный из Oracle TIMESTAMPLTZ.

toBytes(java.sql.Connection *conn*, DATE *date*)

Возвращает байтовый массив, преобразованный в тип Oracle TIMESTAMPTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.lang.String str)

Возвращает байтовый массив, преобразованный в тип Oracle TIMESTAMPTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.lang.String *str*, java.util.Calendar *cal*)

Возвращает байтовый массив, преобразованный в тип Oracle TIMESTAMPTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.sql.Date *date*)

Возвращает байтовый массив, преобразованный в тип Oracle TIMESTAMPTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.sql.Date *date*, java.util.Calendar *cal*)

Возвращает байтовый массив, преобразованный в тип Oracle TIMESTAMPTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.sql.Time *time*)

Возвращает байтовый массив, преобразованный в тип Oracle TIMESTAMPTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.sql.Time *time*, java.util.Calendar *cal*)

Возвращает байтовый массив, преобразованный в тип Oracle TIMESTAMPTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.sql.Timestamp *timestamp*)

Возвращает байтовый массив, преобразованный в тип Oracle TIMESTAMPTZ. Статический метод.

toBytes(java.sql.Connection *conn*, java.sql.Timestamp *timestamp*, java.util.Calendar *cal*)

Возвращает байтовый массив, преобразованный в тип Oracle TIMESTAMPTZ. Статический метод.

toDate(java.sql.Connection *conn*, byte[] *timestamptz*)

Возвращает java.sql.Date, преобразованный из типа TIMESTAMPTZ. Статический метод.

toDATE(java.sql.Connection *conn*, byte[] *timestamptz*)

Возвращает DATE, преобразованный из байтового массива, представляющего TIMESTAMPTZ. Статический метод.

toJdbc()

Возвращает java.lang.Object – представление объекта TIMESTAMPTZ.

toString(java.sql.Connection *conn*, byte[] *timestamptz*)

Возвращает java.lang.String, преобразованный из TIMESTAMPTZ. Статический метод.

toTime(java.sql.Connection *conn*, byte[] *timestamptz*)

Возвращает static java.sql.Time, преобразованный из байтового массива, представляющего TIMESTAMPTZ. Статический метод.

toTimestamp(java.sql.Connection *conn*, byte[] *timestamptz*)

Возвращает статический java.sql.Timestamp, преобразованный из байтового массива, представляющего TIMESTAMPTZ. Статический метод.

Унаследованные методы

Наследует все методы от oracle.sql.Datum, за исключением isConvertibleTo, makeJdbcArray и toJdbc.

Наследует все методы от java.lang.Object, за исключением clone, equals и finalize.

TypeDescriptor

Расширяет класс java.lang.Object. Реализует java.io.Serializable. Является родительским классом по отношению к ArrayDescriptor, OpaqueDescriptor и StructDescriptor.

Поле

DEBUG_SERIALIZATION

Методы

getName()

Возвращает в java.lang.String полное уточненное имя типа.

getSubtypeName(OracleConnection *conn*, byte[] *image*, long *offset*)

Возвращает java.lang.String. Статический метод.

setConnection(java.sql.Connection *connection*)

См. документацию Oracle.

Унаследованные методы

Наследует все методы от java.lang.Object, за исключением clone и finalize.

III

Инструменты и утилиты

В этой части книги представлены команды и спецификации файлов для различных инструментов и утилит, применяемых для управления СУБД Oracle и взаимодействия с ней.

Глава 12 «SQL*Plus» содержит описание команд и элементов форматирования, доступных пользователю в SQL*Plus – интерфейсе командной строки для Oracle, который предназначен для ввода команд SQL, кода PL/SQL, а также для выполнения файлов сценариев.

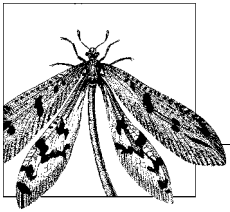
Глава 13 «Экспорт и импорт» содержит перечень команд, предлагаемых утилитами Export (копирование данных из базы данных в двоичный файл) и Import (загрузка данных из двоичного файла в базу данных Oracle). Эти утилиты позволяют получать сведения о структуре и содержимом базы данных Oracle.

Глава 14 «SQL*Loader» описывает команды утилиты SQL*Loader, предназначенной для загрузки данных в стандартных файловых форматах операционной системы в базу данных Oracle и выполнения различных преобразований данных в процессе загрузки.

Глава 15 «Резервное копирование и восстановление» кратко описывает принципы резервного копирования и восстановления данных Oracle и шаблоны процедур, обеспечивающих управляемое пользователем копирование и восстановление. Приводится перечень команд специальной утилиты Oracle RMAN (Recovery Manager).

Глава 16 «Enterprise Manager» описывает возможности Enterprise Manager – консоли с графическим интерфейсом пользователя, позволяющей управлять сервером Oracle.

Глава 17 «Производительность» рассматривает основные инструменты Oracle, с помощью которых можно оценить и улучшить производительность. Описываются оптимизаторы и подсказки для них в SQL. Приводится описание таких средств оптимизации, как Explain Plan, TKPROF, AUTOTRACE, UTLBSTAT, UTLESTAT и Statspack.



12
SQL*Plus

SQL*Plus предоставляет интерфейс командной строки к БД Oracle, и наверное, эту утилиту Oracle применяют чаще других. Она позволяет вводить команды SQL, код PL/SQL и выполнять сценарии из файлов. Несмотря на то что некоторые из ее возможностей присутствуют в других продуктах, таких как Recovery Manager, Enterprise Manager и различных средствах построения отчетов, благодаря долговечности и универсальности, эта программа заняла прочное место в инструментарии большинства пользователей Oracle.

SQL*Plus – одна из старейших утилит Oracle. Первоначально она разрабатывалась как интерфейс командной строки UFI (User Friendly Interface) – дружественный пользовательский интерфейс для System R, одной из первых реляционных баз данных, разработанной IBM. В 1980-х годах корпорация Oracle расширила возможности UFI и вскоре переименовала ее в AUFI (Advanced User Friendly Interface). С выходом Oracle 5.0 в 1985 году продукт был переименован в SQL*Plus. В каждой очередной версии сервера Oracle этот почтенный продукт пополняется новыми возможностями, но основа его не изменяется.

Запуск SQL*Plus

Практически всегда SQL*Plus запускается командой *sqlplus* из командной строки операционной системы. В системах семейства Microsoft Windows команды *sqlplus* и *sqlplusw* запускают SQL*Plus в командном окне или в собственном окне соответственно; можно также запустить программу, выбрав ее в меню запуска. Первые реализации SQL*Plus для Windows имели имена, подобные *PLUS33* или *PLUS80W*, в зависимости от версии сервера и типа пользовательского интерфейса.

Остерегайтесь вводить пароль в качестве аргумента SQL*Plus в командной строке. В Linux и Unix такие пароли могут быть легко доступны другим пользователям.

Рассмотрим синтаксис команды запуска SQL*Plus. Обратите внимание на то, что параметры -RESTRICT и -MARKUP появились в Oracle8*i*, а -HELP и -VERSION – в Oracle9*i*.

```
SQLPLUS [[-S[ILENT]] [-H[ELP]] [-V[ERSION]]
    [-R[ESTRICT] уровень] [L[OGON]]
```

```
[-M[ARKUP] 'параметры_разметки']
[пользователь[/пароль][@строка_соединения] | / [AS (SYSDBA | SYSOPER)]] | /NOLOG]
[@файл_сценария [арг1 арг2 арг3...]]] | - | -?
```

–S[*ILENT*]

Предписывает SQL*Plus запуститься в фоновом режиме без стартовых сообщений, приглашения на ввод и отображения выполняемых команд.

–H[*ELP*]

Выводит краткую справку по синтаксису SQL*Plus. В версиях, предшествующих Oracle9*i*, используйте команду *sqlplus*.

–V[*ERSION*]

Выводит версию SQL*Plus и сведения об авторских правах. В версиях, предшествующих Oracle9*i*, для получения этой информации следует применять команду *sqlplus –?*.

–R[*ESTRICT*] *уровень*

Ограничивает доступные пользователю возможности SQL*Plus. Уровень может принимать одно из следующих значений:

1 Запрещает команды EDIT, HOST и !.

2 Запрещает команды EDIT, HOST, !, SAVE, SPOOL и STORE.

3 Запрещает команды EDIT, GET, HOST, SAVE, START, @, @@, SPOOL и STORE.

Уровень 3 также отменяет чтение файла *login.sql*. Файл *glogin.sql* будет прочитан, но запрещенные команды не выполнятся.

–L[*OGON*]

Предотвращает повторный запрос утилитой SQL*Plus имени пользователя и пароля, если первые введенные значения оказались неверными. Появился в Oracle9*i* Release 2.

–M[*ARKUP*] *параметры_разметки*

Позволяет указать язык разметки, применяемый для оформления выводимых данных. Все параметры, за исключением HTML, могут отсутствовать. Допустимы следующие варианты:

HTML { [*ON* | *OFF*] }

Определяет применяемый язык разметки и разрешает или запрещает его применение. В Oracle 8.1.6 это обязательный параметр.

HEAD текст

Определяет содержимое тега <head>, выводимого в виде <head>текст</head>.

BODY текст

Определяет содержимое тега <body>, выводимого в виде <body текст>.

TABLE текст

Определяет содержимое тега <table>, используемого для форматирования результатов запроса. Тег выводится в виде <table текст>.

ENTMAP (ON | OFF)

Определяет, будут ли в SQL*Plus действовать принятые в HTML обозначения для специальных символов, такие как < и > для < и >.

SPOOL (ON|OFF)

Определяет, будет ли SQL*Plus выводить в файл простой текст или отформатированный по правилам указанного языка разметки (в данный момент – HTML).

PRE[FORMAT] *(ON|OFF)*

Определяет, должен ли выводимый результат заключаться в теги <pre> ... </pre>.

В некоторых операционных системах весь список параметров разметки должен быть заключен в двойные кавычки.

пользователь [*/пароль*][*@строка_соединения*]

Информация для регистрации в БД.

/

Устанавливает соединение с локальной БД, при этом аутентификация выполняется средствами операционной системы.

AS (SYSDBA|SYSOPER)

Устанавливает привилегированное соединение для пользователя, имеющего административные права (например, на запуск и останов экземпляра). Если указывается данный параметр, то вся регистрационная информация, как правило, должна заключаться в кавычки:

```
sqlplus "sys/пароль as sysdba"
```

/NOLOG

Запускает SQL*Plus без соединения с БД при запуске.

файл_сценария

Имя файла, содержащего сценарий для SQL*Plus. Исполнив данный сценарий, SQL*Plus завершит работу.

арг1 арг2 арг3

Необязательные аргументы, передаваемые сценарию из командной строки.

Ввод команд в SQL*Plus

Способ ввода команд в SQL*Plus зависит от того, вводите ли вы собственные команды утилиты, команды SQL или PL/SQL-блок.

Ввод команд SQL*Plus

Такие команды, как DESCRIBE, COLUMN, TTITLE, SET и все прочие, перечисленные в разделе «Команды», предназначены для управления самой утилитой SQL*Plus. Они вводятся в одной строке и выполняются сразу после ввода. Например:

```
SET ECHO ON
DESCRIBE employee
```

Допускается наличие точки с запятой в конце команды SQL*Plus. Например:

```
PROMPT Эта точка с запятой не выводится.;
CONNECT system/manager;
```

Вы можете изменить реакцию утилиты на символ точки с запятой, переопределив параметр SQLTERMINATOR.

Длинные команды SQL*Plus могут состоять из нескольких строк. В качестве знака продолжения используется дефис (-) в конце строки, означающий, что команда SQL*Plus будет продолжена на следующей строке. Например, следующие три строки составляют одну команду:

```
COLUMN employee_id -
FORMAT 099999 -
HEADING 'Emp ID'
```

Пробел перед знаком продолжения необязателен. Строковые константы в кавычках также можно переносить, например:

```
SELECT 'Hello-
World!' FROM dual;
```

При переносе строковой константы в нее включаются все пробелы, стоящие перед знаком переноса. Перевод строки также считается пробелом.

Ввод команд SQL

Команды SQL могут располагаться на нескольких строках и всегда должны содержать завершающий символ, в качестве которого может выступать точка с запятой (;) или наклонная черта (/). Например:

```
SELECT user
FROM dual;

SELECT user
FROM dual
/
```

В обоих случаях команда SQL будет сначала помещена в буфер, называемый буфером SQL, а затем выполнена. Команду также можно завершить пустой строкой или точкой, в этом случае она будет сохранена в буфере, но не будет выполняться. Например:

```
SQL> SELECT user
  2  FROM dual
  3
SQL> SELECT user
  2  FROM dual
  3  .
```

Команда SET SQLTERMINATOR позволяет установить вместо точки с запятой другой символ завершения. Посредством команды SET SQLBLANKLINES ON можно разрешить наличие пустых строк в команде SQL. Ввод наклонной черты (/) инициирует выполнение команды, находящейся в буфере.

Ввод PL /SQL-блоков

Блоки PL/SQL могут располагаться на нескольких строках и содержать пустые строки. Они должны завершаться наклонной чертой или точкой на отдельной строке. Например:

```
BEGIN
   DBMS_OUTPUT.PUT_LINE('Hello World!');
END;
/

BEGIN
   DBMS_OUTPUT.PUT_LINE('Hello World!');
```

```
END;
.
```

В случае наклонной черты блок отправляется на сервер для немедленного выполнения. Если ставится точка, то блок только сохраняется в буфере SQL. Команда SET BLOCKTERMINATOR позволяет назначить другой символ завершения блока вместо точки.

Строки в командах SQL*Plus

Многие собственные команды SQL*Plus принимают в качестве параметров строковые значения. Простые строки, не содержащие пробелов и знаков препинания, можно вводить без кавычек, например так:

```
COLUMN employee_id HEADING emp_id
```

Но, как правило, надежнее заключать строки в кавычки. Кавычки могут быть как одинарными, так и двойными. Например:

```
COLUMN employee_id HEADING 'Emp #'
COLUMN employee_id HEADING "Emp #"
```

Если необходимо вставить кавычки в строку, то их надо либо удвоить, либо заключить строку в кавычки другого типа. Две приведенные ниже команды приводят к одинаковым результатам:

```
COLUMN employee_id HEADING '''Emp #'''
COLUMN employee_id HEADING "'Emp #'"
```

Единственное исключение из этого правила относится к команде PROMPT. Все кавычки в этой команде будут выведены без изменений.

Имена файлов

Некоторые команды SQL*Plus позволяют указывать имена файлов. Имя файла может включать путь и/или расширение. Например:

```
SPOOL my_report
SPOOL c:\temp\my_report
SPOOL create_synonyms.sql
```

Большинство команд, работающих с файлами, при отсутствии расширения в имени файла используют расширение файла по умолчанию. Значение по умолчанию зависит от команды.

Именование переменных

SQL*Plus позволяет объявлять переменные двух видов: пользовательские переменные и переменные связывания. Правила именования для них различаются:

Имена пользовательских переменных

Эти имена могут содержать буквы, цифры и символы подчеркивания (_) в любом порядке. Они не чувствительны к регистру, и их длина не должна превышать 30 символов.

Имена переменных связывания

Такие имена должны начинаться с буквы и могут содержать буквы, цифры, символы подчеркивания, доллара ($) и номера (#). Они также не чувствительны к регистру, и их длина ограничена 30 символами.

Форматирование текстовых отчетов

Отчеты SQL*Plus состоят из столбцов. SQL*Plus позволяет задавать заголовки столбцов, формат отображения каждого столбца, верхний и нижний колонтитулы страниц, разделители строк и страниц, а также выполнять подсчет итоговых сумм.

Заголовки столбцов

Заголовки столбцов указываются в инструкции HEADING команды COLUMN:

```
COLUMN employee_name HEADING "имя сотрудника"
```

Текст заголовка может быть заключен в одинарные или двойные кавычки.

Если требуется расположить заголовок на нескольких строках, то в качестве ограничителя строк используют вертикальную черту (|). Например:

```
COLUMN employee_name HEADING "Имя|сотрудника"
```

Заголовки текстовых столбцов выравниваются по левому краю, заголовки числовых столбцов – по правому. Инструкция JUSTIFY изменяет способ выравнивания:

```
COLUMN employee_name HEADING "Имя|сотрудника" -
   JUSTIFY RIGHT
COLUMN employee_name HEADING "Имя|сотрудника" -
   JUSTIFY CENTER
```

Команда SET HEADSEP заменяет вертикальную черту в ограничителе строк другим символом. Команда SET UNDERLINE изменяет признак подчеркивания с дефиса на другой символ.

Форматы столбцов

Инструкция FORMAT команды COLUMN позволяет определить формат отображения столбца. Для числовых полей спецификация формата может быть очень подробной – с указанием длины, количества десятичных разрядов и разделителей. Для полей, содержащих текст и даты, можно указать ширину столбца и необходимость перехода на новую строку. Форматирование различных типов данных рассматривается в разделе «Элементы формата SQL*Plus» этой главы.

Длина и ширина страницы

Шириной страницы управляет команда SET LINESIZE. По умолчанию ширина равна 80 символам. Изменить ее – например, сократить до 60 символов – можно командой:

```
SET LINESIZE 60
```

Значение LINESIZE задается при центрировании и выравнивании по правой границе верхних и нижних колонтитулов.

Длиной страницы управляет команда SET PAGESIZE. По умолчанию страница содержит 24 строки, включая верхний и нижний колонтитулы. Следующая команда устанавливает размер страницы в 50 строк:

```
SET PAGESIZE 50
```

В режиме вывода в HTML (включаемом командой SET MARKUP HTML ON) параметр PAGESIZE управляет количеством строк HTML-таблицы, выводимых до появ-

ления следующего заголовка столбцов; каждая строка таблицы может состоять из одной или нескольких физических строк в зависимости от размера окна броузера.

Нулевое значение параметра PAGESIZE имеет специальное значение в SQL*Plus: оно подавляет вывод верхнего и нижнего колонтитулов и заголовков столбцов.

Колонтитулы

Верхний и нижний колонтитулы определяются командами TTITLE и BTITLE соответственно. Команды имеют одинаковый синтаксис.

Определение колонтитула

В следующем примере определяется многострочный верхний колонтитул, содержащий название компании слева и номер страницы справа:

```
TTITLE LEFT "моя компания " CENTER "Текущая" -
RIGHT "Страница" FORMAT 999 SQL.PNO SKIP 1 -
CENTER "Список сотрудников " SKIP 4
```

Результат имеет такой вид:

```
Моя компания       Текущая        Страница   1
            Список сотрудников
```

Завершающая инструкция SKIP вставляет три пустых строки между верхним колонтитулом и заголовками столбцов.

Отображение даты в заголовке

Для того чтобы вставить в заголовок дату, надо присвоить значение переменной, а затем указать эту переменную в команде BTITLE или TTITLE.

Присваивание текущей даты переменной SQL*Plus можно выполнить следующими командами:

```
SET TERMOUT OFF
COLUMN curdate NEW_VALUE report_date
SELECT TO_CHAR(SYSDATE,'dd-Mon-yyyy') curdate
   FROM DUAL;
SET TERMOUT ON
```

После выполнения этих команд пользовательская переменная REPORT_DATE будет содержать текущую дату. Следующая команда помещает это значение в нижний колонтитул страницы:

```
BTITLE LEFT "Дата составления отчета: " report_date
```

Тем же способом можно вывести в колонтитул и любое другое значение, полученное из базы данных.

Разделители страниц

По умолчанию SQL*Plus помещает между страницами одну пустую строку. Количество строк, указанное в параметре PAGESIZE, в результате добавления этой пустой строки должно равняться фактическому количеству строк, печатаемых принтером на одном листе.

Команда SET PAGESIZE управляет количеством строк на одной странице. Команда SET NEWPAGE определяет действие SQL*Plus при достижении конца страницы. Ко-

личество разделяющих страницы пустых строк можно изменить командой такого вида:

```
SET NEWPAGE 10
```

Команда SET NEWPAGE 0 указывает SQL*Plus, что страницы разделяются одним символом перевода страницы.

В последних версиях SQL*Plus допускается команда SET NEWPAGE NONE, отменяющая вывод между страницами пустых строк и символов перевода страницы.

Разрывы в отчетах

Команды BREAK и COMPUTE могут применяться для определения разрывов строк и страниц, а также итоговых вычислений для отчета. Команда BREAK также позволяет подавить вывод повторяющихся значений столбцов.

Команда BREAK

Команда BREAK позволяет избавиться от повторяющихся значений столбцов следующим образом:

```
SQL> BREAK ON owner

SQL> SELECT owner, table_name
  2  FROM all_tables
  3  ORDER BY owner, table_name;

OWNER      TABLE_NAME
========== ===============
CTXSYS     DR$CLASS
           DR$DELETE
           DR$INDEX
DEMO       CUSTOMER
           DEPARTMENT
           EMPLOYEE
```

Если название столбца приводится в команде BREAK, то SQL*Plus выводит значение данного столбца, только если оно изменяется. Для обеспечения корректной работы не забудьте упорядочить результаты запроса по этому же столбцу.

Кроме того, команда BREAK позволяет пропускать строки или переходить к новой странице при каждом изменении значения. Рассмотрим, например, такие команды:

```
BREAK ON owner SKIP 1
BREAK ON owner SKIP PAGE
```

Первая команда вызовет печать пустой строки при каждом изменении значения owner. Вторая команда приводит к переходу на новую страницу при каждом изменении значения owner.

Можно задать для отчета и несколько разделителей, но только одной и той же командой. Рассмотрим пример, в котором происходит переход на новую страницу при каждом изменении значения owner, а при каждом изменении типа объекта выводится пустая строка:

```
BREAK ON owner SKIP PAGE ON object_type SKIP 1
SELECT owner, object_type, object_name
   FROM dba_objects
ORDER BY owner, object_type, object_name;
```

Прежде чем выполнить действия по разбиению для столбца, SQL*Plus сначала выполнит такие действия для всех внутренних столбцов. Поэтому изменение поля owner приведет сначала к пропуску строки, а затем – к переходу на новую страницу.

COMPUTE команда

Команда COMPUTE указывает SQL*Plus на необходимость вычисления итоговых значений для группы записей. COMPUTE всегда используется совместно с BREAK. Например, для подсчета количества таблиц, принадлежащих каждому из пользователей, вы можете сделать следующее:

```
BREAK ON owner
COMPUTE COUNT OF table_name ON owner
SELECT owner, table_name
   FROM dba_tables
ORDER BY owner, table_name;
```

SQL*Plus подсчитывает количество названий таблиц для каждого пользователя и выводит результаты тогда, когда происходит разрыв по полю owner.

Можно вычислять итоги для нескольких столбцов сразу, используя несколько команд COMPUTE.

Имейте в виду, что порядок вывода (порядок, использованный в списке SELECT) не обязательно должен совпадать с порядком сортировки или порядком разбиения.

Элементы формата SQL*Plus

Команды COLUMN, ACCEPT, SET NUMBER, TTITLE, BTITLE, REPHEADER и REPFOOTER позволяют управлять форматами данных посредством спецификации формата. *Спецификация формата* – это символьная строка, которая сообщает SQL*Plus о том, как именно следует форматировать число, дату или текст при их отображении.

Форматирование чисел

Полный список элементов форматирования, применяемых для форматирования числового вывода, приведен в табл. C.1 приложения C. Это приложение также содержит целый ряд примеров, иллюстрирующих применение элементов форматирования чисел.

Форматирование символьных строк

Для форматирования символьных строк применяется всего один элемент – A, за которым следует число, определяющее ширину столбца в символах. Например:

```
SQL> COLUMN a FORMAT A40

SQL> SELECT 'An apple a day keeps the doctor away.' A
  2    FROM dual;

A
----------------------------------------
An apple a day keeps the doctor away.
```

По умолчанию длинные текстовые значения внутри столбца переносятся. Для того чтобы указать, требуются ли переносы и какие именно, существуют параметры WORD_WRAPPED, WRAPPED и TRUNCATED команды COLUMN. Например:

```
SQL> COLUMN a FORMAT A18 WORD_WRAPPED

SQL> SELECT 'An apple a day keeps the doctor away.' A
  2    FROM dual;

A
------------------
An apple a day
keeps the doctor
away.
```

Если текст с переносами занимает несколько строк, то SQL*Plus выводит вслед за записью пустую строку, называемую *разделителем записей*. Такое поведение можно отменить посредством команды SET RECSEP OFF.

Если формат задается в команде ACCEPT, то он указывает максимальное количество символов, которое SQL*Plus будет принимать от пользователя.

Форматирование дат

Полный список элементов форматирования, применяемых для вывода дат, приведен в таблице B.1 приложения B. В приложении приведены разнообразные примеры, иллюстрирующие применение элементов форматирования дат, а также перечень идентификаторов часовых поясов.

Для преобразования значений дат в символьные строки применяются приведенные в приложении B элементы форматирования дат в сочетании со встроенной функцией Oracle TO_CHAR. Например:

```
SQL> SELECT TO_CHAR(SYSDATE, 'dd-Mon-yyyy hh:mi:ss PM')
  3    FROM dual;

TO_CHAR(SYSDATE,'DD-MON
-----------------------
13-Dec-1999 09:13:59 PM
```

Если формат задается в команде ACCEPT, это означает, что пользователю необходимо вводить даты в указанном формате.

Команды

В разделе приводится список всех команд в алфавитном порядке с краткими описаниями. Значения параметров, которые выделены жирным шрифтом, являются значениями по умолчанию.

/*...*/

```
/* текст_комментария текст_комментария текст_комментария */
```

Разделители /* и */ открывают и закрывают комментарии в SQL*Plus. Введенный таким образом комментарий может занимать несколько строк. Если в файле сценария указать комбинацию /*...*/, то при выполнении сценария комментарии будут отображаться на экране.

-- (двойной дефис)

```
--текст_комментария
```

Двойной дефис (--) позволяет поместить в сценарий SQL*Plus однострочный комментарий.

@ (коммерческое «at»)

```
@файл_сценария [аргумент...]
```

Применяется для выполнения файла сценария SQL*Plus.

Параметры

файл_сценария

Имя или URL (в версии Oracle9*i* и выше) исполняемого файла. В имя файла можно включить путь к нему. Если путь не указан, то SQL*Plus сначала обратится к текущему каталогу, а затем последует по пути поиска SQL*Plus. По умолчанию файл сценария имеет расширение *.sql*.

В Oracle9*i* Release 1 только Windows-версия SQL*Plus позволяет идентифицировать файл сценария по его URL. В Release 2 эта функциональность поддерживается уже всеми версиями SQL*Plus. Адреса URL могут относиться к протоколу HTTP или FTP. В утилите *i*SQL*Plus в данной команде можно указывать только URL; задать имя файла нельзя.

аргумент

Аргумент, передаваемый сценарию. Количество аргументов не ограничено (их может быть столько, сколько необходимо сценарию). Аргументы должны быть отделены друг от друга хотя бы одним пробелом.

@@ (двойное коммерческое «at»)

```
@@файл_сценария [аргумент...]
```

Применяется внутри файла сценария для выполнения другого файла сценария из того же каталога, что и первый. Описание параметров приведено в разделе для команды @.

/ (наклонная черта)

Выполняет команду SQL или блок PL/SQL, которые в настоящий момент находятся в буфере.

ACCEPT

```
ACC[EPT] пользовательская_переменная [NUM[BER] | CHAR | DATE]
    [FOR[MAT] спецификация_формата]
    [DEF[AULT] значение_по_умолчанию]
    [PROMPT текст_приглашения | NOPR[OMPT]] [HIDE]
```

Принимает вводимые пользователем данные. Команда ACCEPT не поддерживается в *i*SQL*Plus. В последних версиях SQL*Plus синтаксис команды ACCEPT был значи-

тельно изменен и расширен. Приведенное описание справедливо для версий Oracle8*i* и выше. В более ранних версиях доступны не все инструкции.

Параметры

пользовательская_переменная

Имя переменной, которую необходимо определить.

NUM[*BER*] | ***CHAR*** | *DATE*

Указывает тип ожидаемых данных.

FOR[*MAT*] *спецификация_формата*

Указывает спецификацию формата, которая может быть заключена в кавычки.

DEF[*AULT*] *значение_по_умолчанию*

Приводит значение по умолчанию, присваиваемое переменной.

PROMPT текст_приглашения

Задает текст приглашения, выводимого для пользователя.

***NOPR*[*OMPT*]**

Указывает, что не следует отображать приглашение на ввод для пользователя.

HIDE

Вынуждает SQL*Plus не отображать ответ пользователя на мониторе. Это полезно в случае, если у пользователя запрашивается пароль.

APPEND

```
A[PPEND] текст
```

Позволяет добавить текст в конец текущей строки буфера SQL. APPEND – это команда редактирования и не поддерживается в *i*SQL*Plus. Для того чтобы добавляемая строка начиналась с пробела, необходимо ввести после команды APPEND два пробела.

ARCHIVE LOG

```
ARCHIVE LOG {LIST | STOP | START [TO направление] |
    NEXT [TO направление] | ALL [TO направление] |
    порядковый_номер_журнала [TO направление]}
```

Управляет архивированием журналов или выводит информацию о таком архивировании. Выполнять эту команду могут лишь пользователи, зарегистрировавшиеся как SYSDBA, SYSOPER или INTERNAL.

Параметры

LIST

Указывает SQL*Plus на необходимость вывода информации о текущем состоянии архивирования.

STOP

Останавливает автоматическую архивацию журнальных файлов.

START

Включает автоматическую архивацию журнальных файлов.

NEXT

Вручную архивирует следующую по порядку журнальную группу, если она заполнена. Порядковый номер файла можно узнать посредством команды ARCHIVE LOG LIST.

ALL

Вручную архивирует все уже заполненные, но еще не заархивированные журнальные группы.

порядковый_номер_журнала

Вручную архивирует указанную журнальную группу при условии, что она еще доступна.

направление

Определяет место хранения архивированного журнального файла. В команде ARCHIVE LOG START указывает место архивирования всех журнальных файлов. Если применяется для NEXT, ALL или определенного порядкового номера, то определяет место хранения файлов, архивируемых только данной командой. Если не указать направление в команде ARCHIVE LOG START, то будет взято значение параметра инициализации LOG_ARCHIVE_DEST.

ATTRIBUTE

```
ATTRIBUTE [объектный_тип.атрибут | псевдоним_атрибута [ALI[AS] псевдоним |
    CLE[AR]| FOR[MAT] спецификация_формата |
    LIKE исходный_атрибут |
    ON | OFF...]]
```

Форматирует атрибуты объектного типа Oracle. Если в команде ATTRIBUTE нет параметров, то будет выведен список всех текущих параметров для атрибутов. Инструкции ALIAS, CLEAR, FORMAT, LIKE, ON и OFF действуют так же, как для команды COLUMN.

Параметры

объектный_тип

Имя объектного типа Oracle.

атрибут

Имя атрибута указанного объектного типа, который форматируется. Если не задавать больше никаких параметров, то будут выведены текущие параметры для данного атрибута.

BREAK

```
BRE[AK] [ON (имя_столбца | ROW | REPORT)
    [SKI[P] (пропускаемые_строки | PAGE) |
    NODUP[LICATES] | DUP[LICATES]...]...]
```

Определяет разрывы строк и страниц в зависимости от изменения значений столбцов отчета. Управляет выводом повторяющихся значений и выводом вычисленных значений, таких как окончательные и промежуточные итоги. Выполнение команды BREAK без параметров приводит к выводу текущих параметров разрывов отчетов SQL*Plus.

Параметры

имя_столбца

Указывает имя обрабатываемого столбца. Если значение в данном столбце изменяется, то SQL*Plus выполняет предписанные действия по разрыву.

ROW

Вынуждает SQL*Plus осуществлять разрыв после каждой строки.

REPORT

Задает разрыв на уровне отчета и вынуждает SQL*Plus выводить общие итоги в конце отчета. Если используется разрыв на уровне отчета, то инструкция SKIP PAGE будет проигнорирована, но (что достаточно странно) можно будет пропускать строки при разбиении отчета.

SKI[*P*] *пропускаемые_строки*

Указывает SQL*Plus, что следует пропустить указанное количество строк при разрыве.

SKI[*P*] *PAGE*

Указывает SQL*Plus, что при разрыве следует перейти на новую страницу.

***NODUP*[*LICATES*]**

Указывает SQL*Plus, что следует выводить значение столбца, только если оно изменилось. По умолчанию, если для столбца применяется разрыв, поведение SQL*Plus будет именно таким.

DUP[*LICATES*]

Указывает SQL*Plus, что следует выводить значение столбца в каждой строке отчета вне зависимости от того, отличается ли оно от значения, выведенного для предыдущей записи.

BTITLE

```
BTI[TLE] [[OFF | ON] |
    COL x | S[KIP] x | TAB x | LE[FT] | CE[NTER] | R[IGHT] |
    BOLD | FOR[MAT] спецификация_формата | текст | переменная...]
```

Работает так же, как команда TTITLE, только BTITLE определяет нижний колонтитул страницы, а не верхний. Описание параметров приведено в описании команды TTITLE.

CHANGE

```
C[HANGE] /старый_текст[/[новый_текст[/]]]
```

Позволяет осуществить операции поиска и замены для текущей строки в буфере SQL. CHANGE – это команда редактирования. Ее можно применять не только для изменения текста, но и для его удаления. CHANGE не поддерживается в *i*SQL*Plus.

Параметры

старый_текст

Текст, подлежащий изменению или удалению.

новый_текст

Текст замены.

/

Общепринятое обозначение разделения строк нового и старого текста, однако эту функцию может выполнять и любой другой символ, не являющийся буквой или цифрой, он лишь должен быть одним и тем же для всей команды.

CLEAR

```
CL[EAR] (BRE[AKS] | BUFF[ER] | COL[UMNS] | COMP[UTES] |
    SCR[EEN] | SQL | TIMI[NG])
```

Позволяет быстро удалить все определения столбцов, параметры разрывов, определения вычислений и т.д.

Параметры

BRE[*AKS*]

Удаляет любые параметры разрывов, которые могли быть определены командой BREAK.

BUFF[*ER*]

Очищает содержимое буфера.

COL[*UMNS*]

Удаляет любые определения столбцов, которые могли быть сделаны командой COLUMN.

COMP[*UTES*]

Удаляет любые вычисления, которые могли быть определены командой COMPUTE.

SCR[*EEN*]

Очищает экран. Не поддерживается в *i*SQL*Plus.

SQL

Очищает содержимое буфера SQL.

TIMI[*NG*]

Удаляет любые таймеры, которые могли быть созданы командой TIMING.

COLUMN

```
COL[UMN] [имя_столбца [ALI[AS] псевдоним |
    CLE[AR] | ENTMAP (ON|OFF) |
    FOLD_A[FTER] | FOLD_B[EFORE] |
    FOR[MAT] спецификация_формата | HEA[DING] текст_заголовка |
    JUS[TIFY] (LEFT | CENTER | CENTRE | RIGHT) |
    LIKE имя_исходного_столбца  | NEWL[INE] |
    NEW_V[ALUE] пользовательская_переменная | NOPRI[NT] |
    PRI[NT] | NUL[L] null_текст |
    OLD_V[ALUE] пользовательская_переменная | ON | OFF |
    TRU[NCATED] | WOR[D_WRAPPED] | WRA[PPED]...]]
```

Форматирует вывод для отчетов, состоящих из столбцов. Команда COLUMN без параметров выводит список всех текущих форматов столбцов.

Команда COLUMN является кумулятивной – две команды, указывающие два разных параметра для одного поля эквиваленты одной команде, в которой задаются оба параметра.

Параметры

имя_столбца

Имя форматируемого столбца. Если это вычисляемый столбец, то именем будет выражение. Если в команде SELECT указан псевдоним для данного столбца, то этот псевдоним должен быть здесь. Если в команде COLUMN *имя_столбца* не указаны другие параметры, то SQL*Plus выведет текущий формат данного столбца.

ALI[*AS*]

Позволяет указать альтернативное имя для столбца, являющееся более значимым.

псевдоним

Альтернативное имя для столбца, которое может задаваться в командах BREAK, COMPUTE и других командах COLUMN.

CLE[*AR*]

Удаляет все параметры форматирования данного столбца.

ENTMAP (*ON* | *OFF*)

Определяет, должны ли такие символы, как < и >, отображаться в HTML-отчетах как < и >. По умолчанию действует значение параметра ENTMAP, которое было определено командой SET MARKUP или параметром командной строки –M.

FOLD_A[*FTER*]

Указывает SQL*Plus, что следует перейти на новую строку после вывода данного столбца.

FOLD_B[*EFORE*]

Указывает SQL*Plus, что следует перейти на новую строку, прежде чем выводить данный столбец.

FOR[*MAT*]

Позволяет управлять выводом данных для столбца.

спецификация_формата

Строка, указывающая формат отображения столбца.

HEA[*DING*]

Позволяет указать заголовок для столбца.

текст_заголовка

Текст, который будет виден в заголовке столбца. Может быть заключен в одинарные или двойные кавычки.

JUS[*TIFY*] (*LEFT* | *CENTER* | *CENTRE* | *RIGHT*)

Регулирует вывод текста заголовка по отношению к ширине столбца. По умолчанию для числовых полей заголовки выравниваются по левому краю, а для текстовых полей – по правому. Данный параметр позволяет изменить поведение по умолчанию.

LIKE

Указывает, что столбец должен иметь те же атрибуты форматирования, что и другой столбец.

имя_исходного_столбца

Имя исходного столбца для параметра LIKE.

NEWL[*INE*]

Указывает SQL*Plus, что следует перейти на новую строку, прежде чем печатать отчет.

NEW_V[*ALUE*]

Указывает SQL*Plus, что следует обновить значение пользовательской переменной текущим значением столбца.

пользовательская_переменная

Имя пользовательской переменной для параметров NEW_VALUE и OLD_VALUE.

NOPRI[*NT*]

Указывает SQL*Plus, что столбец выводить не надо.

***PRI*[*NT*]**

Разрешает вывод столбца.

NUL[*L*]

Позволяет указать текст, который будет отображаться, если столбец содержит значение NULL.

null_текст

Текст, который будет отображаться, если данный столбец содержит значение NULL.

OLD_V[*ALUE*]

Указывает SQL*Plus, что следует обновить значение пользовательской переменной предыдущим значением столбца.

ON

Указывает SQL*Plus, что следует выводить столбец в сответствии с заданным форматом. Это поведение по умолчанию.

OFF

Отключает форматирование столбца.

TRU[*NCATED*]

Указывает SQL*Plus, что следует усекать текст до ширины столбца. Длинные значения не переносятся.

WOR[*D_WRAPPED*]

Указывает SQL*Plus, что следует переносить длинные значения столбцов по словам.

***WRA*[*PPED*]**

Указывает SQL*Plus, что следует переносить длинные значения столбцов. Строка разрывается точно на границе столбца, даже если речь идет о середине слова.

COMPUTE

```
COMP[UTE] [(AVG | COU[NT] | MAX[IMUM] | MIN[IMUM] | NUM[BER] |
    STD | SUM | VAR[IANCE])...  [LABEL текст_метки] OF имя_столбца...
    ON (имя_группируемого_столбца | ROW | REPORT)...]
```

Определяет правила вычисления выводимых в отчете итоговых значений. COMPUTE можно применять в сочетании с командой BREAK для вычисления и вывода итого-

вых значений для столбцов, средних значений, минимумов и максимумов и т. д. Вычисления выполняются SQL*Plus по мере формирования отчета.

Команда COMPUTE без параметров перечисляет все вычисления, определенные на текущий момент.

Параметры

AVG

Вычисляет среднее значение для всех отличных от NULL значений числового столбца.

COU[*NT*]

Вычисляет общее количество отличных от NULL значений столбца.

MAX[*IMUM*]

Вычисляет максимальное значение, содержащееся в столбце. Применяется к столбцам типа NUMBER, CHAR, VARCHAR2, NCHAR и NVARCHAR2.

MIN[*IMUM*]

Вычисляет минимальное значение, содержащееся в столбце. Применяется к столбцам типа NUMBER, CHAR, VARCHAR2, NCHAR и NVARCHAR2.

NUM[*BER*]

Работает аналогично COUNT, но подсчитывает все значения, включая NULL.

STD

Вычисляет стандартное отклонение для всех отличных от NULL значений числового столбца.

SUM

Вычисляет сумму всех отличных от NULL значений числового столбца.

VAR[*IANCE*]

Вычисляет дисперсию всех отличных от NULL значений числового столбца.

LABEL

Позволяет указать метку для вычисляемого значения, которая будет напечатана слева от вычисленного значения.

текст_метки

Текст, который будет отображаться при выводе вычисленного значения.

имя_столбца

Имя столбца, для которого вычисляются сводные данные. Если это вычисляемый столбец, то именем является выражение. Если в команде SELECT указан псевдоним для столбца, то этот псевдоним должен быть здесь.

имя_группируемого_столбца

Указывает SQL*Plus, что следует заново выполнять вычисление при каждом изменении данного столбца.

ROW

Приводит к тому, что вычисление выполняется один раз для каждой строки, возвращенной запросом.

REPORT

Приводит к тому, что вычисление выполняется в конце отчета и включает в себя значения из всех строк. REPORT применяется для получения общих итогов.

CONNECT

```
CONN[ECT] [имя_пользователя[/пароль][@строка_соединения] | / ]
    [AS (SYSOPER | SYSDBA)] | [INTERNAL]
```

Позволяет изменить соединение с базой данных, зарегистрироваться под другим именем или подключиться к БД в административном режиме.

Параметры

имя_пользователя

Имя пользователя БД.

пароль

Пароль для доступа к БД.

строка_соединения

Идентификатор соединения, сообщающий SQL*Plus, к какой БД вы хотите подключиться.

/

Наклонная черта указывается вместо имени пользователя, пароля и строки соединения в том случае, если требуется, чтобы при подключении к локальной базе данных аутентификация выполнялась средствами операционной системы.

AS

Сообщает SQL*Plus, что пользователь подключается с административной ролью.

SYSOPER

Сообщает SQL*Plus, что пользователь подключается как оператор.

SYSDBA

Сообщает SQL*Plus, что пользователь подключается как администратор БД.

INTERNAL

Сообщает SQL*Plus, что пользователь подключается как внутренний. Начиная с версии Oracle9*i* данный параметр более не поддерживается.

COPY

```
COPY { FROM соединение | TO соединение}
    (APPEND | CREATE | INSERT | REPLACE)
    таблица_назначения [(список_столбцов)] USING команда_select
```

Позволяет применять SQL*Plus как средство передачи данных от одной БД Oracle к другой.

Применять команду COPY теперь уже не рекомендуется. Она не умеет обрабатывать новые типы данных, появившиеся в версиях Oracle8 и выше, такие как объектные типы.

Параметры

FROM/TO

Задает базу данных назначения или исходную БД. К одной из баз данных вы должны быть уже подключены. Эта инструкция применяется для указания второй БД.

соединение

Информация о регистрации, необходимая для соединения с другой БД. Она должна быть представлена в стандартном формате *имя_пользователя/пароль@строка_соединения*.

APP[*END*]

Указывает SQL*Plus, что следует вставить скопированные строки в таблицу назначения. При необходимости SQL*Plus сначала создаст таблицу назначения.

CRE[*ATE*]

Указывает SQL*Plus, что следует копировать данные, только если таблица назначения – это новая таблица.

INSERT

Указывает SQL*Plus, что следует копировать данные, только если таблица назначения уже существует.

REP[*LACE*]

Указывает SQL*Plus, что следует удалить и создать заново таблицу назначения, если она существует в настоящий момент.

таблица_назначения

Имя таблицы, в которую предполагается копировать данные.

список_столбцов

Определяет имена столбцов для случая, если команда COPY будет создавать новую таблицу назначения. Имена в списке разделяются запятыми; их количество должно совпадать с количеством столбцов в списке SELECT.

команда_select

Команда SELECT, которая возвращает данные, подлежащие копированию.

DEFINE

```
DEF[INE] [имя_переменной [= текст]]
```

Позволяет создать пользовательскую переменную (или переменную подстановки) и присвоить ей значение или же вывести значения одной или нескольких переменных.

Параметры

имя_переменной

Имя создаваемой переменной. Если в команде указать только имя переменной, то SQL*Plus выведет текущее значение данной переменной, если она существует.

текст

Текст, который следует присвоить переменной.

DEL

```
DEL [(n | * | LAST)[ (e | * | LAST)]]
```

Применяется для удаления одной или нескольких строк из буфера. DEL – это команда редактирования, она не поддерживается в *i*SQL*Plus.

Параметры

b Номер строки, идентифицирующий удаляемую строку, или же начало диапазона удаляемых строк.

e Номер строки, задающий конец диапазона удаляемых строк.

* Ссылка на текущую строку.

LAST Ссылка на последнюю строку в буфере.

DESCRIBE

```
DESC[RIBE] [схема.]имя_объекта[@имя_канала_связи_бд]
```

Выводит информацию о таблице, представлении, объектном типе, хранимом пакете, хранимой процедуре, хранимой функции или синониме (см. также описание команды SET DESCRIBE).

Параметры

схема

Имя владельца объекта.

имя_объекта

Имя объекта, описание которого требуется получить.

имя_канала_связи_бд

Имя канала связи, указывающего на базу данных, в которой существует объект.

DISCONNECT

```
DISC[ONNECT]
```

Закрывает соединение с базой данных без завершения сеанса SQL*Plus.

EDIT

```
ED[IT] [имя_файла]
```

Позволяет вызвать внешний редактор для редактирования содержимого буфера SQL (при выполнении команды без параметра) или для редактирования содержимого файла операционной системы (если указано имя данного файла). EDIT не поддерживается в *i*SQL*Plus.

Команда, выполняющая вызов внешнего редактора, содержится в пользовательской переменной _EDITOR. Значение этой переменной можно изменить посредством команды DEFINE.

EXECUTE

```
EXEC[UTE] команда
```

Позволяет выполнить отдельную команду PL/SQL.

EXIT

```
EXIT [SUCCESS | FAILURE | WARNING | значение | пользовательская_переменная |
    :переменная_связывания] [COMMIT | ROLLBACK]
```

Завершает сеанс SQL*Plus и возвращает пользователя в операционную систему.

Параметры

SUCCESS

Возвращает статус, соответствующий нормальному завершению (поведение по умолчанию).

FAILURE

Возвращает статус ошибки.

WARNING

Возвращает статус предупреждения.

значение

Возвращает в качестве статуса произвольное значение.

пользовательская_переменная

Возвращает в качестве статуса значение указанной пользовательской переменной.

:переменная_связывания

Возвращает в качестве статуса значение указанной переменной связывания.

COMMIT

Вынуждает SQL*Plus выполнить фиксацию перед выходом (поведение по умолчанию).

ROLLBACK

Вынуждает SQL*Plus откатить все открытые транзакции перед выходом.

GET

```
GET имя_файла [LIS[T] | NOL[IST]]
```

Читает SQL-команду из файла и загружает ее в буфер. GET не поддерживается в *i*SQL*Plus.

Параметры

имя_файла

Имя файла, содержащего загружаемую SQL-команду.

LI[ST]

Указывает SQL*Plus, что следует отобразить буфер после загрузки файла (поведение по умолчанию).

NOL[IST]

Указывает SQL*Plus, что следует загрузить файл без его отображения.

HELP

```
HELP [раздел]
```

Выводит справочную информацию о командах SQL*Plus. В версиях, предшествовавших Oracle8*i*, HELP также предоставляла сведения о синтаксисе SQL и PL/SQL. Обратите внимание на то, что некоторые Windows-версии SQL*Plus, особенно до Oracle9*i*, не поддерживают оперативную справку.

Параметры

раздел

Искомый раздел справочной информации. Если ввести HELP INDEX (или HELP MENU в некоторых старых версиях), то будет выведен полный список имеющихся разделов.

HOST

```
HO[ST] [команда_ос]
```

Позволяет выполнить команду операционной системы или вызвать интерпретатор команд для выполнения нескольких таких команд. Команда HOST без указания команды ОС приведет к выводу приглашения на ввод команд ОС. Для того чтобы вернуться в SQL*Plus, обычно выполняется команда выхода операционной системы. Команда HOST не поддерживается в *i*SQL*Plus.

INPUT

```
I[NPUT] [текст]
```

Вставляет одну или несколько строк текста в буфер. Строки вставляются вслед за текущей. По команде INPUT без указания текста SQL*Plus переведет вас в режим вставки, позволяющий вводить любое количество строк. Команда INPUT не поддерживается в *i*SQL*Plus.

LIST

```
L[IST] [(b | * | LAST)[ (e | * | LAST)]]
```

Применяется для отображения текущей строки буфера. LIST – это команда редактирования. Если выдать просто команду LIST, то SQL*Plus выведет все строки буфера.

Параметры

b Номер строки, указывающий начало диапазона отображаемых строк. Если конечный номер не указан, то будет выведена лишь одна указанная строка.

e Номер строки, указывающий конец диапазона отображаемых строк.

* Ссылка на номер текущей строки.

LAST Ссылка на последнюю строку буфера.

PASSWORD

```
PASSW[ORD] [имя_пользователя]
```

Позволяет изменить пароль в Oracle при помощи SQL*Plus. Команда PASSWORD не поддерживается в *i*SQL*Plus. После ввода команды у пользователя будет запрошен его старый пароль, после чего надо будет дважды ввести новый пароль.

Параметр

имя_пользователя

Имя пользователя, пароль которого предполагается изменить. По умолчанию изменяется собственный пароль. Для изменения чужого пароля необходимо обладать привилегией ALTER USER.

PAUSE

```
PAU[SE] [сообщение_паузы]
```

Указывает SQL*Plus, что следует вывести соответствующее сообщение, а затем приостановить работу. Для того чтобы продолжить работу, пользователь должен будет нажать клавишу Enter. Команда PAUSE не поддерживается в *i*SQL*Plus.

PRINT

```
PRI[NT] [имя_переменной_связывания]
```

Выводит значение переменной связывания.

Параметр

имя_переменной_связывания

Имя переменной связывания, значение которой требуется вывести. Если имя не указано, то будут выведены значения всех переменных связывания.

PROMPT

```
PRO[MPT] выводимый_текст
```

Выводит сообщение, видимое пользователю.

Параметр

выводимый_текст

Текст, который требуется отобразить для пользователя. Заключать строку в кавычки не надо. Если строка содержит кавычки, они будут выведены.

QUIT

```
QUIT FAILURE ROLLBACK QUIT [SUCCESS | FAILURE | WARNING |
    значение | пользовательская_переменная | :переменная_связывания] |
    [COMMIT | ROLLBACK]
```

Аналогична команде EXIT. Описание параметров приведено в описании команды EXIT.

RECOVER

Инициирует восстановление носителя для базы данных, табличного пространства или файла данных. Выполнять команду могут лишь пользователи, зарегистрировавшиеся как SYSDBA, SYSOPER или INTERNAL (INTERNAL не поддерживается начиная

с версии Oracle9*i*). Команда RECOVER не поддерживается в *i*SQL*Plus. Функции, реализуемые RECOVER, также может выполнять Recovery Manager (см. главу 15).

Рекомендуем выполнять команду RECOVER только тем пользователям, которые уверены, что полностью представляют себе процесс восстановления базы данных.

Синтаксис команды RECOVER часто меняется. Рассматриваемый далее синтаксис относится к версии Oracle9*i* Release 2:

```
RECOVER {общее | управляемое | END BACKUP}

общее ::=
   [AUTOMATIC] [FROM каталог]
   { {полное_восстановление_бд
     |частичное_восстановление_бд
     |LOGFILE имя_файла}
     [параметр_восстановления [параметр_восстановления...]]
   |CONTINUE [DEFAULT] | CANCEL}

полное_восстановление_бд ::=
   [STANDBY] DATABASE
   [UNTIL {CANCEL | TIME дата_и_время | CHANGE scn}]
   [USING BACKUP CONTROLFILE]
или
   [STANDBY] DATABASE
   [USING BACKUP CONTROLFILE]
   [UNTIL {CANCEL | TIME дата_и_время | CHANGE scn}]

частичное_восстановление_бд ::=
   {TABLESPACE табличное_пространство [, табличное_пространство]...
   | DATAFILE имя_файла_данных [, имя_файла_данных]...
   | STANDBY {TABLESPACE табличное_пространство [, табличное_пространство]...
             | DATAFILE имя_файла_данных [, имя_файла_данных]...}
   UNTIL [CONSISTENT] [WITH] CONTROLFILE }

параметр_восстановления ::=
     {TEST | ALLOW блоки CORRUPTION
     | PARALLEL [степень] | NOPARALLEL}

управляемое ::=
   MANAGED STANDBY DATABASE
   [ {NODELAY | [TIMEOUT] минуты
     | CANCEL [IMMEDIATE] [NOWAIT]}
   | [DISCONNECT [FROM SESSION] ] [FINISH [NOWAIT] ] ]
```

Параметры

AUTOMATIC

Указывает серверу Oracle, что следует автоматически определять имена журнальных файлов, которые будут использованы при восстановлении.

FROM каталог

Задает каталог, в котором находятся архивные журнальные файлы.

LOGFILE имя_файла

Начинает восстановление с использованием указанного архивного журнального файла.

TEST

Выполняет тестовое восстановление, при котором журнал считывается и применяется в оперативной памяти, но сами файлы БД остаются нетронутыми.

ALLOW блоки CORRUPTION

Указывает количество поврежденных блоков, которое допускается в журнальных файлах, применяемых при восстановлении. При нормальном восстановлении значение не должно превышать единицу.

PARALLEL [*степень*]

Указывает, что восстановление выполняется параллельно; дополнительно может быть задана необходимая степень параллелизма. По умолчанию степень параллелизма равна количеству процессоров, доступных всем экземплярам, умноженному на значение параметра инициализации PARALLEL_THREADS_PER_CPU.

NOPARALLEL

Указывает, что восстановление выполняется последовательно.

CONTINUE [*DEFAULT*]

Продолжает прерванное многоэкземплярное восстановление. Команда CONTINUE DEFAULT позволяет серверу Oracle определить, какой журнальный файл применить следующим.

CANCEL

Завершает восстановление, прерываемое командой CANCEL.

STANDBY DATABASE

Восстанавливает резервную базу данных, руководствуясь содержимым управляющего и архивного журнальных файлов основной базы данных.

DATABASE

Начинает восстановление носителя для всей БД. База данных должна быть смонтирована, но не открыта.

TABLESPACE табличное_пространство

Начинает восстановление носителя для указанного табличного пространства или списка табличных пространств (максимальное количество равно 16). Табличное пространство должно находиться в автономном режиме, база данных должна быть смонтирована и открыта.

DATAFILE имя_файла_данных

Начинает восстановление носителя для указанного файла данных или списка файлов данных. Восстанавливаемые файлы данных должны находиться в автономном режиме. Если ни один из файлов данных не относится к табличному пространству SYSTEM, то база данных может оставаться открытой.

STANDBY TABLESPACE табличное_пространство

Восстанавливает табличные пространства резервной базы данных.

STANDBY DATAFILE имя_файла_данных

Восстанавливает файлы данных резервной базы данных.

UNTIL CANCEL

Позволяет восстанавливать по одному журнальному файлу за раз, при этом после обработки каждого журнального файла можно отменить операцию.

UNTIL CHANGE scn

Выполняет неполное восстановление на основе системного номера SCN. Имейте в виду, что транзакция с указанным номером не восстанавливается.

UNTIL TIME дата_и_время

Выполняет восстановление, регулируемое по времени. Восстанавливаются все транзакции, завершенные до указанного времени.

USING BACKUP CONTROLFILE

Вынуждает использовать при восстановлении управляющий файл резервного копирования.

UNTIL CONSISTENT WITH CONTROLFILE

Восстанавливает резервную базу данных, применяя управляющий файл резервной БД.

MANAGED STANDBY DATABASE

Переводит резервную базу данных в режим непрерывного восстановления.

NODELAY

Применяет архивные журнальные файлы к резервной базе данных без задержки. Подменяет настройки, сделанные для времени задержки в LOG_ARCHIVE_DEST.

TIMEOUT минуты

Определяет тайм-аут, по прошествии которого восстановление резервной базы данных завершается, если нет журнального файла, который можно было бы применить.

CANCEL [*IMMEDIATE*] [*NOWAIT*]

Завершает восстановление резервной базы данных после применения текущего архивного журнального файла. IMMEDIATE применяется для завершения восстановления при чтении следующего журнального файла. NOWAIT работает так же, но незамедлительно возвращает управление пользователю.

DISCONNECT [*FROM SESSION*]

Создает фоновый процесс для применения журнала к резервной базе данных, с тем чтобы можно было выполнять другие операции в рамках текущей сессии.

FINISH [*NOWAIT*]

Используется в случае сбоя основной базы данных, для того чтобы закончить применение всех журнальных файлов основной БД к резервной базе данных. По умолчанию команда ожидает восстановления для того, чтобы закончить работу. NOWAIT позволяет сразу же вернуть управление.

REMARK

```
REM[ARK] текст_комментария
```

Применяется для помещения комментария в сценарий SQL*Plus.

REPFOOTER

```
REPF[OOTER] [OFF | ON] |
    [COL x | S[KIP] x | TAB x |
    LE[FT] | CE[NTER] | R[IGHT] | BOLD |
    FOR[MAT] спецификация_формата | текст | переменная...]
```

Определяет нижний колонтитул отчета. Нижние колонтитулы выводятся на последней странице отчета – за последней строкой подробной информации и перед нижним колонтитулом страницы. Описания параметров приведены в разделе для команды TTITLE.

REPHEADER

```
REPH[EADER] [OFF  | ON] |
    [COL x | S[KIP] x | TAB x |
    LE[FT] | CE[NTER] | R[IGHT] | BOLD |
    FOR[MAT] спецификация_формата | текст | переменная...]
```

Определяет заголовок отчета. Заголовки отчета выводятся на первой странице отчета – после верхнего колонтитула страницы и перед первой строкой детальной информации. Описания параметров приведены в разделе для команды TTITLE.

RUN

```
R[UN]
```

Выводит, а затем исполняет текущую команду из буфера SQL.

SAVE

```
SAV[E] имя_файла [CRE[ATE] | REP[LACE] | APP[END]]
```

Записывает содержимое буфера SQL в файл операционной системы. Команда SAVE не поддерживается в *i*SQL*Plus.

Параметры

имя_файла

Имя файла, в который будет записано содержимое буфера. По умолчанию имеет расширение *.sql*.

***CRE*[*ATE*]**

Приводит к тому, что операция выполняется, только если файл ранее не существовал. Это поведение по умолчанию.

REP[*LACE*]

Перезаписывает любой существующий файл с тем же именем.

APP[*END*]

Дописывает содержимое буфера в файл.

SET

Позволяет устанавливать параметры SQL*Plus в соответствии с потребностями пользователя. Далее приведены разнообразные команды семейства SET.

Команды SET

SET APPI[*NFO*] (*ON* | *OFF* | *текст*)

Управляет автоматической регистрацией командных файлов при помощи пакета DBMS_APPLICATION_INFO.

SET ARRAY[*SIZE*] (***15*** | *размер_массива*)

Задает количество строк, которое SQL*Plus будет возвращать при выполнении запроса за одно обращение к базе данных.

SET AUTO[*COMMIT*] (*ON* | ***OFF*** | *IMMEDIATE* | *количество_команд*)

Определяет, фиксирует ли SQL*Plus изменения автоматически. Также можно указать количество команд, которое будет выполнено между фиксациями.

SET AUTOP[*RINT*] (*ON* | ***OFF***)

Определяет, печатает ли SQL*Plus автоматически содержимое переменных связывания после того, как на них ссылается SQL-команда или PL/SQL-блок.

SET AUTORECOVERY (*ON* | ***OFF***)

Будучи включенной, позволяет команде RECOVER работать без вмешательства пользователя.

SET AUTOT[*RACE*] (*ON* | ***OFF*** | *TRACE*[*ONLY*]) [*EXP*[*LAIN*]] [*STAT*[*ISTICS*]]

Включает и выключает автоматический вывод плана исполнения и статистики исполнения для команды SQL.

SET BLO[*CKTERMINATOR*] (. | *завершающий_символ* | ***ON*** | *OFF*)

Задает символ, указывающий на завершение ввода PL/SQL-блока. По умолчанию это точка.

SET BUF[*FER*] (*имя_буфера* | ***SQL***)

Позволяет переключаться от одного буфера к другому. Обратите внимание на то, что для исполнения команд SQL может использоваться только один буфер.

SET CLOSECUR[*SOR*] (*ON* | ***OFF***)

Определяет, держит ли SQL*Plus командный курсор все время открытым.

SET CMDS[*EP*] (*ON* | ***OFF*** | *символ_разделитель*)

Определяет, можно ли вводить несколько команд SQL в одной строке, а также указывает символ-разделитель. Символом-разделителем по умолчанию является точка с запятой.

SET COLSEP разделитель_столбцов

Определяет, какой текст будет разделять столбцы данных. По умолчанию они разделяются одним пробелом.

SET COM[*PATIBILITY*] (*V7* | *V8* | ***NATIVE***)

Указывает версию Oracle, с которой должен быть совместим SQL*Plus. По умолчанию SQL*Plus автоматически сам определяет свое поведение.

SET CON[*CAT*] (***ON*** | *OFF* | *символ_конкатенации*)

Задает символ конкатенации, обозначающий конец имени переменной подстановки в команде SQL*Plus, команде SQL или блоке PL/SQL. По умолчанию это точка.

SET COPYC[*OMMIT*] *счетчик*

Определяет, как часто SQL*Plus выполняет фиксацию в процессе исполнения команды COPY. Значение по умолчанию – 0.

SET COPYTYPECHECK (***ON*** | *OFF*)

Определяет, проверяется ли тип при выполнении команды COPY для копирования данных из одной таблицы в другую.

SET DEF[INE] (***ON*** | *OFF* | *префикс*)

Указывает символ, который будет определять переменную подстановки. По умолчанию это &.

SET DESCRIBE [*DEPTH* (***1*** | *n* | *ALL*)] [*LINENUM* (*ON* | ***OFF***)][*INDENT* (***ON*** | *OFF*)]

Управляет поведением команды DESCRIBE. Параметр DEPTH определяет уровень рекурсивного описания объекта (например, таблица может содержать объектный столбец, который, в свою очередь, может содержать вложенную таблицу). LINENUM добавляет номера строк в выводимую командой DESCRIBE информацию. Параметр INDENT приводит к структурированному выводу описаний вложенных объектов (с отступом).

SET DOC[UMENT] (***ON*** | *OFF*)

Указывает, выводит ли SQL*Plus документацию, предопределенную для команды DOCUMENT.

SET ECHO (*ON* | ***OFF***)

Указывает, выводит ли SQL*Plus команды из командного файла по мере их исполнения.

SET EDITF[ILE] имя_файла_редактора

Указывает имя рабочего файла, используемого при вызове внешнего редактора командой EDIT. По умолчанию – *afiedt.buf*. Не поддерживается в *i*SQL*Plus.

SET EMB[EDDED] (*ON* | ***OFF***)

Включает и выключает возможность создания вложенных отчетов, позволяющую скомбинировать два отчета в одном, не меняя параметров нумерации страниц.

SET ESC[APE] (*ON* | ***OFF*** | *символ_экранирования*)

Задает символ экранирования, предшествующий символу-префиксу переменной подстановки (обычно это амперсанд) в случае, если требуется, чтобы символ интерпретировался буквально, а не как часть имени переменной. По умолчанию это символ обратной наклонной черты (\).

SET FEED[BACK] (*ON* | *OFF* | ***6*** | *кол-во_строк*)

Определяет, выводит ли SQL*Plus (и когда) количество строк, затронутых командой SQL. Установка параметра *кол-во_строк* в 0 имеет тот же эффект, что и SET FEEDBACK OFF.

SET FLAGGER (***OFF*** | *ENTRY* | *INTERMED[IATE]* | *FULL*)

Определяет, проверяет ли SQL*Plus команды пользователя на соответствие синтаксису ANSI/ISO.

SET FLU[SH] (***ON*** | *OFF*)

Определяет, поддерживается ли буферизация для вывода. Не поддерживается в *i*SQL*Plus.

SET HEA[DING] (***ON*** | *OFF*)

Указывает, выводятся ли заголовки столбцов при выборке данных.

SET HEADS[EP] (***ON*** | *OFF* | *разделитель_заголовка*)

Определяет символ, используемый для разрыва строки в заголовке столбца. По умолчанию это вертикальная черта (|).

SET INSTANCE (*имя_сервиса* | ***LOCAL***)

Определяет имя сетевого сервиса, задаваемого по умолчанию для команды CONNECT.

SET LIN[*ESIZE*] *ширина_строки*

Задает ширину строки в символах. По умолчанию – 80 символов.

SET LOBOF[*FSET*] *сдвиг*

Задает индекс в столбце типа CLOB, указывающий первый символ для отображения. Значение по умолчанию равно 1.

SET LOGSOURCE путь

Указывает SQL*Plus, где искать архивные журнальные файлы для восстановления. Значение по умолчанию отсутствует.

SET LONG длина

Указывает максимальное количество символов, которое может быть отображено из столбца типа LONG, CLOB, NCLOB и XMLType. По умолчанию равно 80.

SET LONGC[*HUNKSIZE*] *размер*

Указывает количество символов, извлекаемых за один прием из столбца типа LONG, CLOB, NCLOB и XMLType. По умолчанию равно 80.

SET MARK[*UP*] *параметры_разметки*

Позволяет указать язык разметки, применяемый для формирования вывода. Необходимо указать параметр HTML; все остальные могут отсутствовать.

HTML [*ON* | ***OFF***]

Определяет, следует ли использовать HTML в качестве языка разметки.

HEAD "*текст*"

Определяет содержимое тега <head>, выводимого в виде <head>текст</ head>.

BODY "*текст*"

Определяет содержимое тега <body>, выводимого в виде <body *текст*>.

TABLE "*текст*"

Определяет содержимое тега <table>, применяемого для форматирования результатов запроса. Тег выводится в виде <table *текст*>.

ENTMAP (***ON*** | *OFF*)

Определяет, будет ли SQL*Plus использовать принятые в HTML обозначения для специальных символов, такие как < и &gt для < и >.

SPOOL (*ON* | ***OFF***)

Определяет, будет ли SQL*Plus выводить теги <head> и <body> в файл спулинга для HTML-вывода.

PRE[*FORMAT*] [(*ON* | ***OFF***)

Определяет, должен ли вывод отчета заключаться в теги <pre> ... </pre> вместо того, чтобы помещаться в таблицу HTML.

SET MAXD[*ATA*] *макс_длина_строки*

Определяет максимальную длину строки, которую может обработать SQL*Plus. Это устаревший параметр; значения по умолчанию не существует.

SET NEWP[*AGE*] (*выводимые_строки* | *NONE*)

Определяет количество строк, которое SQL*Plus выводит между страницами. Если параметр равен 0, то SQL*Plus выводит перед каждой страницей символ новой страницы. Значение по умолчанию – 0. Не поддерживается в *i*SQL*Plus.

SET NULL текст_для_null

Определяет, какой текст SQL*Plus использует для представления значения NULL. По умолчанию NULL представляется пробелом.

SET NUMF[*ORMAT*] *спецификация_формата*

Устанавливаемый по умолчанию формат отображения чисел. Значения по умолчанию не существует.

SET NUM[*WIDTH*] (***10*** | *ширина*)

Устанавливаемая по умолчанию ширина поля для вывода чисел. Установка SET NUMFORMAT имеет больший приоритет.

SET PAGES[*IZE*] *строк_на_странице*

Задает количество строк, выводимых на одной странице. Значение по умолчанию равно 24.

SET PAU[*SE*] (*ON* | ***OFF*** | *пауза*)

Определяет, делает ли SQL*Plus паузу после каждой выведенной страницы. Не поддерживается в *i*SQL*Plus.

SET RECSEP (***WR***[***APPED***] | *EA*[*CH*] | *OFF*)

Определяет, будет ли между строками вывода печататься разделяющая строка. По умолчанию разделители выводятся, только если значение хотя бы в одном из столбцов записи занимает несколько строк.

SET RECSEPCHAR символ_разделитель

Определяет, какой символ будет выступать в качестве разделителя записей. По умолчанию – строка пробелов.

SET SCAN [***ON*** | *OFF*]

Включает и отключает подстановку пользовательской переменной. Устаревший параметр, сейчас замененный параметром SET DEFINE.

SET SERVEROUT[*PUT*] (*ON* | ***OFF***) [*SIZE размер_буфера*] [*FOR*[*MAT*] (*WRA* [*PPED*] | ***WOR***[***D_WRAPPED***] | *TRU*[*NCATED*])

Определяет, печатает ли SQL*Plus данные, выводимые блоками PL/SQL.

SET SHIFT[*INOUT*] (*VIS*[*IBLE*] | ***INV***[***ISIBLE***])

Управляет выводом символа смены регистра на терминалах IBM 3270. Не поддерживается в *i*SQL*Plus.

SET SHOW[*MODE*] (*ON* | ***OFF*** | *BOTH*)

Определяет, выводит ли SQL*Plus при изменении параметра его значения до и после изменения. Не поддерживается в *i*SQL*Plus.

SET SPACE количество_пробелов

Задает количество пробелов, выводимых между столбцами. По умолчанию равно 1. Устаревший параметр, вместо которого теперь используется SET COLSEP.

SET SQLBLANKLINES (*ON* | ***OFF***)

Определяет, можно ли включать пустые строки в состав команды SQL. Появился в версии 8.1.5. Не поддерживается в *i*SQL*Plus.

SET SQLC[*ASE*] (***MIXED*** | *UPPER* | *LOWER*)

Управляет автоматическим преобразованием регистров для команд SQL и блоков PL/ SQL.

SET SQLCO[*NTINUE*] *приглашение_на_продолжение*

Позволяет изменить приглашение на продолжение ввода для многострочных команд SQL. По умолчанию используется символ «больше» (>). Не поддерживается в *i*SQL*Plus.

SET SQLN[*UMBER*] (***ON*** | *OFF*)

Определяет, использует ли SQL*Plus номер строки как приглашение при вводе многострочной команды SQL. Не поддерживается в *i*SQL*Plus.

SET SQLPLUSCOMPAT[*IBILITY*] (*x.y*[*.z*] | ***8.1.7***)

Указывает, что при работе SQL*Plus обеспечивается совместимость с ранней версией ПО. Такая возможность появилась в Oracle9*i* и в настоящее время влияет лишь на работу команды VARIABLE, когда она применяется для объявления переменных типов NCHAR и NVARCHAR2.

SET SQLPRE[*FIX*] *префикс*

Задает префиксный символ SQL*Plus, позволяющий выполнить команду SQL*Plus во время ввода в буфер команды SQL или блока PL/SQL. По умолчанию – знак «решетка» (#). Не поддерживается в *i*SQL*Plus.

SET SQLP[*ROMPT*] *текст_приглашения*

Позволяет изменить приглашение SQL*Plus на ввод команды. По умолчанию – SQL>. Не поддерживается в *i*SQL*Plus.

SET SQLT[*ERMINATOR*] (***ON*** | *OFF* | *заверш_символ*)

Определяет, приводит ли завершение команды SQL символом «точка с запятой» к ее исполнению. Кроме того, он позволяет заменить завершающий символ на отличный от точки с запятой.

SET SUF[*FIX*] *расширение*

Задает расширение для командных файлов. По умолчанию – *.sql*. Не поддерживается в *i*SQL*Plus.

SET TAB (***ON*** | *OFF*)

Определяет, использует ли SQL*Plus знаки табуляции при форматировании. Не поддерживается в *i*SQL*Plus.

SET TERM[*OUT*] (***ON*** | *OFF*)

Определяет, печатает ли SQL*Plus вывод, сформированный файлом сценария SQL*Plus. Не поддерживается в *i*SQL*Plus.

SET TI[*ME*] (*ON* | ***OFF***)

Определяет, выводит ли SQL*Plus текущее время в составе приглашения на ввод команды. Не поддерживается в *i*SQL*Plus.

SET TIMI[*NG*] (*ON* | ***OFF***)

Определяет, выводит ли SQL*Plus фактическое время исполнения каждой команды SQL или блока PL/SQL.

SET TRIM[*OUT*] (***ON*** | *OFF*)

Определяет, убирает ли SQL*Plus завершающие пробелы из строк, выводимых на экран. Не поддерживается в *i*SQL*Plus.

SET TRIMS[*POOL*] (*ON* | ***OFF***)

Определяет, убирает ли SQL*Plus завершающие пробелы из строк, записываемых в файл спулинга. Не поддерживается в *i*SQL*Plus.

SET TRU[*NCATE*] (*ON* | ***OFF***)

Определяет, усекает ли SQL*Plus длинные строки.

SET UND[*ERLINE*] (*символ_подчеркивания* | (***ON*** | *OFF*))

Задает символ, используемый для подчеркивания заголовков столбцов. По умолчанию это дефис.

SET VER[*IFY*] (***ON*** | *OFF*)

Определяет, выводит ли SQL*Plus строки до и после подстановки переменных.

SET WRA[*P*] (***ON*** | *OFF*)

Определяет, переносит или усекает SQL*Plus длинные строки.

SHOW

```
SHO[W] [параметр| ALL | BTI[TLE] | ERR[ORS] [(FUNCTION |
    PROCEDURE | PACKAGE |PACKAGE BODY | TRIGGER | TYPE |
    TYPE BODY | DIMENSION | JAVA CLASS) [владелец.]имя_объекта] |
    LNO | PARAMETER[S] [имя_параметра] | PNO | REL[EASE] |
    REPF[OOTER] | REPH[EADER] | SGA | SPOO[L] |
    SQLCODE | TTI[TLE] | USER]
```

Позволяет увидеть текущее состояние окружения SQL*Plus.

Параметры

параметр

Любой из параметров, задаваемых командой SET.

ALL

Выводит все, за исключением ошибок и SGA.

BTI[*TLE*]

Выводит нижний колонтитул текущей страницы.

ERR[*ORS*]

Выводит перечень ошибок для хранимого объекта. Команда SHOW ERRORS без дополнительных параметров выводит список ошибок для объекта, созданного последним. Для того чтобы получить список ошибок для определенного объекта, следует указать его тип (функция, процедура и т. д.) и имя объекта.

FUNCTION | *PROCEDURE* | *PACKAGE* | *PACKAGE BODY* | *TRIGGER* | *TYPE* | *TYPE BODY* | *DIMENSION* | *JAVA CLASS*

Применяется совместно с SHOW ERRORS для указания типа интересующего вас объекта.

[*владелец.*] *имя_объекта*

Применяется совместно с SHOW ERRORS для указания имени объекта, для которого необходимо вывести перечень ошибок.

LNO

Выводит номер текущей строки.

PARAMETER[*S*] [*имя_параметра*]

Выводит текущее значение одного или нескольких параметров инициализации БД.

PNO

Выводит номер текущей страницы.

REL[*EASE*]

Выводит номер версии БД Oracle, к которой вы подключены.

REPF[*OOTER*]

Выводит текущие суммарные данные отчета.

REPH[*EADER*]

Выводит текущий заголовок отчета.

SGA

Выводит информацию о текущем состоянии SGA.

SPOO[*L*]

Сообщает, происходит ли в настоящее время спулинг вывода в файл.

SQLCODE

Выводит код, возвращенный последней командой SQL.

TTI[*TLE*]

Выводит заголовок текущей страницы.

USER

Выводит текущее имя пользователя.

SHUTDOWN

```
SHUTDOWN [NORMAL | IMMEDIATE | TRANSACTIONAL [LOCAL] | ABORT]
```

Позволяет остановить экземпляр Oracle. Команду могут выполнять только пользователи SYSDBA, SYSOPER и INTERNAL (INTERNAL не поддерживается начиная с версии Oracle9*i*.)

Параметры

NORMAL

Сервер Oracle ожидает добровольного отключения всех пользователей, прежде чем остановить экземпляр. Это поведение по умолчанию.

IMMEDIATE

Сервер Oracle немедленно отключает каждого пользователя, как только завершается его текущая команда SQL. Открытые транзакции откатываются.

TRANSACTIONAL [*LOCAL*]

Сервер Oracle ожидает, когда каждый пользователь завершит свою текущую транзакцию, затем отключает его. Если указать LOCAL, то он будет ожидать завершения только локальных (в отличие от распределенных) транзакций.

ABORT

Все фоновые процессы экземпляра незамедлительно прерываются. При следующем открытии БД будет выполнено восстановление после сбоя (или, если исполь-

зуется компонент Oracle Parallel Server или Real Application Clusters, то один из выживших экземпляров восстановит тот, работа которого была прервана).

SPOOL

```
SP[OOL] имя_файла | OFF | OUT
```

Заставляет записывать выводимую информацию в текстовый файл. SPOOL не поддерживается в *i*SQL*Plus.

Параметры

имя_файла

Имя файла, в который предполагается записывать выводимую информацию. Расширение по умолчанию зависит от операционной системы, это или LST, или LIS (в Windows – LST). Имя файла может включать в себя путь.

OFF

Отключает запись выводимой информации в текстовый файл.

OUT

Отключает запись выводимой информации в текстовый файл и печатает ее на принтере по умолчанию. Этот параметр недоступен в Windows-версиях SQL*Plus.

START

```
STA[RT] файл_сценария [аргумент...]
```

Выполняет сценарий SQL*Plus. команды START и @ работают одинаково. Описание параметров приведено в соответствующем разделе для команды @.

STARTUP

```
STARTUP [FORCE] [RESTRICT] [PFILE=имя_файла] [QUIET]
    [MOUNT [имя_бд] | OPEN [параметры_открытия] [имя_БД] | NOMOUNT]
    [EXCLUSIVE | {PARALLEL | SHARED}]

параметры_открытия::=
   READ (ONLY | WRITE [RECOVER]) | RECOVER
```

Альтернативная форма команды STARTUP (для миграции):

```
STARTUP [PFILE=имя_файла] MIGRATE [QUIET]
```

Позволяет запустить экземпляр Oracle и открыть БД. Команду могут выполнять только пользователи SYSDBA, SYSOPER и INTERNAL (INTERNAL не поддерживается начиная с версии Oracle9*i*.)

Параметры

FORCE

Принудительный запуск экземпляра. Если экземпляр работал, то выдача команды с указанием FORCE приведет к тому, что сначала будет выполнена команда SHUTDOWN ABORT; затем экземпляр будет перезапущен.

RESTRICT

Открывает БД в режиме ограниченного доступа. Подключиться смогут только пользователи, имеющие привилегию RESTRICTED SESSION.

PFILE=_имя_файла

Указывает SQL*Plus, что при запуске экземпляра следует использовать указанный файл параметров инициализации (*INIT.ORA* или *SPFILE*). Имя может включать в себя путь к файлу.

Файл параметров читает SQL*Plus, а не экземпляр Oracle, поэтому следует указывать путь к файлу параметров относительно компьютера, на котором работает SQL*Plus.

MIGRATE

Запускает базу данных в режиме OPEN MIGRATE с установкой параметров инициализации, допускающей запуск сценариев миграции к более новым или старым версиям. Этот параметр применяется только при работе с новыми версиями сервера.

QUIET

Запускает экземпляр без вывода информации о выделении памяти в рамках SGA.

MOUNT [*имя_БД*]

Означает, что база данных будет смонтирована, но не открыта.

[*имя_БД*]

Может (необязательно) указывать имя базы данных для подмены значения параметра инициализации DB_NAME.

OPEN [*параметры_открытия*] [*имя_БД*]

Означает, что база данных будет смонтирована, а затем открыта для нормальной работы.

NOMOUNT

Запускает экземпляр без монтирования базы данных.

READ (*ONLY* | *WRITE* [*RECOVER*])

READ ONLY применяется для того, чтобы открыть базу данных для чтения, но не для записи в нее. READ WRITE позволяет явно указать поведение по умолчанию.

RECOVER

Сообщает серверу Oracle, что при необходимости перед открытием БД следует выполнить восстановление носителя.

EXCLUSIVE

Приводит к тому, что база данных открывается и монтируется только текущим экземпляром. Другие экземпляры не могут ее использовать. Это установка по умолчанию, действующая, если не указаны ни SHARED, ни PARALLEL.

PARALLEL

Приводит к тому, что база данных открывается и монтируется таким образом, чтобы позволить одновременный доступ к ней нескольких экземпляров.

SHARED

Оказывает тот же эффект, что и PARALLEL.

STORE

```
STORE SET имя_файла [CRE[ATE] | REP[LACE] | APP[END]]
```

Формирует файл, состоящий из команд SET на основе текущего значения параметров. STORE не поддерживается в *i*SQL*Plus.

Параметры

имя_файла

Имя файла, в который записываются команды SET.

***CRE*[*ATE*]**

Приводит к тому, что команда не выполняется в случае, если файл с указанным именем уже существует. Это поведение по умолчанию.

REP[*LACE*]

Указывает SQL*Plus, что следует перезаписать любой существующий файл с тем же именем.

APP[*END*]

Дописывает команды SET в конец существующего файла.

TIMING

```
TIMI[NG] [START [имя_таймера] | SHOW | STOP]
```

Позволяет запустить и остановить таймер или вывести его значение для измерения истекшего времени.

Параметры

START [*имя_таймера*]

Запускает новый таймер и (необязательно) дает ему указанное имя.

SHOW

Выводит текущее значение таймера, запущенного последним.

STOP

Останавливает таймер, запущенный последним, выводит его текущее значение, затем удаляет его.

TTITLE

```
TTI[TLE] [OFF | ON] |
    [COL x | S[KIP] x | TAB x | LE[FT] | CE[NTER] | R[IGHT] |
    BOLD | FOR[MAT] спецификация_формата | текст | переменная...]
```

Определяет верхний колонтитул страницы для отчета. Команда TTITLE без параметров выводит текущие значения.

Параметры

OFF

Отключает вывод колонтитула страницы, но не стирает его определение.

ON

Включает вывод колонтитулов страниц.

COL x

Приводит к тому, что любой текст колонтитула, следующий за данным параметром, выводится в указанной позиции столбца.

S[*KIP*] *x*

Вставляет указанное количество разрывов строки перед выводом последующего текста колонтитула.

TAB x

Переходит на указанное количество символьных столбцов вперед. Отрицательное значение приводит к переходу назад.

LE[*FT*]

Приводит к тому, что последующий текст колонтитула выводится, начиная с самого левого столбца текущей строки колонтитула.

CE[*NTER*]

Приводит к тому, что последующий текст колонтитула выравнивается по центру текущей строки.

R[*IGHT*]

Приводит к тому, что последующий текст колонтитула выводится с выравниванием по правому краю.

BOLD

Выделяет колонтитул, печатая его полужирным шрифтом либо выводя на экран трижды. Выделяется только текст колонтитула, следующий за командой BOLD.

FOR[*MAT*]

Позволяет управлять выводом последующих числовых данных в колонтитуле.

спецификация_формата

Строка, указывающая формат отображения, который должен использоваться для последующего вывода числовых данных в колонтитуле.

текст

Любой текст, который должен быть помещен в колонтитул.

переменная

Вставляет значение указанной пользовательской переменной. Можно также использовать одну из системных переменных, поддерживаемых SQL*Plus, которые перечислены в табл. 12.1:

*Таблица 12.1. Системные переменные SQL*Plus*

Системная переменная	Значение
SQL.PNO	Номер текущей страницы
SQL.LNO	Номер текущей строки
SQL.RELEASE	Номер текущей версии Oracle
SQL.SQLCODE	Код ошибки, возвращенной последним запросом SQL
SQL.USER	Имя пользователя Oracle, запустившего отчет

UNDEFINE

```
UNDEF[INE] имя_переменной [имя_переменной...]
```

Удаляет определение пользовательской переменной.

Параметр

имя_переменной

Имя удаляемой пользовательской переменной. Можно удалять несколько переменных одной командой, перечисляя их через пробелы.

VARIABLE

```
VAR[IABLE] [имя_переменной [тип_данных]]
```

Позволяет объявить переменные связывания. Переменные связывания – это фактические переменные, которые могут использоваться в блоке PL/SQL или команде SQL. Команда VARIABLE без параметров выводит список переменных связывания, определенных на текущий момент.

Параметры

имя_переменной

Имя, которое вы хотите дать переменной. Если не указывать тип данных, то SQL*Plus выведет тип данных для переменной, имя которой было указано.

тип_данных

Тип данных переменной. Разрешены следующие типы:

NUMBER

Число с плавающей точкой, аналогично переменной типа NUMBER в PL/SQL или столбцу NUMBER в таблице. В отличие от PL/SQL, SQL*Plus не позволяет указать длину или точность, то есть такое объявление, как NUMBER (9,2), не разрешено.

CHAR **[(***длина* **[***CHAR* | *BYTE***])]**

Символьная строка фиксированной длины. Указывать длину не обязательно. Если длина не задана, строка будет однобайтовой.

NCHAR **[(***длина***)]**

Символьная строка фиксированной длины в национальном наборе символов. Указывать длину не обязательно. Если длина не задана, строка будет однобайтной.

VARCHAR2 **(***длина* **[***CHAR* | *BYTE***])**

Символьная строка переменной длины.

NVARCHAR2 **(***длина***)**

Символьная строка переменной длины в национальном наборе символов.

CLOB

Символьный большой объект.

NCLOB

Символьный большой объект (используется национальный набор символов).

REFCURSOR

Курсорная переменная, через которую можно осуществить возврат результатов SQL-запроса из PL/SQL в SQL*Plus.

WHENEVER

```
WHENEVER (OSERROR | SQLERROR)
    (EXIT [SUCCESS | FAILURE | значение | :переменная_связывания ]
    [COMMIT | ROLLBACK] | CONTINUE [COMMIT | ROLLBACK | NONE])
```

Управляет поведением SQL*Plus при ошибке операционной системы или ошибке SQL.

Параметры

WHENEVER OSERROR

Эта форма команды применяется для того, чтобы сообщить SQL*Plus, что следует делать в случае ошибки операционной системы.

WHENEVER SQLERROR

Данная форма команды позволяет сообщить SQL*Plus, что следует делать в случае ошибки команды SQL или блока PL/SQL.

EXIT SUCCESS

Выход со статусом успешного завершения.

EXIT FAILURE

Выход со статусом ошибки.

EXIT значение

Выход со статусом, равным указанному значению. Значение может быть литералом или пользовательской переменной.

EXIT :переменная_связывания

Выход со статусом, равным значению указанной переменной связывания.

CONTINUE

Не выходить в случае ошибки. Это поведение по умолчание при первом запуске SQL*Plus.

COMMIT

Может применяться совместно с EXIT и CONTINUE. Вынуждает SQL*Plus зафиксировать текущую транзакцию в случае ошибки. Это поведение по умолчанию при использовании ключевого слова EXIT.

ROLLBACK

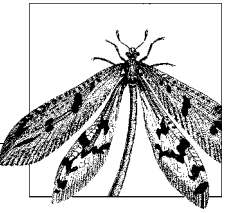
Может применяться совместно с EXIT и CONTINUE и вынуждает SQL*Plus откатывать текущую транзакцию при ошибке.

NONE

Может применяться совместно с CONTINUE и вынуждает SQL*Plus не осуществлять ни фиксации, ни отката при ошибке. Это поведение по умолчанию при использовании ключевого слова CONTINUE.

13

Экспорт и импорт



СУБД Oracle хранит информацию в собственных файлах данных в своем формате. Любой пользователь может получить доступ к этим данным при помощи SQL.

Кроме того, СУБД Oracle также включает в себя утилиты, которые позволяют извлекать данные из базы данных в двоичный файл (утилита Export) и для внесения данных в БД Oracle из двоичного файла (утилита Import). Эти утилиты можно использовать для создания копий структур и данных БД Oracle. Утилиты Import и Export полезны для перемещения данных в другую базу данных, а также для резервного копирования и восстановления. (Однако знайте, что Oracle предоставляет более развитые средства резервного копирования и восстановления, которые рассмотрены в главе 15).

В главе описаны синтаксис и основные операции утилит Export и Import.

Основы экспорта и импорта

Существуют два режима работы утилит Export и Import:

Интерактивный режим

В этом режиме утилита запрашивает у пользователя информацию, которая необходима для выполнения задачи.

Командный режим

В этом режиме параметры, необходимые для выполнения задачи, берутся или из командной строки, или из указанного файла параметров.

Интерактивный режим для утилит Export и Import является наследием более ранних версий программного обеспечения. Применение командного интерфейса обеспечивает большую гибкость работы.

Перемещая данные между СУБД разных версий посредством утилит Export и Import, необходимо обращать внимание на версию самих утилит. Утилита Import должна иметь ту же версию, что и СУБД, получающая данные, в то время как версия утилиты Export должна быть взята из младшей из используемых версий БД.

Прежде чем запускать какую-либо версию Export или Import, необходимо запустить сценарий *catexp.sql* или *catalog.sql*, для того чтобы подготовить базу данных к таким операциям.

В этой главе команды для вызова утилит Export и Import будут называться *exp* и *imp* соответственно. Для большинства систем это будут корректные имена команд. Однако имейте в виду, что в старых версиях Oracle для Windows в имена исполняемых файлов включался номер версии, поэтому в подобных системах можно встретить такие названия, как *imp80*, *exp73* и т. д. Названия команд, применяемых для вызова данных утилит на конкретной платформе, можно уточнить, обратившись к соответствующей документации для этой платформы.

Интерактивная работа

Утилиты Export и Import можно вызвать при помощи следующих команд:

```
exp [id_пользователя]
imp [id_пользователя]
```

где *id_пользователя* – это имя пользователя или комбинация «имя пользователя/пароль». Кроме того, данный параметр может просто отсутствовать. Если пароль не указан, то он будет запрошен в интерактивном диалоговом окне. Параметр *id_пользователя* может быть уточнен идентификатором *@имя_сетевой_службы* или ключевыми словами AS SYSDBA. Данные параметры необходимо указывать в ряде конкретных случаев применения утилиты Export, например при восстановлении табличного пространства на момент времени в Oracle8.

Параметры, запрашиваемые утилитой Export

В разделе будут рассмотрены параметры, запрашиваемые утилитой Export при работе в интерактивном режиме. В отличие от других глав, в этой значения по умолчанию будут приводиться между названием запрашиваемого параметра и символом «>"», как они отображаются в реальном интерактивном сеансе. Для наглядности значения по умолчанию будут выделены полужирным шрифтом (например, SYSTEM – это имя пользователя по умолчанию).

Username: ***SYSTEM*** *>*

Имя пользователя для данного сеанса экспорта.

Password:

Пароль для указанного имени пользователя.

Enter array fetch buffer size: ***4096*** *>*

Размер буфера при выборке массивом, который может влиять на производительность. Более подробная информация о данном параметре приведена в разделе «Общие параметры» далее в этой главе при описании ключевого слова BUFFER.

Export file: ***expdat.dmp*** *>*

Имя файла экспорта.

(1)E(ntire database), (2)U(sers), or (3)T(ables): ***(2)U*** *>*

Выбор типа экспорта. Если выбрать U или T, то утилита Export будет запрашивать значения дополнительных параметров.

Export grants (yes/no): ***yes*** *>*

Указывает, следует ли экспортировать выданные права.

Export table data (yes/no): ***yes*** >

Указывает, следует ли экспортировать фактические данные таблиц. Если не включать данные в экспорт, то будут экспортированы только определения указанных таблиц.

Compress extents (yes/no): ***yes*** >

Указывает, следует ли при экспорте выполнять сжатие экстентов. Более подробная информация о данном параметре приведена в разделе «Параметры экспорта» далее в этой главе при описании ключевого слова COMPRESS.

Утилита может запрашивать не все перечисленные параметры. Например, если не ввести имя пользователя и пароль, то программа запросит эти данные. Если выбрать экспорт всей базы данных, то не будут запрашиваться имена экспортируемых таблиц.

После того как вы сообщите программе все необходимые сведения, начнет выполняться задание экспортирования и информация о состоянии исполнения будет отображаться в виде последовательности сообщений. Выводимая информация включает в себя сведения об экспортируемых сущностях (например, об экспортируемых таблицах) и о размере этих сущностей (например, о количестве строк).

Параметры, запрашиваемые утилитой Import

В этом разделе будут рассмотрены параметры, запрашиваемые утилитой Import при работе в интерактивном режиме. Как и для утилиты Export, значения по умолчанию будут приводиться между названием запрашиваемого параметра и символом «>», как они отображаются в реальном интерактивном сеансе. Для наглядности значения по умолчанию будут выделены полужирным шрифтом (например, SYSTEM – это имя пользователя по умолчанию).

Username: ***SYSTEM*** >

Имя пользователя для данного сеанса импорта.

Password:

Пароль для указанного имени пользователя.

Import file: ***expdat.dmp*** >

Импортируемый файл.

Enter insert buffer size (minimum is 4096): ***30720***>

Размер буфера при выборке массивом, который может влиять на производительность. Более подробная информация приведена в разделе «Общие параметры» далее в главе при описании ключевого слова BUFFER.

List contents of import file only (yes/no): ***no*** >

Может применяться для определения содержимого файла импорта без реального импортирования каких-либо данных.

Ignore create error due to object existence (yes/no): ***no*** >

Параметр определяет поведение утилиты Import в случае, если импортируемый объект уже существует. Если выбран вариант no (умолчание), то выводится сообщение об ошибке создания объекта и строки для объекта не импортируются. Если же выбран вариант yes, то ошибка создания не отображается и строки импортируются. В любом случае существующий объект никогда не заменяется.

Import grants (yes/no): ***yes*** >

Указывает, следует ли импортировать выданные права из файла экспорта.

Import table data (yes/no): ***yes*** *>*

Указывает, следует ли импортировать данные из таблиц в файле экспорта.

Import entire export file (yes/no): ***no*** *>*

Указывает, следует ли импортировать весь файл экспорта.

Username: ***SCOTT*** *>*

Если импортируется не весь файл экспорта, то можно указать, какие именно схемы будут импортированы.

Enter table(T) or partition (T:P) names: Null list means all tables for user >

Указывает список импортируемых таблиц или разделов. Если ничего не ввести, то будут импортированы все таблицы или разделы.

Как и для утилиты Export, могут быть запрошены не все перечисленные параметры. А какие именно параметры потребуется ввести, зависит от уже предоставленной информации.

Командный режим

Синтаксис вызова утилит Export и Import из командной строки таков:

```
exp | imp id_пользователя {имя_параметра = значение ... | PARFILE = имя_файла}
```

Параметр *id_пользователя* – это имя пользователя или комбинация имя пользователя/пароль, при необходимости с указанием ключевого слова *@имя_сетевой_службы* для идентификации экземпляра. Если пароль не указан, то утилиты запросят его у пользователя.

Такой синтаксис применяется для задания параметров в составе команды. Каждый параметр, задаваемый в командной строке, отделяется от остальных параметров пробелом или запятой. Отдельное значение параметра может быть заключено в скобки. Несколько значений для параметра должны быть заключены в скобки и разделены запятыми. Общая длина команды не может превышать максимальную длину команды, определяемую операционной системой.

Существует возможность хранения значения параметров в файле параметров. Для этого укажите имя такого файла в инструкции PARFILE *имя_файла* следующим образом:

```
exp greenie/****@bailey FULL=Y FILE=EXP.DMP ROWS=Y PARFILE=parms.dat
```

Даже если вы указываете файл параметров, все равно можно включать параметры и в состав команды. Если один и тот же параметр указан и в командной строке, и в файле параметров, то более высокий приоритет имеет значение, указанное в командной строке позже. Например, если один параметр упомянут и в командной строке, и в файле параметров, то значение из файла параметров будет иметь более высокий приоритет, если инструкция PARFILE, ссылающаяся на данный файл, расположена после устанавливающего значение параметра ключевого слова командной строки. В предыдущем примере, если бы PARFILE содержал настройку FULL=N, то команда не выполнила бы экспорт всей базы данных, даже несмотря на то, что именно это указывает настройка FULL=Y в командной строке.

В последующих разделах будут описаны общие идентификаторы и параметры (они относятся и к утилите Export, и к утилите Import), а затем – параметры, специфичные для каждой из утилит.

Общие параметры

Множество различных команд Export и Import используют следующий идентификатор:

имя_файла

Имя файла экспорта или журнального файла. По умолчанию файлу экспорта присваивается имя *expdat.dmp*. Файл экспорта по умолчанию имеет расширение *.dmp*, а журнальный файл – *.log*.

Параметры, перечисленные далее, могут применяться в обеих утилитах:

BUFFER = размер_буфера

Указывает размер буфера (в байтах), используемого для выборки или загрузки строк. Этот параметр определяет максимальное количество одновременно извлекаемых утилитой Export строк по такой формуле:

кол-во извлекаемых строк = размер буфера / макс длина строки

Если указать 0, то строки будут выбираться или загружаться по одной. Таблицы, в которых есть столбцы типов LONG, LOB, BFILE, REF или ROWID, всегда извлекаются или загружаются по одной строке. Параметр BUFFER применяется только к обычному экспорту (conventional path export) и не влияет на прямой экспорт. Значение по умолчанию определяется операционной системой.

*CONSTRAINTS = **Y** | N*

Определяет, экспортируются ли (или импортируются) ограничения для таблиц. Значение по умолчанию – Y.

FEEDBACK = число

Если задано положительное значение, то утилиты Export или Import должны выводить индикатор хода процесса в виде строки точек. Одна точка выводится после обработки указанного количества строк. Значение по умолчанию равно 0.

FILE = имя_файла

Указывает имя файла, в который Export выгружает данные, или имя файла, из которого Import загружает данные. Если размер файла ограничен параметром FILESIZE, как в Oracle8*i*, то можно указать несколько имен файлов (параметр *имя_файла* был описан в начале этого раздела).

FILESIZE = число

Ограничивает размер отдельного файла экспорта. Если экспортируемые данные превышают указанный размер, то утилита Export создаст несколько файлов, для этого можно указать в параметре FILES несколько имен файлов. Максимально возможное значение параметра ограничивается операционной системой и версией Oracle. Параметр FILESIZE необходимо применять при импортировании из файла экспорта, созданного с указанием параметра FILESIZE. Появился в Oracle8*i*.

*FULL = Y | **N***

Если значение параметра равно Y, то Export осуществляет полный экспорт базы данных, т. е. экспортирует все объекты из БД, а Import импортирует весь файл экспорта целиком. По умолчанию равен N. Для полного экспорта БД необходима роль EXP_FULL_DATABASE.

GRANTS = ***Y*** | *N*

Определяет, будут ли экспортироваться выданные права и будут ли импортироваться ранее экспортированные права. Какие именно права экспортируются, зависит от режима экспорта. При полном экспорте БД экспортируются все права на таблицы; в пользовательском режиме экспортируются только права, выданные владельцем таблицы. Значение по умолчанию равно Y.

HELP = *Y* | ***N***

Вывод справки с описаниями параметров импорта и экспорта. Значение по умолчанию равно N. Если указать HELP=Y, то команда только выведет справку и не будет выполнять экспорт или импорт.

INDEXES = ***Y*** | *N*

Определяет, будут экспортированы все индексы и будут ли индексы создаваться или обновляться после импортирования таблицы. Значение по умолчанию равно Y. Если же задано значение N, то индексы не будут экспортированы. Установка для INDEXES значения Y при импорте предполагает, что именно такое значение параметра было определено при создании файла экспорта. Индексы, формируемые системой, такие как LOB-индексы, OID-индексы или индексы уникальных ограничений, повторно создаются утилитой Import вне зависимости от значения данного параметра.

LOG =*имя_файла*

Определяет имя файла, в котором будут сохраняться все выводимые утилитой сообщения (параметр *имя_файла* был описан в начале этого раздела). Если параметр LOG не задан, то сообщения не будут отправляться в файл.

PARFILE = *имя_файла*

Если этот параметр указан, то он задает имя файла, содержащего перечень параметров утилиты Export или Import (параметр *имя_файла* описан в начале данного раздела).

POINT_IN_TIME_RECOVER = *Y* | ***N***

Определяет, будет ли утилита Export экспортировать в БД Oracle одно или несколько табличных пространств таким образом, чтобы при запуске утилиты Import можно было восстановить табличное пространство на более ранний момент времени, не затрагивая при этом остальную часть БД. Для того чтобы выполнить восстановление на момент времени при помощи утилиты Import, необходимо установить данный параметр в Y при экспорте, а также установить его значение равным Y в команде *imp*. Значение по умолчанию – N. Не поддерживается после Oracle8.

RECORDLENGTH = *длина*

Задает длину записи для файла экспорта или файла для импорта. Значение по умолчанию определяется стандартом для операционной системы хоста. Если файл экспорта передается для импорта в другую систему, то данный параметр позволяет привести длину записи в соответствие с длиной записи в другой системе. Кроме того, можно повысить производительность прямого экспорта (DIRECT=Y), увеличив значение RECORDLENGTH.

RESUMABLE = ***N*** | *Y*

Применяется для разрешения и запрещения возобновляемого выделения памяти. Значение по умолчанию равно N.

RESUMABLE_NAME = имя

Применяется для идентификации возобновляемой операции для оператора. Требует, чтобы значение параметра RESUMABLE было равно Y. Появился в Oracle9*i*.

RESUMABLE_TIMEOUT = число

Указывает время (в секундах), в течение которого операция будет ожидать своего возобновления от оператора после выделения большего объема памяти. По умолчанию равен 7200. Появился в Oracle9*i*.

*ROWS = **Y** | N*

Указывает, будут ли строки таблиц записываться в файл экспорта или импортироваться в процессе импорта. По умолчанию равен Y. Значение N применяется для экспорта структуры всех объектов без их содержимого или для создания при импорте структуры без каких-либо данных.

TABLES = (имя_таблицы[,имя_таблицы ...])

При экспорте перечисляет имена экспортируемых таблиц и разделов, указывая, что утилита Export будет работать в табличном режиме. Значения по умолчанию не существует. Таблицы или разделы могут быть заданы в формате *схема.таблица:имя_раздела*. Для импорта параметр задает названия импортируемых таблиц. Помните, что при импорте нет возможности уточнить имя таблицы именем схемы.

схема

Задает имя схемы, из которой будет экспортироваться таблица или раздел. Если не указано, то используется схема из USERID, если только параметр FULL не установлен в Y.

таблица

Задает имя экспортируемой таблицы. Если таблица из списка является секционированной, а имя раздела не указано, то будут экспортированы все разделы данной таблицы.

имя_раздела

Указывает, что экспорт будет выполняться на уровне разделов, что позволит экспортировать один или несколько определенных разделов таблицы. Если данное значение не указано для секционированной таблицы, то экспортируются все ее разделы.

TABLESPACES = (имя_табличного_пространства[,имя_табличного_пространства...])

При экспорте будут экспортированы все таблицы, которые существуют в перечисленных табличных пространствах или имеют раздел в одном из таких пространств.Для импорта параметр задает список импортируемых табличных пространств. Появился в Oracle8*i*.

USERID = имя_пользователя/пароль@имя_сетевой_службы

Указывает регистрационные данные пользователя, выполняющего импорт или экспорт (если такие данные отсутствуют в командной строке). Составляющие *пароль* и *@имя_сетевой_службы* могут отсутствовать; если не указать их в составе данного параметра, то утилита запросит их у пользователя.

VOLSIZE = число

Определяет максимальный размер (в байтах) файла экспорта на магнитной ленте. Появился в Oracle8*i*.

Параметры экспорта

Следующие параметры можно задавать только для утилиты Export:

*COMPRESS = **Y** | N*

Определяет, каким образом Export поступает с исходным экстентом для табличных данных. Значение по умолчанию, Y, приводит к формированию команды создания объекта, которая в дальнейшем будет использована при импорте для создания объекта, у которого все экспортированные данные находятся в одном исходном экстенте. Если установить COMPRESS=N, то Export возьмет текущие параметры хранения, в том числе и значения размеров экстентов INITIAL и NEXT.

*CONSISTENT = Y | **N***

Определяет, использует ли Export команду SET TRANSACTION READ ONLY для гарантии того, что доступные утилите данные согласованы на единый момент времени и не изменятся в процессе выполнения команды *exp*. Параметр необходимо установить в Y, если предполагается, что другие приложения будут обновлять БД после запуска экспорта. Значение по умолчанию – N.

*DIRECT = Y | **N***

Указывает путь экспорта – *прямой* (*direct*) или *обычный* (*conventional*). При прямом экспорте данные извлекаются непосредственным считыванием, без привлечения уровня SQL, поэтому он может быть гораздо более быстрым, чем экспорт по обычному пути. По умолчанию равен N. Прямой экспорт невозможен для таблиц, содержащих столбцы любого из типов REF, LOB, BFILE, а также столбцы объектных типов: VARRAY и вложенные таблицы.

FLASHBACK_SCN = n

Указывает системный номер изменения (SCN) для ретроспективного экспорта. Появился в Oracle9*i*.

FLASHBACK_SCN = время

Указывает время для ретроспективного экспорта. Сервер Oracle идентифицирует наиболее близкий к этому времени номер SCN и использует его для экспорта. Появился в Oracle9*i*.

INCTYPE = INCREMENTAL | CUMULATIVE | COMPLETE

Если параметр задан, то он определяет одну из разновидностей инкрементного экспорта:

INCREMENTAL

Экспортирует все объекты БД, изменившиеся после последнего инкрементального, кумулятивного или полного экспорта. Изменения отслеживаются в таблице SYS.INCEXP, которая после экспорта обновляется новыми значениями ITIME и EXPID.

CUMULATIVE

Экспортирует все объекты БД, изменившиеся после последнего кумулятивного или полного экспорта. Изменения отслеживаются в таблице SYS.INCEXP, которая после экспорта обновляется новыми значениями CTIME, ITIME и EXPID.

COMPLETE

Экспортирует все объекты, затем обновляет таблицы SYS.INCEXP и SYS.INCVID. (Экспорт, выполняемый при установке в Y параметра FULL, обновляет эти таблицы, только если задан параметр INCTYPE). Параметр не поддерживается начиная с Oracle9*i*.

OBJECT_CONSISTENT = *Y* | ***N***

Приводит к тому, что каждый объект экспортируется в своей собственной транзакции, в отличие от экспорта с установленным в Y параметром CONSISTENT, когда используется всего одна транзакция для всего экспорта. По умолчанию равен N. Появился в Oracle9*i* Release 2.

QUERY = *инструкция_запроса*

Инструкция_запроса – это инструкция WHERE, дополняющая команду SELECT для каждой таблицы или раздела, приведенных в инструкции TABLE. *Инструкция_запроса* – это строка, поэтому она должна начинаться и заканчиваться символами \". Если необходимо разделить символьные строки в запросе при помощи одинарных кавычек, используйте символы \'. Появился в Oracle8*i*.

OWNER = (*имя_пользователя*[,*имя_пользователя*...])

Если параметр задан, то он указывает, что экспорт производится в пользовательском режиме и перечисляет пользователей, объекты которых будут экспортированы.

RECORD = ***Y*** | *N*

Определяет, будет ли Export создавать запись об инкрементном или кумулятивном экспорте в системных таблицах SYS.INCEXP, SYS.INCFIL и SYS.INCVID. По умолчанию равен Y. Параметр недоступен, начиная с Oracle9*i*.

RECOVERY_TABLESPACES = (*имя_табличного_пространства*[,*имя_табличного_пространства* ...])

Определяет табличные пространства, которые будут экспортироваться для восстановления на момент времени. Значения по умолчанию не существует. Параметр недоступен начиная с Oracle8.

STATISTICS = ***ESTIMATE*** | *COMPUTE* | *NONE*

Определяет тип SQL-команд ANALYZE, которые будут формироваться при восстановлении экспортированных данных с помощью утилиты Import. По умолчанию задается значение ESTIMATE. Появился в Oracle8*i*.

TRANSPORTABLE_TABLESPACE = *Y* | ***N***

Позволяет экспортировать метаданные для переносимых табличных пространств. Применяется в сочетании с параметром TABLESPACES. По умолчанию задается значение N. Появился в Oracle8*i*.

TRIGGERS = ***Y*** | *N*

Приводит к экспортированию триггеров. По умолчанию задается значение Y. Появился в Oracle9*i*.

TTS_FULL_CHECK = ***FALSE*** | *TRUE*

Если значение параметра TTS_FULL_CHECK равно TRUE, то утилита Export проверяет, не имеют ли указанные табличные пространства зависимостей вне рамок экспорта. По умолчанию задается значение FALSE. Появился в Oracle9*i*.

Параметры импорта

Следующие параметры имеют смысл только для утилиты Import:

ANALYZE = ***Y*** | *N*

Определяет, будет ли утилита Import выполнять SQL-команды ANALYZE, обнаруженные в файле экспорта. По умолчанию устанавливается значение Y. Не поддерживается начиная с Oracle9*i*.

CHARSET = *имя_набора_символов*

Относится только к файлам экспорта Oracle Version 6 и указывает фактический набор символов, использованный в момент экспорта. Утилита Import проверит, является ли указанный набор символом ASCII или EBCDIC на основе набора символов, заданного в файле экспорта. Если такой файл создается в Oracle7 или Oracle8, то набор символов указывается внутри файла экспорта и преобразование к текущему набору символов БД происходит автоматически.

COMMIT = *Y* | ***N***

Определяет, должен ли процесс импорта фиксироваться после вставки каждого массива. По умолчанию равен N, что означает, что результат импорта фиксируется после загрузки каждой таблицы. Если COMMIT=N и таблица является секционированной, то каждый раздел в файле экспорта импортируется в отдельной транзакции. Установка параметра COMMIT в Y предотвращает слишком сильное увеличение размеров сегментов отката. Такая настройка рекомендуется в тех случаях, когда таблица имеет ограничение уникальности, т. к. при перезапуске импорта все уже импортированные строки будут отвергнуты с нефатальной ошибкой. Если таблица не имеет ограничения уникальности и COMMIT=Y, то при повторном импорте данных возможно появление повторяющихся строк.

COMPILE = ***Y*** | *N*

Указывает, должен ли процесс импорта компилировать импортируемые пакеты, процедуры и функции. По умолчанию устанавливается значение Y. Появился в Oracle9*i*.

DATAFILES = (*имя_файла* [*имя_файла*...])

В случае использования переносимых табличных пространств данный параметр предоставляет список файлов данных, которые будут переданы. Появился в Oracle8*i*.

DESTROY = *Y* | ***N***

Определяет, должны ли повторно использоваться существующие файлы данных, составляющие БД, если импорт пересоздает табличные пространства. По умолчанию устанавливается значение N.

Если файлы данных хранятся на RAW-устройстве, то установка DESTROY в N не предотвращает перезаписи файлов.

FROMUSER = (*имя_пользователя* [,*имя_пользователя*...])

Задает список схем, содержащих объекты для импортирования. По умолчанию для пользователей, не обладающих ролью IMP_FULL_DATABASE, импорт выполняется в пользовательском режиме, т. е. импортируются объекты текущего пользователя. Если указанный пользователь не существует в БД, то объекты бу-

дут импортированы в схему текущего пользователя, если только не задан и параметр TOUSER.

*IGNORE = Y | **N***

Определяет, каким образом обрабатываются ошибки создания объектов.Если значение параметра IGNORE равно Y, то утилита Import не записывает в журнал ошибки создания объектов БД и объекты импортируются. По умолчанию задается значение N, и в этом случае Import регистрирует в журнале и/или выводит на экран сообщение об ошибке создания объекта прежде, чем продолжить работу, и не импортирует такой объект.

*INCTYPE = {SYSTEM | **RESTORE**}*

Указывает, что будет применяться один из описанных ниже типов инкрементного импорта. По умолчанию задается значение RESTORE. Не поддерживается начиная с Oracle9*i*.

SYSTEM

Импортирует самую свежую версию системных объектов, включая библиотеки функций и определения объектных типов, но при этом пользовательские данные или объекты не импортируются.

RESTORE

Импортирует все пользовательские объекты БД и данные, содержащиеся в файле экспорта.

INDEXFILE = имя_файла

Указывает файл (параметр *имя_файла* описан ранее), который будет получать команды создания таблицы, индекса и кластера. Команды для таблицы и кластера включаются как комментарии, но могут редактироваться. Значения по умолчанию для данного параметра не существует. Он может использоваться, только если выполняется импорт полной БД или заданы параметры FROMUSER, TOUSER или TABLES.

*SHOW = Y | **N***

Если параметр SHOW установлен в Y,[1] то команды SQL, содержащиеся в файле экспорта, выводятся только на дисплей, а объекты не импортируются. Значение Y можно задать, только если выполняется импорт полной БД или заданы параметры FROMUSER, TOUSER или TABLES. Значение по умолчанию – N.

*SKIP_UNUSABLE_INDEXES = Y | **N***

Определяет, будет ли утилита Import пропускать создание индексов, находящихся в состоянии Unusable. Без данного параметра не удастся выполнить вставку строк, которая пытается обновить непригодные для использования индексы. По умолчанию устанавливается значение N.

*STREAMS_CONFIGURATION = Y | **N***

Определяет, следует ли импортировать любые имеющиеся в файле экспорта метаданные, связанные с потоками. По умолчанию устанавливается значение N. Появился в Oracle9*i*.

---

[1] Параметр SHOW можно использовать для повторного формирования DDL. Выполните экспорт, а затем импорт, установив SHOW в Y, и перенаправьте вывод в файл. Отредактируйте файл, чтобы избавиться от сообщений, которые не относятся к DDL, и вы получите ваши DDL-команды.

STREAMS_INSTANTIATION = *Y* | *N*

Определяет, следует ли импортировать любые метаданные, относящиеся к реализации потоков. Значение по умолчанию – N. Появился в Oracle9*i*.

TOID_NOVALIDATE

Подавляет проверку корректности типов для существующих идентификаторов типов в базе данных импорта. Появился в Oracle8*i*.

TOUSER = (*имя_пользователя*[,*имя_пользователя*...])

Задает список имен пользователей, в схемы которых будет выполняться импорт. Если задается несколько схем, то их имена должны образовывать пары с именами схем в соответствующем параметре FROMUSER, например:

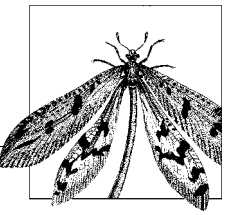
```
FROMUSER=(scott,harry) TOUSER=(dave,brian)
```

TTS_OWNERS = (*имя_владельца*[,*имя_владельца* ...])

Указывает владельцев импортированного переносимого табличного пространства. Появился в Oracle8*i*.

14

SQL*Loader



В главе 13 рассказывалось о применении утилит Export и Import для перемещения данных между базами данных Oracle. При этом для перемещения используется двоичный файл. Утилита SQL*Loader, которой посвящена данная глава, – это еще одна утилита Oracle, предназначенная для загрузки данных, представленных в БД Oracle в стандартных форматах файлов операционной системы.

SQL*Loader позволяет преобразовывать данные в процессе загрузки и воздействовать на них. Вы можете:

- Считывать несколько файлов и выполнять загрузку в несколько таблиц БД; причем речь может идти не только об обычных реляционных таблицах, но и об объектных таблицах с такими столбцами, как вложенные таблицы, массивы переменной длины или большие объекты (LOB).
- Обрабатывать файлы с данными фиксированной длины, переменной длины и поточными данными, которые могут относиться к различным типам.
- Воздействовать на загружаемые данные, используя разнообразные разделители, а также преобразуя наборы символов.

В SQL*Loader могут применяться файлы пяти основных типов:

Входные файлы данных

Файлы операционной системы, содержащие фактические данные. Это могут быть файлы с записями фиксированной длины или файлы с разделителями.

Управляющий файл

Файл, содержащий всю информацию, необходимую SQL*Loader для загрузки данных из файла данных.

Файл журнала

В файле регистрируются результаты операций SQL*Loader.

Файл некорректных записей (bad file)

Создается, если ошибки возникают из-за некорректных данных во входных файлах. Этот файл хранит записи, содержащие некорректные данные (например, с несоответствующими типами данных, нарушениями ограничений и т. д.). Если работа SQL*Loader завершается успешно, то файл некорректных записей не создается. Каждому входному файлу может соответствовать свой собственный файл некорректных записей.

Файл отвергнутых записей (discard file)

Хранит все строки, отвергнутые в процессе выполнения операций как не соответствующие условиям инструкций WHEN. Наличие файла отвергнутых записей не обязательно. Каждому входному файлу может соответствовать свой собственный файл отвергнутых записей.

Подробную информацию об утилите SQL*Loader и ее возможностях можно найти в книге «Oracle SQL*Loader: The Definitive Guide» (Oracle SQL*Loader. Подробное руководство) Джонатана Генника (Jonathan Gennick) и Санжея Мишры (Sanjay Mishra), выпущенной издательством «O'Reilly & Associates», а также в документации Oracle.

Запуск SQL*Loader

SQL*Loader может вызываться из командной строки двумя способами. Если ввести

```
sqlldr
```

то SQL*Loader выведет перечень разрешенных параметров командной строки. Можно вызвать программу следующим образом:

```
sqlldr ключевое_слово=значение [ключевое_слово=значение ...]
```

например:

```
sqlldr system/manager CONTROL=product.ctl LOG=product.log
```

Можно указывать значения и без соответствующих ключевых слов, но в этом случае порядок значений должен полностью совпадать с тем порядком, в котором параметры перечислены в справочном файле или в документации Oracle. Поступая подобным образом, необходимо указать значения для всех параметров, от первого и до последнего. Можно комбинировать два подхода, например ввести имя пользователя без ключевого слова USERID (это самый первый параметр), а далее указывать пары *ключевое_слово=значение*.

Кроме того, можно задать все параметры командной строки в файле параметров и указать имя этого файла в параметре PARFILE командной строки sqlldr.

Параметры командной строки

Параметры командной строки разделяются запятыми, пробелами или и тем, и другим. Параметр *имя_файла* может задавать либо отдельное имя файла, либо имя файла с путем.

BAD = путь_к_файлу

Указывает имя файла некорректных записей. По умолчанию имя файла некорректных записей совпадает с именем управляющего файла, но имеет расширение *.bad*; этот файл записывается в тот же каталог, что и управляющий файл. Если задать другое имя, то расширение файла не изменится. Но если имя файла некорректных записей задавалось с параметром BAD, то по умолчанию каталогом хранения файла будет текущий рабочий каталог. Если загрузка данных осуществлялась из нескольких файлов, то имя файла некорректных записей будет сопоставлено только первому загружаемому файлу.

BINDSIZE = размер_массива_связывания

Определяет максимальный размер (в байтах) массива связывания. Этот параметр имеет более высокий приоритет, чем значение, полученное в результате вычислений с применением параметра ROWS. Размер массива связывания по умолчанию равен 65 536 байтам или 64K.

COLUMNARRAYROWS = целое

Определяет количество строк, выделяемых для массивов столбцов при загрузке в прямом режиме. Появился в Oracle9*i*.

CONTROL = имя_управляющего_файла

Определяет имя управляющего файла (может включать в себя и путь). По умолчанию файл имеет расширение *.ctl*. Если не указать управляющий файл, утилита запросит его у пользователя.

DATA = путь_к_файлу

Определяет имя файла с загружаемыми данными. По умолчанию имя файла совпадает с именем управляющего файла, но имеет расширение *.dat*. Если указать другое имя, то расширение файла не изменится. Если данные загружаются из нескольких файлов, то в этом параметре можно указать только имя первого файла. Имена остальных загружаемых файлов необходимо поместить в соответствующие инструкции INFILE управляющего файла.

DATE_CACHE = целое_число

Определяет размер кэша данных, который используется заданиями прямой загрузки для снижения накладных расходов на преобразование данных. Значение по умолчанию – 1000. Появился в Oracle9*i* Release 2.

*DIRECT = TRUE | **FALSE***

Определяет режим загрузки данных. При прямой загрузке данные помещаются непосредственно в файлы данных, команды INSERT для выполнения такой операции не создаются, что позволяет повысить производительность за счет снижения накладных расходов. В документации Oracle описаны ограничения на типы данных, для которых разрешена прямая загрузка, применение триггеров и ограничений в процессе загрузки, а также другие аналогичные вопросы.

Если параметр принимает значение FALSE, то устанавливается обычный режим загрузки, а если – TRUE, то осуществляется прямая загрузка. Значение по умолчанию – FALSE.

DISCARD = имя_файла

Задает имя файла отвергнутых записей. По умолчанию имя файла совпадает с именем управляющего файл, но имеет расширение *.dis*. Если указать другое имя, то расширение файла не изменится. Если данные загружаются из нескольких файлов, то файл с данным именем будет сопоставлен только первому загружаемому файлу.

DISCARDMAX = количество_логических_записей

Задает верхнюю границу для количества логических записей, которые могут быть отвергнуты, прежде чем процесс загрузки будет завершен. Когда количество отвергнутых записей достигает значения, указанного в параметре DISCARDMAX, загрузка заканчивается. По умолчанию количество отвергнутых записей не ограничено.

ERRORS = количество_ошибок

Устанавливает предельное количество ошибок, которое может быть допущено, прежде чем загрузка будет прервана. По умолчанию загрузка прерывается, если количество ошибок превышает 50.

*EXTERNAL_TABLE = **NOT_USED** | GENERATE_ONLY | EXECUTE*

Указывает, следует ли использовать для загрузки данных внешние таблицы. Значение по умолчанию, NOT_USED, выполняет загрузку в обычном или прямом режиме. GENERATE_ONLY формирует команды SQL, необходимые для загрузки, в файле журнала, где они могут быть отредактированы и запущены. EXECUTE создает команды SQL и выполняет их. Появился в Oracle9*i*.

FILE = имя_файла_данных_бд

Определяет имя файла данных БД, в котором выделяются экстенты. Этот параметр применяется в случае выполнения параллельных загрузок, для того чтобы обеспечить использование каждым сеансом загрузки собственного диска. Параметр применяется только при прямой загрузке.

LOAD = количество_логических_записей

Устанавливает ограничение для количества загружаемых логических записей. По умолчанию загружаются все записи.

LOG = имя_файла

Определяет имя файла журнала, который будет сформирован для сеанса загрузки. По умолчанию имя файла совпадает с именем управляющего файла, но имеет расширение *.log*; файл записывается в тот же каталог, что и управляющий файл. Если указать другое имя, то расширение файла не изменится. Но если для задания имени файла журнала применен параметр LOG, то файл журнала уже не будет автоматически записываться в каталог, содержащий управляющий файл, вместо этого он будет сохранен по указанному пути. Если путь не указан, то файл будет помещен в текущий рабочий каталог.

MULTITHREADING = TRUE | FALSE

Определяет, следует ли применять многопоточную обработку при прямой загрузке. В системах с одним процессором значение по умолчанию равно FALSE. В многопроцессорных системах значение по умолчанию равно TRUE. Появился в Oracle9*i*.

*PARALLEL = TRUE | **FALSE***

Указывает, применяется ли параллельная прямая загрузка. Если один и тот же объект загружается в нескольких сеансах прямой загрузки, то значение данного параметра должно быть равно TRUE. Значение по умолчанию – FALSE.

PARFILE = путь_имя_файла

Указывает SQL*Loader, что следует читать значения параметров командной строки из текстового файла. Такой текстовый файл называется *файлом параметров* и содержит пары *ключевое_слово=значение*. Обычно такие пары разделяются символами перевода строки. Применение параметра PARFILE позволяет избавиться от бесконечного повторного ввода значений в командную строку в случае, если необходимо несколько раз выполнить одну и ту же загрузку. Расширения по умолчанию для файла параметров не существует.

READSIZE = размер_буфера_считывания

Определяет размер буфера (в байтах), который SQL*Loader использует при считывании данных из файла ввода. Значение по умолчанию равно 64K. Значения пара-

метров READSIZE и BINDSIZE должны совпадать. Если для двух этих параметров указаны разные значения, то SQL*Loader увеличит значение меньшего из параметров так, чтобы они совпали.

ROWS = количество_строк

Точный смысл данного параметра зависит от того, какая загрузка выполняется – прямая или обычная. При обычной загрузке параметр служит для регулирования количества строк в массиве связывания. Речь идет о количестве строк, которое SQL*Loader загружает каждой командой INSERT, а также о частоте выполнения фиксации. По умолчанию загружаются 64 строки. Значение BINDSIZE должно быть достаточно большим, чтобы обеспечить хранение указанного количества строк. Если для хранения указанного количества строк требуется меньше пространства, чем определено параметром BINDSIZE, то объем памяти, занимаемый массивом связывания, уменьшается. Если же для хранения указанного количества строк требуется больше пространства, чем определено BINDSIZE, то массив связывания будет хранить меньше строк, чем запрошено.

При прямой загрузке параметр ROWS определяет количество строк, которые будут считываться из входного файла, прежде чем данные будут сохранены в БД. SQL*Loader округлит значение ROWS, чтобы оно совпадало с четным количеством блоков БД. По умолчанию в случае прямой загрузки выполняется всего одно сохранение при завершении загрузки.

SILENT = [(]ключевое_слово[,ключевое_слово...][)]

Позволяет подавить вывод различных заголовочных и поясняющих сообщений, которые утилита SQL*Loader обычно отображает в течение сеанса загрузки. Можно применять следующие ключевые слова:

ALL

Подавляет вывод всех сообщений. ALL – это эквивалент указания всех остальных ключевых слов вместе.

DISCARDS

Подавляет вывод сообщения, которое записывается в файл журнала для каждой отвергнутой записи.

ERRORS

Подавляет запись в файл журнала сообщений об ошибках, возникающих, когда запись в базу данных приводит к ошибке.

FEEDBACK

Подавляет вывод сообщений о достижении точки фиксации, которые отображаются каждый раз, когда SQL*Loader выполняет фиксацию или сохранение.

HEADER

Подавляет вывод начальных сообщений на экран, но не их запись в файл журнала.

PARTITIONS

Подавляет вывод статистики для разделов, записываемой в файл журнала при выполнении прямой загрузки секционированной таблицы.

SKIP = количество_логических_записей

Позволяет продолжить прерванную загрузку, пропустив указанное количество логических записей, добавляемых в БД. Если обычная загрузка заканчивается неудачно, то можно определить количество добавленных логических записей, по-

смотрев на последнее сообщение о достижении точки фиксации, выведенное на монитор. Если необходимо продолжить прямую загрузку для нескольких таблиц, то может потребоваться применение не параметра SKIP в командной строке, а инструкции CONTINUE_LOAD в управляющем файле. Эта инструкция позволит указать количество пропускаемых строк для каждой загружаемой таблицы.

*SKIP_INDEX_MAINTENANCE = TRUE | **FALSE***

Определяет, поддерживаются ли индексы при выполнении прямой загрузки. Данный параметр не действует для обычной загрузки. Если параметр принимает значение TRUE, то индексы не поддерживаются. Любые сегменты (разделы) индекса, которые должны были быть обновлены, помечаются как непригодные для использования. Значение FALSE обеспечивает нормальную поддержку индексов. Значение по умолчанию равно FALSE.

*SKIP_UNUSABLE_INDEXES = TRUE | **FALSE***

Определяет, каким образом выполняется загрузка таблиц, имеющих непригодные индексы. Значение TRUE вынуждает SQL*Loader загружать данные в таблицы, даже если эти таблицы имеют индексы, помеченные как непригодные для использования. Индексы останутся непригодными и после загрузки. Если как непригодный помечен индекс UNIQUE, то загрузка будет запрещена.

Значение FALSE приводит к тому, что SQL*Loader не вставляет те записи, которые привели бы к вставке значений в индекс, помеченный как непригодный для использования. Для обычной загрузки это означает, что любые записи, которые обновили бы непригодные индексы, отбрасываются как ошибочные. В случае прямой загрузки это означает, что загрузка будет прервана, как только встретится первая подобная запись. Значение по умолчанию равно FALSE.

STREAMSIZE = целое

Определяет размер (в байтах) для потоков прямой загрузки. Такая возможность появилась в Oracle9*i*.

USERID = {имя_пользователя[/пароль][@имя_сетевой_службы]|/}

Задает имя пользователя и пароль, которые будут указываться при подключении к базе данных. Значение *имя_сетевой_службы* позволяет (при желании) соединиться с удаленной базой данных. Для соединения с локальной БД с применением аутентификации операционной системы указывается символ косой черты (/). Если не ввести ни имя пользователя, ни пароль, то SQL*Loader запросит оба эти значения.

Управляющий файл

Управляющий файл сообщает SQL*Loader о том, как следует загружать данные. При формировании управляющего файла следует помещать каждый параметр на отдельную строку. В управляющий файл можно включать комментарии, предваряя их двойным тире (--).

Для того чтобы включить в управляющий файл зарезервированное слово SQL*Loader, необходимо заключить его в двойные кавычки. В Unix-системах можно предварять специальные символы знаком (\).

Основная команда, которая начинает управляющий файл, – это команда LOAD. Команда может содержать одну или несколько инструкций INFILE для определения файлов данных, а также одну или несколько инструкций INTO TABLE для указания таблиц, в которые будут загружаться данные.

Управляющий файл также может включать в себя данные для загрузки. Если такие данные включаются в управляющий файл, то они должны следовать за командой LOAD и начинаться с ключевого слова BEGINDATA.

В последующих разделах будут описаны основные ключевые слова управляющего файла, инструкции INFILE, INTO TABLE и некоторые другие, а также правила конкатенации, которые могут быть определены при помощи управляющего файла.

LOAD

```
[OPTIONS ключевое_слово/значение [, ключевое_слово/значение]...] [UNRECOVERABLE | RECOVERABLE]
   LOAD | CONTINUE_LOAD [DATA]
   [CHARACTERSET набор_символов] [BYTEORDER]
   [INFILE инструкция [INFILE инструкция...]]
   [MAXRECORDSIZE байтов]
   [READBUFFERS целое]
   [INSERT | APPEND | REPLACE | TRUNCATE]
   [правила_конкатенации]
   [PRESERVE BLANKS]
   INTO TABLE инструкция [INTO TABLE инструкция...]
   [BEGINDATA]
```

Определяет, какие данные будут загружаться, откуда и куда, и как следует интерпретировать загружаемые данные.

Ключевые слова

OPTIONS

Задает пары *ключевое_слово/значение* (которые были описаны в предыдущем разделе). Ключевые слова могут быть такими:

```
SKIP
LOAD
ERRORS
ROWS
BINDSIZE
SILENT
DIRECT
PARALLEL
READSIZE
```

Если пара *ключевое_слово/значение* указана и в командной строке, и в составе команды OPTIONS управляющего файла, то больший приоритет имеет значение, введенное в командной строке.

Если указать параметр командной строки, отсутствующий в данном списке, то SQL*Loader проигнорирует его, не выводя сообщение об ошибке.

Данное ключевое слово должно предшествовать в командном файле ключевому слову LOAD (подробную информацию о параметрах вы найдете в предыдущем разделе).

UNRECOVERABLE | ***RECOVERABLE***

Определяет, записывается ли информация о загрузке в журнал БД. Значение UNRECOVERABLE увеличивает производительность, но требует повторной загрузки данных в случае, если какие-то загружаемые файлы были потеряны, а затем восстановлены. Применяется только для прямой загрузки. Значение по умолчанию – RECOVERABLE.

LOAD [*DATA*] | *CONTINUE_LOAD* [*DATA*]

Основное ключевое слово для запуска команды LOAD. Если не удалось выполнить прямую загрузку и надо начать ее заново с точки сбоя, указав другое количество пропускаемых строк для каждой таблицы, ключевое слово LOAD следует заменить на CONTINUE_LOAD.

CHARACTERSET набор_символов

Задает набор символов, используемый для входных данных.

BYTEORDER

Определяет, что двоичные целые числа, включая указываемые в полях типа VARCHAR для задания длины поля, были сгенерированы на компьютере, порядок байтов на котором противоположен тому, который в настоящее время применяется в SQL*Loader. Данное ключевое слово доступно только в версии SQL*Loader для Oracle9*i*.

INFILE инструкция

Инструкция задает файл, содержащий данные для загрузки. В команде LOAD может быть одна или несколько инструкций INFILE. Синтаксис данной инструкции подробно описан в следующем разделе.

MAXRECORDSIZE байтов

Определяет максимальный размер логической записи в байтах.

READBUFFERS целое

Указывает количество буферов, которое будет использоваться при прямой загрузке. Значение по умолчанию равно 4. Количество буферов следует увеличивать только при появлении ошибки ORA-02374. Устарело в Oracle9*i*.

INSERT | *APPEND* | *REPLACE* | *TRUNCATE*

Определяет на глобальном уровне, что должно произойти с данными в загружаемых таблицах. Также имеется возможность задать поведение на уровне таблиц, а не глобально, посредством инструкции INFILE. Значение по умолчанию равно INSERT.

INSERT

Требует, чтобы к началу загрузки все таблицы были пустыми. Если какая-то таблица не будет пустой, то работа SQL*Loader будет прервана с сообщением об ошибке.

APPEND

Сохраняет все существующие данные в загружаемых таблицах.

REPLACE

Применяет SQL-команду DELETE для удаления всех данных, имеющихся в загружаемых таблицах.

TRUNCATE

Применяет SQL-команду TRUNCATE для удаления всех данных, имеющихся в загружаемых таблицах.

PRESERVE BLANKS

Определяет, следует ли сохранять начальные и завершающие пробелы в полях входных записей. Способ обработки пробелов в SQL*Loader зависит от того, какие данные загружаются: фиксированной ширины или данные с разделителями.

INTO TABLE инструкция

Указывает, в какую таблицу загружаются данные. Можно использовать несколько инструкций INTO TABLE, если планируется распределять загружаемые данные между несколькими таблицами.

BEGINDATA

Обозначает конец команды LOAD в том случае, если в управляющий файл включены данные. Тогда на строке, следующей сразу же за ключевым словом BEGINDATA, начинаются собственно данные для загрузки.

Инструкция INFILE

```
INFILE | INDDN имя_файла|* [параметры_зависящие_от_системы]
   [BADFILE | BADDN имя_файла_некорректных_записей]
   [DISCARDFILE | DISCARDDN имя_файла_отвергнутых записей]
   [DISCARDS | DISCARDMAX макс_отвергнутых]
   [FIELDS TERMINATED BY символ]
```

Указывает файл, содержащий данные для загрузки.

Ключевые слова BADDN, INDDN и DISCARDDN обеспечивают совместимость с синтаксисом DB2. Каждому входному файлу может соответствовать собственный файл некорректных записей и файл отвергнутых записей, а также собственное ограничение на количество отвергнутых записей. Если не указать параметр BADFILE или DISCARDFILE, а такой файл потребуется при выполнении задания, то SQL*Loader создаст файл с именем INFILE и расширением *.bad* или *.dis* соответственно.

Ключевые слова

имя_файла

Имя файла, содержащего данные для загрузки.

*

Указывает, что данные включены в управляющий файл.

BADFILE

Указывает местоположение файла некорректных записей, который создается, если ошибки вызваны некорректными данными в исходном файле.

DISCARDFILE

Указывает местоположение файла отвергнутых записей, содержащего все строки, отвергнутые в ходе работы.

DISCARDS, DISCARDMAX

Определяет максимальное количество записей, которое может быть отвергнуто, прежде чем загрузка будет завершена.

FIELDS TERMINATED BY

Данное ключевое слово должно быть включено в инструкцию FIELDS перед всеми полями для определения стандартного символа-разделителя полей. Глобальное значение может быть изменено в определениях отдельных полей или в указанных в INFILE зависящих от операционной системы параметрах. За более подробными сведениями о символах-разделителях обратитесь к описанию инструкций для типов данных.

Параметры_зависящие_от_системы

Описание параметров, относящихся к конкретной платформе, можно найти в документации Oracle.

Инструкция INTO TABLE

```
INTO TABLE имя_таблицы
   [PARTITION | SUBPARTITION (имя_раздела)]
   [{INSERT | REPLACE | TRUNCATE | APPEND}]
   [SORTED [INDEXES] (список_индексов)] [SINGLEROW]
   [{INSERT | REPLACE | TRUNCATE | APPEND}]
   [OPTIONS (FILE=имя_файла_БД)]
   [REENABLE [DISABLED_CONSTRAINTS][EXCEPTIONS имя_таблицы_исключений]]
   [WHEN условия_для_полей]
   [OID(имя_поля) | SID(имя_поля)]
   [FIELDS [описание_разделителя]]
   [TRAILING [NULLCOLS]]
   [SKIP количество_пропусков]
   (field_list)
```

Указывает таблицу, в которую загружаются данные. Для загрузки данных в несколько таблиц можно применить несколько инструкций INTO TABLE. Все переменные **имя_файла* могут содержать просто имя файла или же имя файла с путем.

Ключевые слова

имя_таблицы

Имя загружаемой таблицы. Дополнительно при желании можно указать имя владельца таблицы в формате *владелец.имя_таблицы*.

имя_раздела

Задает имя раздела или подраздела загружаемой таблицы. Инструкция PARTITION применяется в том случае, если загружается секционированная таблица и все данные попадают в один и тот же раздел.

INSERT | *REPLACE* | *TRUNCATE* | *APPEND*

Определяет метод загрузки на уровне таблиц. Значение данного параметра заменяет метод, выбранный для всей загрузки в целом (в команде LOAD). Данные ключевые слова могут появиться в инструкции INTO TABLE в двух местах: после спецификации PARTITION или после спецификации SORTED.

SORTED [*INDEXES*] (*список_индексов*)

Определяет имя одного или нескольких индексов в соответствии с порядком сортировки входных данных при прямой загрузке. Несколько имен индексов можно указать при помощи списка значений, разделенных запятыми.

SINGLEROW

Применяется только при прямой загрузке методом APPEND. Данное ключевое слово указывает, что каждая запись индекса должна быть вставлена прямо в индекс. Обычно при загрузке методом APPEND индексные записи накапливаются во временной области хранения и передаются в существующий индекс в конце загрузки.

имя_файла_БД

Имя файла данных в табличном пространстве загружаемой таблицы или раздела. Применимо только для параллельной прямой загрузки. При выполнении такой

загрузки SQL*Loader использует указанный файл данных для любых временных сегментов, которые создаются в процессе загрузки. Это позволяет выполнять несколько параллельных загрузок для одной и той же таблицы (раздела), но с использованием различных файлов данных.

REENABLE [*DISABLED_CONSTRAINTS*]

Вынуждает SQL*Loader вновь ввести в действие все отключенные ограничения при завершении прямой загрузки.

имя_таблицы_исключений

Имя таблицы, хранящей строки, нарушающие ограничения, которые были вновь введены в действие в результате применения инструкции REENABLE. Имя данной таблицы задается в команде SQL, которую SQL*Loader выполняет для того, чтобы ввести в действие ограничения.

условия_для_полей

Условия, которым должна соответствовать входная запись, чтобы быть загруженной. Условия для полей указываются аналогично тому, как это делается в инструкции WHERE SQL-команды SELECT. Подробная информация приведена далее в разделе «Инструкция для полей».

OID(*имя_поля*) | *SID*(*имя_поля*)

Поле входного файла, содержащее 32-разрядный идентификатор объекта, генерируемый Oracle в шестнадцатеричном формате, или системный идентификатор для вложенной таблицы. Данная инструкция применяется только при загрузке объектных таблиц. До выхода версии Oracle9*i* данная инструкция могла применяться только к обычной загрузке. Начиная с Oracle9*i* инструкция может применяться и для прямой загрузки.

описание_разделителя

Разделители и ограничивающие символы, служащие для разделения данных. Данная инструкция работает на уровне таблицы и должна использоваться, только когда одни и те же разделители и ограничивающие символы применяются для всех полей входного файла. Подробная информация приведена в разделе «Инструкция для полей».

TRAILING [*NULLCOLS*]

Применяется при загрузке данных с разделителями и указывает SQL*Loader, что отсутствующие поля в конце записи принимают значения NULL.

количество_пропусков

Количество логических записей, которое должно быть пропущено перед загрузкой. Инструкция SKIP применяется для продолжения закончившейся неудачно загрузки.

список_полей

Список полей в записи входных данных, включающей в себя типы данных и длины. Подробная информация приведена в разделе «Инструкция для полей».

Инструкция для полей

```
имя_столбца {[[BOUND] FILLER] | [сформированное]}
   [POSITION({начало|*[+сдвиг]}[{:|-}конец])]
   [инструкция_типа_данных] [PIECED]
    WHEN [(]условие[)] [AND [(]условие[)]...]
```

```
[NULLIF условие [AND условие...]]
[DEFAULTIF условие [AND условие...]]
["sql_выражение"]
```

Описывает поля, извлекаемые SQL*Loader из входной записи. Инструкция для полей заключается в скобки и может содержать описания одного или нескольких полей, разделенных запятыми.

Ключевые слова

имя_столбца

Имя столбца БД, в который предполагается загрузить данные из поля.

[*BOUND*] *FILLER*

Указывает, что поле в INFILE не загружается в таблицу. Данное ключевое слово появилось в Oracle8*i*. Поле BOUND FILLER, появившееся в Oracle9*i*, – это поле-заполнитель, которое может применяться с инструкциями NULLIF или DEFAULTIF.

сформированное

Приводит к тому, что SQL*Loader использует для поля сформированное значение. При формировании могут указываться следующие параметры:

RECNUM

Номер записи.

SYSDATE
CONSTANT "строка"
SEQUENCE

Аналогична RECNUM, но не включает в себя порядковые номера для некорректных записей. Спецификация SEQUENCE может содержать следующие элементы: ключевое слово COUNT, которое начинает последовательность с числа, превышающего количество записей, имеющихся в таблице; ключевое слово MAX, которое начинает последовательность с ближайшего целого, превышающего максимальное значение столбца; целое число, обозначающее начало последовательности; и приращение – положительное целое число (больше 1), указывающее, как должна увеличиваться последовательность для каждой следующей записи.

начало | *

Начальная позиция полей во входной записи. Первый байт соответствует позиции 1. Символ * означает, что поле начинается сразу же после предыдущего поля или с первого байта записи для первого поля.

сдвиг

Необязательный сдвиг, который SQL*Loader может добавить к позиции, представленной символом *.

конец

Позиция последнего байта данных в поле (можно не указывать). Если конечная позиция не задана, то SQL*Loader определит ее на основе информации из спецификации типа данных.

инструкция_типа_данных

Спецификация типа данных для поля. Может включать в себя информацию о разделителях и ограничивающих символах. Подробные сведения приведены в разделе «Инструкция для типов данных».

PIECED

Параметр вынуждает SQL*Loader загружать поля по одному. Применяется только при прямой загрузке и для последнего поля записи.

WHEN

Логическое условие, оцениваемое для каждого экземпляра поля. Условие для данной инструкции (как и для инструкций NULLIF и DEFAULTIF) состоит из имени поля или его позиции, оператора равенства или неравенства и строкового или шестнадцатеричного значения. Ключевое слово WHEN также может использовать для сравнения ключевое слово BLANKS, которое указывает, что поле полностью состоит из пробелов. Если запись не проходит данную проверку, то она помещается в файл отвергнутых записей.

NULLIF и DEFAULTIF

Логическое условие для инструкции NULLIF или DEFAULTIF. Если указанное условие выполнено, то данные инструкции вынуждают SQL*Loader записать в столбец БД NULL или 0 вместо того значения, которое на самом деле содержит поле. В инструкциях можно ссылаться на другие поля. Если запись не проходит данную проверку, то она помещается в файл отвергнутых записей.

Если какая-то из этих инструкций применяется для поля объекта или коллекции, то указывать уточненные имена всех полей инструкции необходимо при помощи точечной нотации.

sql_выражение

SQL-выражение, результат которого сохраняется в столбце БД. Можно применять любую разрешенную функцию Oracle SQL, любую пользовательскую функцию и любое поле, не являющееся заполнителем. Именам полей должен предшествовать символ двоеточия (:). До выхода версии Oracle9*i* прямая загрузка не выполняла такое выражение.

Инструкции для LOB, BFILE, объектов и коллекций

Для данных типа LOB, BFILE и объектных данных и коллекций необходимо применять специальный синтаксис SQL*Loader.

Синтаксис для данных LOB

Синтаксис загрузки из внешнего файла данных типа LOB:

```
имя_столбца LOBFILE (имя_поля | CONSTANT имя_файла
   [CHARACTERSET имя_набора_символов])
```

где *имя_поля* – это имя поля входного файла с именем *имя_файла* для данных LOB (обычно речь идет о поле-заполнителе). Если данные LOB загружаются из внешнего файла, то SQL*Loader не должен выделять память для их временного хранения.

Синтаксис для данных BFILE

Для данных, находящихся в файлах BFILE SQL*Loader применяется такой синтаксис:

```
имя_столбца BFILE({поле_каталога | CONSTANT имя_каталога},
   {поле_файла | CONSTANT имя_файла})
```

где *поле_каталога* или *имя_каталога* – это имена поля входного файла, которое содержит псевдоним каталога Oracle, а *поле_файла* или *имя_файла* – это имена файла операционной системы, который содержит данные.

Синтаксис для данных объектов

Для объектных данных следует применять ключевые слова OBJECT DATA после имени поля, а затем указать другой список полей вслед за ключевыми словами.

Синтаксис для коллекций

Для коллекций применяется такой синтаксис:

```
имя_столбца {VARRAY | NESTED TABLE | вспомогательный_файл_данных} {COUNT(имя_поля)
   | COUNT(CONSTANT целое)
```

где COUNT – это количество элементов коллекции, а *имя_поля* – имя поля входного файла, содержащего данные элементы. Значение *вспомогательный_файл_данных* – это имя внешнего файла, хранящего коллекцию в следующем формате:

```
SDF({имя_поля|CONSTANT 'имя_файла'}) зависящие_от_системы_параметры
   [MAXRECORDSIZE байтов] [CHARACTERSET 'имя_набора_символов']
```

где *имя_поля* – это имя поля входного файла, которое содержит имя внешнего файла. Сведения о *зависящих_от_системы_параметрах* содержатся в документации Oracle для вашей конкретной платформы.

Инструкция для типов данных

```
имя_типа_данных инструкции_типа_данных
```

Описывает тип данных входного файла.

Ключевые слова

имя_типа_данных

Имя одного из описанных далее типов данных. Для разных типов применяются разные инструкции, но две составляющие инструкции присутствуют для всех типов: *ограничители* (*enclosure*) и *окончание* (*termination*).

ограничители

Синтаксис данной инструкции таков:

```
[OPTIONAL] ENCLOSED [BY] ['строка' | X'шестнадцатеричное_значение'] [AND 'строка' |
X'шестнадцатеричное_значение']
```

Показать, что ограничивающие символы являются необязательными, можно посредством ключевого слова OPTIONAL, но только после инструкции TERMINATED BY. Ключевое слово AND указывает, что конечный символ отличается от начального символа.

окончание

Эта инструкция имеет такой синтаксис:

```
{DELIMITED [BY]}|{TERMINATED [BY]} WHITESPACE|X'шестнадцатеричное_значение' |
   'строка' | EOF
```

где WHITESPACE означает, что завершением поля считается любой символ-разделитель, включая символы пробелов, табуляции и новой строки. EOF обозначает конец файла и применяется в основном для загрузки данных LOB.

Типы данных имеют такой синтаксис:

```
CHAR длина ограничители окончание
DATE длина маска ограничители окончание
```

где *маска* – это маска даты, по которой функция Oracle TO_DATE осуществляет преобразование вводимой даты.

```
{INTEGER | DECIMAL | FLOAT | ZONED} [EXTERNAL]
 длина ограничители окончание
GRAPHIC [EXTERNAL] длина
```

где *длина* – это длина двоичного графического файла в двойных байтах. Ключевое слово EXTERNAL указывает, что графические данные заключены в символы перехода на нижний и на верхний регистры (shift-in и shift-out).

```
RAW длина
VARCHARC | VARRAWC (длина[,макс_байтов])
```

Для этих типов данных длина поля указывается в первых байтах поля. Количество байт, указывающих длину поля, – это *длина*, а значение *макс_байтов* определяет максимальное количество байт, допустимое для данного поля. VARRAWC загружает данные, но преобразование не выполняется.

Перечисленные типы данных называются *переносимыми* (*portable datatypes*), т. к. они не зависят от платформы. Кроме того, SQL*Loader поддерживает ряд *непереносимых* типов данных, двоичное представление которых зависит от конкретной платформы. Подробную информацию о таких непереносимых типах данных можно найти в документации Oracle.

Инструкции конкатенации

Правила конкатенации сообщают SQL*Loader, что следует объединить две или более записей входного файла в одну строку БД. Конкатенацию можно задать двумя способами: во-первых, при помощи инструкции CONCATENATE:

```
CONCATENATE число_записей
```

где *число_записей* указывает фиксированное количество записей во входном файле. Есть и другой способ – применить инструкцию CONTINUEIF.

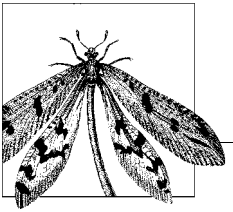
Ключевое слово THIS означает, что символ продолжения находится в текущей записи. Ключевое слово NEXT показывает, что символ продолжения находится в следующей записи, которая будет объединена с данной. Ключевое слово PRESERVE (появилось в Oracle8*i*) означает, что символы продолжения будут сохранены при загрузке данных. Остальная часть инструкции позволяет определять позиции и значения символов продолжения.

Для записей переменной длины применяется другая форма инструкции CONTINUEIF:

```
CONTINUEIF LAST [(]{= | <>}{'символ' | X'шестнадцатеричные_разряды'}[)]
```

Она сообщает SQL*Loader, что следует проверять последний символ записи на совпадение с символом продолжения. Если применяется эта форма инструкции, то символ продолжения всегда сохраняется в строке.

Если применяется конкатенация, то сведения о позиции полей относятся к объединенной строке.



15

Резервное копирование и восстановление

Не существует безукоризненных баз данных, не бывает и безотказных окружений. Эти неопровержимые факты вступают в противоречие с необходимостью абсолютной целостности и надежности информации в базе данных Oracle. Для предохранения БД Oracle и ее данных от непредвиденных проблем, которые могут возникнуть из-за потери части или всех данных, имеется ряд разнообразных средств резервного копирования и восстановления.

В главе будут описаны общеупотребительные способы резервного копирования и восстановления БД Oracle. Прежде чем углубляться в детали этих процессов, мы рассмотрим наиболее важные понятия резервного копирования и восстановления, а также сравним имеющиеся методы резервного копирования и восстановления БД.

Существует три основных метода резервного копирования и восстановления БД Oracle:

Export/Import

Oracle предлагает утилиты командной строки для экспортирования данных из БД (Export) и импортирования данных в БД (Import). В процессе экспорта формируются двоичные файлы, которые могут быть перенесены с одного компьютера на другой. Хотя утилиты Export и Import и могут применяться для выполнения резервного копирования и восстановления, они все же чаще применяются для перемещения данных из одной БД Oracle в другую. Работа с этими утилитами подробно рассмотрена в главе 13.

Пользовательские операции

Существует рекомендуемая Oracle последовательность операций для резервного копирования и восстановления БД при помощи комбинации команд операционной системы (для записи резервной копии БД) и SQL*Plus (для восстановления БД после такого резервирования). В случае применения этого подхода резервное копирование фактически происходит вне области экземпляра Oracle, поэтому может потребоваться наличие в файле журнала дополнительной информации об изменениях, происходивших с данными в процессе выполнения резервного копирования. Методы пользовательского резервного копирования и восстановления будут описаны далее в этой главе.

Recovery Manager

Компонент Oracle Recovery Manager, известный под именем RMAN, появился в Oracle8. RMAN работает внутри экземпляра Oracle, что избавляет от необходи-

мости ведения дополнительного журнала, необходимого в случае, если резервным копированием и восстановлением управляет пользователь. RMAN предоставляет дополнительные возможности, не поддерживаемые пользовательским резервным копированием и восстановлением. Так, применение RMAN позволяет выполнять инкрементное резервное копирование, при котором записывается резервная копия только изменившихся блоков, а не всего файла. Кроме того, RMAN предлагает упрощенный набор команд и может применяться с каталогом восстановления, что позволяет избавиться от некоторого количества накладных расходов на управление наборами резервных копий. RMAN будет описан ниже в этой главе.

В настоящее время корпорация Oracle настоятельно советует применять в качестве средства резервного копирования и восстановления RMAN. Поэтому некоторые новые возможности восстановления, такие как восстановление табличных пространств на заданный момент времени, доступны пока что только при помощи RMAN.

Основы резервного копирования и восстановления

Общие методы резервного копирования и восстановления Oracle можно разделить по ряду признаков, включая тип выполняемого действия и необходимые этапы его выполнения. Рассмотрим основные категории процессов резервного копирования и восстановления, поясняя вводимые понятия.

Восстановление носителя/экземпляра

К выходу из строя БД Oracle могут привести две причины: сбой экземпляра и отказ носителя. При *сбое экземпляра* (*instance failure*) происходит событие, которое не позволяет процессам, составляющим экземпляр БД, обслуживать запросы к базе данных. БД Oracle умеет автоматически восстанавливаться после сбоя такого типа при перезапуске экземпляра за счет повтора всех завершенных и отката всех незавершенных операций. Восстановление экземпляра выполняется автоматически, поэтому мы не будем говорить о нем в данной главе.

Возможен также *отказ носителя* (*media failure*) БД: повреждение или выход из строя базового хранилища файлов БД Oracle. Oracle не предусматривает автоматическое восстановление при таком сбое, поскольку, скорее всего, перед попыткой восстановления работоспособности БД вам придется выполнить некоторые другие операции, например заменить испорченный диск. Рассматриваемые далее операции резервного копирования и восстановления применяются именно при отказе носителя.

Согласованное/несогласованное

Резервное копирование в Oracle может быть согласованным или несогласованным. *Согласованное резервное копирование* (*consistent backup*) – это полное резервирование БД, находящейся в согласованном состоянии. При согласованном резервном копировании не требуется дополнительного изменения данных после восстановления. Обычно согласованное резервное копирование возможно лишь в том случае, если оно выполняется для БД в отключенном или в пассивном состоянии.

Несогласованное резервное копирование (*inconsistent backup*) – это резервирование, при котором данные изменяются после того, как выполнено исходное резервное копирование. При несогласованном резервном копировании для достижения согласованности БД после восстановления потребуется применить журнальные файлы. Так как сбои базы данных имеют неприятное свойство происходить в аб-

солютно неожиданные моменты, обычно восстановление бывает именно несогласованным.

Логическое /физическое

Одно из ключевых свойств любой реляционной базы данных заключается в возможности отделять логическое представление данных, организованных в таблицы и столбцы, от физической структуры таких данных в файлах на диске. *Логическое резервное копирование* (*logical backup*) БД Oracle – это резервирование данных с сохранением логической структуры данных в таблицах. *Физическое резервное копирование* (*physical backup*) БД Oracle – это резервирование, выполняемое для реальных файлов, таких как файлы данных и журнальные файлы, которые составляют полную базу данных.

Полное/инкрементное

Полное резервное копирование (*full backup*) – это резервирование всей БД на определенный момент времени. *Инкрементное резервное копирование* (*incremental backup*) затрагивает только те части БД, которые изменились с момента последнего полного или инкрементного резервного копирования. Инкрементная резервная копия почти всегда меньше по объему, чем полная, поэтому ее создание занимает меньше времени. Однако процесс восстановления работоспособности БД после инкрементного резервного копирования может оказаться долгим, т. к. сначала придется выполнить восстановление на основе полной копии, а затем применить все инкрементные резервные копии, лишь после этого можно будет инициировать любые другие процессы восстановления.

Вся база данных/табличное пространство

Резервная копия может включать в себя как всю БД, состоящую из всех файлов данных, архивных журнальных файлов и управляющих файлов, так и одно или несколько табличных пространств. БД Oracle может включать в себя множество табличных пространств, поэтому резервная копия одного табличного пространства может иметь меньший объем и, следовательно, будет создана быстрее. Резервное копирование может выполняться только для отдельных файлов данных или управляющих файлов.

Некоторые табличные пространства БД Oracle могут быть помечены как предназначенные только для чтения. *Резервное копирование табличного пространства, доступного только для чтения* (*read-only tablespace backup*), всегда будет согласованным, т. к. содержащиеся в нем данные не изменяются. Поэтому для табличного пространства, доступного только для чтения, можно записать резервную копию всего один раз, а затем исключить его из последующих процессов резервного копирования, что позволит ускорить их выполнение.

Упомянутая операция имеет смысл, только если резервное копирование табличного пространства выполняется после того, как оно было переведено в режим «только для чтения». Если такое табличное пространство преобразуется в обычное, то следует изменить процедуры резервного копирования и восстановления с тем, чтобы защитить данные, которые теперь могут подвергаться изменениям.

Завершенное/незавершенное восстановление

Процесс *завершенного*, или *полного, восстановления* (*complete recovery*) восстанавливает БД в ее максимально завершенном состоянии, предшествующем сбою. Незавершенное восстановление – это процесс восстановления, который невоз-

можно завершить. Существует ряд причин, по которым процесс восстановления не может быть завершен, например отсутствие журнального файла.

Начиная с версии Oracle8*i* пользователь имеет возможность выполнить для базы данных восстановление типа PITR (point-in-time recovery), т. е. такое, которое восстанавливает состояние на некоторый момент времени в прошлом. PITR удобно применять для восстановления состояния БД на момент времени, непосредственно предшествующий отсутствующему журнальному файлу или повреждению данных в результате ошибки пользователя (например, случайного удаления таблицы). Восстановление PITR может работать с моментом времени или системным номером изменения (SCN). Oracle9*i* поддерживает восстановление состояния на определенный момент времени и для отдельного табличного пространства.

Восстановление носителя на уровне файлов данных/блоков данных

Для всех версий, предшествующих Oracle9*i*, восстановление после отказа носителя было возможно только на уровне файлов данных. Начиная с Oracle9*i*, когда для резервирования и восстановления стал применяться менеджер RMAN, можно указать, что восстанавливать следует лишь те блоки файла данных, которые были утеряны. Восстановление носителя на уровне блоков данных может быть гораздо более быстрым, чем восстановление файлов, т. к. его объем обычно гораздо меньше.

Сравнение различных способов резервного копирования и восстановления в разрезе перечисленных выше характеристик приведено в табл. 15.1.

Таблица 15.1. Методы резервного копирования и восстановления

Характеристика	**Export/ Import**	**Пользовательские операции**	**RMAN**
Согласованное/ несогласованное	Согласованное, но только если активность БД по отношению к таблице останавливается или приостанавливается	Любое	Любое
Логическое/ физическое	Логическое	Физическое	Физическое
Полное/ инкрементное	Полное	Полное	Любое, возможно восстановление только изменившихся блоков данных
Вся база данных/табличное пространство	Только таблицы	Любое	Любое
Завершенное/незавершенное восстановление	Только завершенное	Любое	Любое
Восстановление носителя на уровне файлов данных/блоков данных	Восстановление на уровне таблиц	Только файлы данных	Любое

Пользовательское резервное копирование и восстановление

Пользовательское резервное копирование и восстановление – это процесс, позволяющий осуществить резервное копирование и восстановление БД Oracle за счет выполнения определенной комбинации команд операционной системы, команд SQL*Plus и команд специальных утилит Oracle.

В настоящее время описанный далее в этой главе Recovery Manager (RMAN) повсеместно заменяет пользовательские резервное копирование и восстановление, и корпорация Oracle больше не занимается усовершенствованием процессов пользовательского резервного копирования и восстановления. RMAN поддерживает всю функциональность пользовательского режима и предлагает ряд дополнительных возможностей, кроме того, с ним удобнее работать.

Тем не менее пользовательское резервное копирование и восстановление все еще не утратило своего значения, и по каким-то причинам вы можете выбрать для себя именно этот способ. Например, если вы поддерживаете базы данных Oracle более ранней версии, чем Oracle8, в которой появился RMAN; если ваши сотрудники лучше знакомы именно с давно известными процедурами пользовательского резервного копирования и восстановления; или же ваш административный план еще не включает в себя RMAN.

Пользовательское резервное копирование и восстановление может быть как согласованным, так и несогласованным. Выполняемые процессы зависят от типа резервного копирования.

Согласованное резервное копирование

Для выполнения согласованного резервного копирования необходимо стандартным образом завершить работу БД. Согласованное резервное копирование – это единственный тип резервного копирования, поддерживаемый в режиме NOARCHIVELOG (т. к. в этом режиме восстановление невозможно).

Несогласованное резервное копирование

Несогласованное резервное копирование можно выполнять как для выключенной, так и для работающей базы данных (или табличного пространства). Операции пользовательского резервного копирования выполняются операционной системой, поэтому процесс резервного копирования не подозревает о тех изменениях, которые могли быть внесены в БД или журнальные файлы процессами Oracle в ходе его работы. И поэтому для добавления соответствующей информации в журнальные файлы необходимо выполнять команды SQL, начинающие и завершающие пользовательское резервное копирование.

Выполнение пользовательского резервного копирования

Резервное копирование файлов данных, архивных журнальных файлов и файлов конфигурации может выполняться посредством соответствующих команд операционной системы. Для резервного копирования управляющих файлов и файлов инициализации применяются команды SQL*Plus.

В последующих разделах приведено поэтапное описание процесса пользовательского резервного копирования.

Выбираем файлы для резервного копирования

База данных Oracle постоянно изменяется, поэтому в первую очередь при выполнении пользовательского резервного копирования надо определить, что именно подлежит резервированию. Для того чтобы определить, какие файлы следует резервировать, можно выполнить ряд запросов к различным представлениям V$, как показано в табл. 15.2.

Таблица 15.2. V$ представления, применяемые при пользовательском резервном копировании

Тип файла	V$ представление	SQL
Файлы данных	V$DATAFILE	`SELECT NAME FROM V$DATAFILE;`
Оперативные журнальные файлы	V$LOGFILE	`SELECT MEMBER FROM V$LOGFILE;`
Архивные журнальные файлы	V$ARCHIVED_LOG	`SELECT THREAD#, SEQUENCE#, NAME FROM V$ARCHIVED_LOG;`
Управляющие файлы	V$CONTROLFILE	`SELECT NAME FROM V$CONTROLFILE;`

Необходимо сохранить список файлов данных и архивных журнальных файлов вместе с управляющим файлом. За время от выполнения резервного копирования до того момента, когда резервная копия будет использована, структура базы данных может измениться, тогда текущий управляющий файл новой БД может не соответствовать файлам, которые были зарезервированы.

При осуществлении пользовательского резервного копирования обрабатываются следующие типы файлов:

- Файлы данных
- Журнальные файлы
- Управляющие файлы
- Другие

Все эти файлы необходимы для операций восстановления, поэтому проследите за тем, чтобы команды пользовательского резервного копирования охватывали файлы всех четырех упомянутых типов.

Резервное копирование файлов данных

Согласованное пользовательское резервное копирование полной БД Oracle выполняется после того, как база данных остановлена с применением ключевого слова NORMAL, IMMEDIATE или TRANSACTIONAL. Остановив БД, можно выполнять команды операционной системы для резервирования или копирования файлов данных.

Можно резервировать отдельные табличные пространства и файлы данных. Если табличное пространство автономно, то для резервирования файлов данных применяются соответствующие команды операционной системы. Если же речь идет об оперативном табличном пространстве, то его необходимо сначала перевести в режим BACKUP, выдав следующую команду SQL*Plus:

```
ALTER TABLESPACE имя_табличного_пространства BEGIN BACKUP;
```

По завершении резервного копирования для возвращения к обычному режиму работы следует выполнить одну из приведенных ниже команд:

```
ALTER TABLESPACE имя_табличного_пространства END BACKUP;
ALTER DATABASE END BACKUP;
```

Запись дополнительной информации в журнальные файлы при работе в режиме BACKUP влечет за собой дополнительные накладные расходы, поэтому следует выходить из этого режима сразу, как только резервное копирование будет успешно завершено.

Если при сбое база данных находилась в режиме BACKUP, то может понадобиться выполнить команду ALTER DATABASE END BACKUP. Сервер Oracle не откроет заново табличное пространство в режиме BACKUP, поэтому придется смонтировать БД, выполнить эту команду, а затем уже открыть БД.

Иногда разумнее резервировать табличные пространства последовательно, а не все сразу. Резервируя табличные пространства по одному, вы избежите переключения всех табличных пространств в режим BACKUP на период полного резервного копирования; каждое табличное пространство будет находиться в этом режиме столько времени, сколько необходимо для его резервного копирования. Это уменьшит накладные расходы на запись дополнительной информации в журнальные файлы в режиме BACKUP.

Резервное копирование журнальных файлов

При работе в режиме ARCHIVELOG журнальные файлы архивируются, как только они заполнены. Архивирующий процесс автоматически записывает резервную копию заполненного журнала в другое место. У пользователя может возникнуть желание скопировать архивные файлы на резервный носитель для защиты от возможного повреждения носителя.

Обычно архивирование текущего журнала инициируется после выполнения резервного копирования файлов данных при помощи следующей команды SQL*Plus:

```
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

Резервное копирование управляющих файлов

Управляющий файл содержит важнейшую информацию о текущем состоянии базы данных. При выполнении пользовательского резервного копирования создавать резервную копию управляющего файла обязательно. Для этого надо выполнить следующую команду SQL*Plus:

```
ALTER DATABASE BACKUP CONTROLFILE TO имя_файла | TO TRACE
```

Резервную копию управляющего файла можно записать в двоичном формате в отдельный файл или в файл трассировки. Двоичная резервная копия управляющего файла содержит больше информации, вот почему именно этот метод рекомендуется корпорацией Oracle. При записи резервной копии управляющего файла в файл трассировки имеется возможность воссоздать управляющий файл из файла трассировки при помощи команды CREATE CONTROLFILE.

Резервное копирование других файлов

Важной составляющей Oracle-среды являются файлы инициализации (*INIT.ORA* и *SPFILE*), сетевые файлы (*TNSNAMES.ORA* и *LISTENER.ORA*) и файлы паролей.

Не забывайте о резервном копировании этих файлов как при выполнении обычного пользовательского резервного копирования, так и при их изменении.

Выполнение пользовательского восстановления

В случае отказа носителя пользователю для восстановления БД понадобятся файлы, созданные при выполнении пользовательского резервного копирования.

Восстановление файлов данных

Процесс восстановления для файлов данных состоит из трех основных этапов: определения файлов, подлежащих восстановлению, копирования соответствующих резервных копий и восстановления базы данных в работоспособном состоянии.

Выявление файлов для восстановления

Если управляющий файл не был поврежден в результате отказа носителя, то список данных, нуждающихся в восстановлении, можно получить, выполнив в SQL*Plus команду

```
SELECT * FROM V$RECOVER_FILE;
```

Такой запрос возвращает множество номеров файлов, подлежащих восстановлению.

Для того чтобы определить, какие файлы данных и табличные пространства соответствуют полученным номерам файлов, выполните такую команду:

```
SELECT d.NAME, t.NAME AS имя_табличного_пространства
    FROM V$DATAFILE d, V$TABLESPACE t
    WHERE d.TS# = t.TS# AND
    d.FILE# IN (номера_файлов);
```

где *номера_файлов* – это разделенный запятыми список номеров файлов, возвращенный предыдущим запросом.

Копирование резервных копий файлов

Определив, какие файлы подлежат восстановлению, необходимо скопировать резервные копии таких файлов. Необходимо перевести табличное пространство поврежденного файла данных в автономный режим и скопировать резервную копию в нужное место.

Если в результате отказа носителя каталог по умолчанию стал недоступен, то вам придется изменить управляющий файл, чтобы распознавались те новые каталоги, в которых сохранены резервные копии.

Восстановление базы данных

Записав резервную копию файла в нужный каталог, необходимо восстановить содержащее данный файл табличное пространство. Для восстановления базы данных выполните в SQL*Plus следующие шаги:

1. Убедитесь в том, что БД была остановлена:

   ```
   SQL> SHUTDOWN IMMEDIATE;
   ```

   или

   ```
   SQL> SHUTDOWN ABORT;
   ```

2. Скопируйте файл данных из хорошей резервной копии.
3. Запустите и смонтируйте базу данных.

   ```
   SQL> STARUP MOUNT;
   ```

4. Восстановите поврежденный файл данных, используя команду SQL*Plus:

```
SQL> RECOVER TABLESPACE <имя_табличного_пространства>;
```

или

```
SQL> RECOVER DATAFILE <имя_файла_данных>;
```

5. Откройте базу данных:

```
SQL> ALTER DATABASE OPEN;
```

Восстановление управляющего файла

Если в параметре инициализации CONTROL_FILES указано несколько копий управляющего файла, то можно восстановить одну из них, остановив базу данных командой SHUTDOWN ABORT, скопировав один из неповрежденных управляющих файлов для замены испорченной копии и снова запустив базу данных.

Если в результате отказа носителя место хранения управляющего файла стало недоступным, необходимо проделать те же операции и, кроме того, отредактировать параметр инициализации CONTROL_FILES, указав в нем новый каталог для хранения управляющего файла.

Если требуется восстановить единственную копию управляющего файла или же если были повреждены все копии мультиплексированного управляющего файла, то надо выполнить следующие шаги:

1. Остановить БД командой SHUTDOWN ABORT.
2. Скопировать резервную копию управляющего файла в соответствующий каталог.
3. Смонтировать БД командой STARTUP MOUNT.
4. Выполнить команду

```
RECOVER DATABASE USING BACKUP CONTROLFILE;
```

5. Применить архивные журнальные файлы в ответ на соответствующие приглашения (см. далее описание команды RECOVER).

Если архивный журнал недоступен или если для восстановления требуется оперативный журнал, а он недоступен, то придется выполнить незавершенное восстановление. Для выполнения незавершенного восстановления добавьте на предыдущем этапе в команду RECOVER ключевые слова UNTIL CANCEL.

6. Открыть базу данных и вернуть в начальное состояние журнальные файлы командой

```
ALTER DATABASE OPEN RESETLOGS;
```

По окончании процесса необходимо выполнить завершенное резервное копирование БД.

Восстановление табличного пространства на момент времени

Oracle предлагает и другой тип восстановления – *восстановление табличного пространства на определенный момент времени в прошлом* (*tablespace point-in-time recovery* – TSPITR). Восстановление TSPITR чаще всего применяется для переносимых табличных пространств после таких ошибок, как случайное удаление таблицы или табличного пространства или выявление логически поврежденных таблиц.

Применение переносимых табличных пространств – это эффективный способ перемещения табличных пространств из одной БД в другую. Вместо того чтобы резервировать, а затем восстанавливать табличное пространство или же перемещать данные посредством утилит Import и Export, можно копировать собственно файлы табличных пространств.

Выполнение такого восстановления состоит из следующих этапов:

1. Переведите табличные пространства, которые планируется восстановить, в автономный режим в исходной базе данных.
2. Создайте вспомогательную базу данных.
3. Восстановите табличное пространство во вспомогательной базе данных на нужный момент времени.
4. Сделайте обрабатываемые пространства доступными только для чтения во вспомогательной БД.
5. Экспортируйте информацию переносимого табличного пространства из вспомогательной БД.
6. Удалите табличное пространство в исходной БД.
7. Скопируйте файлы табличных пространств из вспомогательной БД в исходную.
8. Импортируйте информацию транспортабельного табличного пространства в исходную БД.
9. Используйте переносимые табличные пространства для перемещения восстановленного табличного пространства в исходную базу данных.
10. Разрешите доступ для чтения и записи для новых табличных пространств.

Команды резервного копирования и восстановления

Как уже говорилось, некоторые этапы пользовательского резервного копирования и восстановления реализуются при помощи команд операционной системы, команд Oracle SQL*Plus или же специальных команд утилит резервного копирования/восстановления Oracle. Информацию о соответствующих командах операционной системы можно найти в документации для каждой конкретной ОС.

Формат и назначение команд Oracle, применяемых для резервного копирования и восстановления, описаны в последующих разделах. Эти команды доступны для БД Oracle вне зависимости от конкретной операционной системы. Большинство из них является командами SQL*Plus. Имейте в виду, что в этой главе описаны только параметры команд, относящиеся к пользовательскому резервному копированию и восстановлению. За информацией о полном синтаксисе команд SQL*Plus обращайтесь к главе 12 (в конце книги есть алфавитный указатель, позволяющий определить, где можно найти дополнительную информацию о синтаксисе команды и ее описание).

Две из приведенных команд применяются только для резервного копирования и восстановления. Oracle предоставляет утилиты DBVERIFY и OCOPY для проверки корректности резервирования и резервного копирования *«чистых» разделов* (*raw partitions*) Windows соответственно (за подробной информацией об этих утилитах обращайтесь к документации Oracle).

ALTER DATABASE BACKUP CONTROLFILE

```
ALTER DATABASE BACKUP CONTROLFILE TO имя_файла | TO TRACE
```

Команда SQL*Plus. Создает резервную копию текущего управляющего файла базы данных.

Ключевые слова

имя_файла

Имя файла, в котором будет сохранена копия текущего управляющего файла.

TO TRACE

Приводит к тому, что информация из управляющего файла записывается в виде команд SQL в файл каталога трассировки БД. Вы можете затем использовать эту информацию для восстановления управляющего файла при помощи команды CREATE CONTROLFILE.

ALTER DATABASE OPEN

```
ALTER DATABASE OPEN имя_бд [NORESETLOGS | RESETLOGS]
```

Команда SQL*Plus. Открывает базу данных после выполнения пользовательского восстановления.

Ключевые слова

имя_бд

Определяет имя открываемой БД.

NORESETLOGS

Продолжает нумерацию архивных журналов с последнего использованного номера. Это значение по умолчанию, которое используется после завершенного восстановления БД.

RESETLOGS

Гарантирует, что после незавершенного восстановления или использования резервного управляющего файла старые архивные файлы не участвуют в восстановлении нового воплощения БД.

ALTER SYSTEM SUSPEND/RESUME

```
ALTER SYSTEM SUSPEND | RESUME
```

Команда SQL*Plus. Приостанавливает все операции ввода/вывода для БД, т. к. наличие такой паузы необходимо для некоторых сторонних программ.

Сторонние программы часто применяются для зеркалирования БД Oracle. Зеркалированную БД можно резервировать, сначала отделяя зеркало, а затем резервируя его, при этом не воздействуя на оперативную работу БД. Но некоторые мультиплексированные системы невозможно расщепить в процессе выполнения ввода/вывода, в таких случаях необходимо приостановить операции ввода/вывода на время отделения зеркала.

Обязательно переведите соответствующие табличные пространства в режим BACKUP перед остановкой ввода/вывода, а затем по возобновлении операций ввода/вывода – обратно в обычный режим работы.

Имейте в виду, что у команды ALTER SYSTEM существует также множество других параметров. Команда была подробно рассмотрена в главе 12 (обратитесь к алфавит-

ному указателю в конце книги для того, чтобы посмотреть, где описаны дополнительные ключевые слова команды).

Ключевые слова

SUSPEND Приостанавливает все операции ввода/вывода для всех файлов БД.

RESUME Восстанавливает операции ввода/вывода для всех файлов БД.

CREATE CONTROLFILE

```
CREATE CONTROLFILE [REUSE] [SET]
    DATABASE имя_бд
    {LOGFILE {[GROUP целое] имя_файла}...}
    [RESETLOGS | NORESETLOGS]
    [DATAFILE имя_файла]
    [MAXLOGFILES целое]
    [MAXLOGMEMBERS целое]
    [MAXLOGHISTORY целое]
    [МАХФАЙЛЫ ДАННЫХ целое]
    [MAXINSTANCES целое]
    [ARCHIVELOG | NOARCHIVELOG]
    [CHARACTER SET набор_символов]
```

Команда SQL*Plus. Позволяет создать заново управляющий файл: пользователь может ввести соответствующие значения для управляющего файла вручную или же использовать текстовый файл, созданный ранее командой BACKUP CONTROLFILE TO TRACE (которая обсуждалась чуть раньше).

Если для каких-то параметров в команде CREATE CONTROLFILE значения не будут указаны, то сервер Oracle возьмет значения по умолчанию.

Ключевые слова

REUSE

Приводит к тому, что управляющий файл, имя которого указано в параметре инициализации CONTROL_FILE, становится новым управляющим файлом. Если указанный в файле инициализации управляющий файл существует, а ключевое слово REUSE не указано, то возвращается ошибка.

SET

Означает, что новым именем базы данных будет *имя_бд*.

DATABASE имя_бд

Определяет имя базы данных для управляющего файла.

LOGFILE имя_файла

Указывает имя журнального файла БД. Необходимо перечислить все элементы всех журнальных групп.

GROUP целое

Определяет номер журнальной группы.

RESETLOGS | ***NORESETLOGS***

Определяет, должна ли база данных игнорировать журнальные файлы, перечисленные в соответствующей инструкции. Эти ключевые слова применяются для предотвращения конфликтов журнальных файлов при незавершенном восстановлении.

DATAFILE имя_файла

Указывает файлы данных для БД. Необходимо перечислить все файлы данных для БД.

MAXLOGFILES целое

Определяет максимальное количество журнальных файлов, которые могут быть созданы для БД.

MAXLOGMEMBERS целое

Определяет максимальное количество элементов для файла журнала. Элемент – это идентичная копия журнального файла.

MAXLOGHISTORY целое

Определяет максимальное количество групп архивных журнальных файлов для автоматического восстановления носителя в случае применения компонента Oracle Parallel Server или Real Application Clusters.

MAXDATAFILES целое

Задает максимальное количество файлов данных, с которыми изначально работает база данных. Это значение применяется для определения размера секции файлов данных управляющего файла.

MAXINSTANCES целое

Определяет максимальное количество экземпляров, для которых одновременно БД может быть смонтирована и открыта.

ARCHIVELOG | ***NOARCHIVELOG***

Определяет, будет ли БД работать в режиме ARCHIVELOG. По умолчанию присваивается значение NOARCHIVELOG.

CHARACTER SET набор_символов

Задает набор символов, который воспроизводится в управляющем файле, где он может использоваться для корректной интерпретации имен табличных пространств.

DBVERIFY

```
DBV FILE=имя_файла
```

Утилита командной строки DBVERIFY предназначена специально для проверки корректности резервного копирования.

Ключевые слова

имя_файла

Имя проверяемого файла резервной копии.

OCOPY

```
OCOPY исходный файл файл_назначения |
OCOPY /b исходный_файл устройство_назначения |
OCOPY /r исходное_устройство раздел_назначения |
```

Утилита командной строки OCOPY предназначена специально для резервного копирования неформатированных разделов в Windows-системах.

Ключевые слова

исходный файл

Имя файла на неформатированном разделе.

файл_назначения

Имя файла, в который будет записана резервная копия.

/b

Указывает, что резервная копия может быть разделена на части и записана на несколько дискет.

устройство_назначения

Дисковое устройство, на которое будет записана резервная копия.

/r

Задает восстановление раздела, резервное копирование которого было выполнено частями на нескольких носителях.

исходное_устройство

Дисковое устройство для чтения дискет с резервной копией.

раздел_назначения

Имя раздела, который будет восстановлен из резервной копии на нескольких носителях.

RECOVER

```
RECOVER {общее | управляемое | END BACKUP}

общее :=
[AUTOMATIC] [FROM каталог]
    {{параметры_полного_восстановления_бд |
    параметры_частичного_восстановления_бд |
    LOGFILE имя_файла}
    {[TEST | ALLOW целое CORRUPTION]} ... |
    CONTINUE [DEFAULT] | CANCEL}
    [PARALLEL [целое]]

параметры_полного_восстановления_бд :=
[STANDBY] DATABASE
    [{UNTIL {CANCEL | TIME дата | CHANGE целое} |
    USING BACKUP CONTROLFILE]

параметры_частичного_восстановления_бд :=
TABLESPACE имя_табличного_пространства [, имя_табличного_пространства] . . . |
    DATAFILE имя_файла_данных [, имя_файла_данных ] . . . |
    STANDBY {TABLESPACE имя_табличного_пространства [, имя_табличного_пространства] . . . |
        DATAFILE имя_файла_данных [, имя_файла_данных ] . . .}
        UNTIL [CONSISTENT WITH] CONTROLFILE

управляемое :=
    MANAGED STANDBY DATABASE
    [{NODELAY | [TIMEOUT] целое | CANCEL [IMMEDIATE]
    [NOWAIT]} |
    [DISCONNECT [FROM SESSION]] [FINISH [NOWAIT]]]
```

Команда SQL*Plus. Применяет к файлу данных соответствующие журнальные файлы для восстановления БД. Эта операция повторно выполняет все завершенные транзакции из журналов, затем откатывает незафиксированные транзакции.

Для выполнения аналогичных действий также может применяться параметр RECOVER SQL-команды ALTER DATABASE.

Ключевые слова

END BACKUP

Выводит БД из режима оперативного резервирования.

AUTOMATIC

Приводит к автоматическому формированию имени следующего архивного журнального файла на основе значений параметров LOG_ARCHIVE_DEST и LOG_ARCHIVE_FORMAT. Если файла, соответствующего автоматически сформированному имени, не существует, то SQL*Plus запрашивает имя файла у пользователя. Эта же функциональность реализуется командой SET AUTOBACKUP ON.

FROM каталог

Место хранения архивной журнальной группы. Если параметр не задан, то SQL*Plus использует значение параметра инициализации LOG_ARCHIVE_DEST.

LOGFILE имя_файла

Определяет имя журнального файла, который будет участвовать в восстановлении.

TEST

Позволяет запустить тестовое восстановление для того, чтобы определить, нет ли проблем с файлами, которые будут участвовать в процессе восстановления. Тестовое восстановление считывает нужные файлы, но не записывает изменения на диск.

ALLOW целое CORRUPTION

Указывает, сколько поврежденных блоков может быть обнаружено, прежде чем полное или тестовое восстановление будет отменено.

CONTINUE

Применяется для продолжения многоэкземплярного восстановления после прерывания процесса для отключения потока.

CONTINUE DEFAULT

Указывает, что восстановление должно продолжаться с применением автоматически сформированных имен архивных журнальных файлов.

CANCEL

Отменяет текущий процесс восстановления.

PARALLEL [*целое*]

Разрешает параллельное восстановление. Существуют ситуации, когда установка этого параметра не помогает (например, если проблема с нехваткой ресурсов для ввода/вывода связана с используемым дисковым оборудованием). Если *целое* не указано, сервер Oracle вычислит степень параллелизма на основе количества процессоров и значения параметра PARALLEL_THREADS_PER_CPU.

DATABASE

Восстанавливает всю БД целиком.

STANDBY DATABASE

Восстанавливает резервную БД при помощи управляющего файла и архивных журнальных файлов.

UNTIL CANCEL | TIME | CHANGE

Означает, что процесс восстановления будет продолжаться до тех пор, пока не будет введена команда ALTER DATABASE RECOVER CANCEL, не наступит определенное время или не произойдет изменение с указанным номером SCN.

TABLESPACE

После данного ключевого слова указываются имена табличных пространств, которые следует восстановить.

DATAFILE

После данного ключевого слова указываются имена файлов данных, которые следует восстановить.

UNTIL CONSISTENT WITH CONTROLFILE

Означает, что восстановление должно продолжаться до тех пор, пока файл данных или табличное пространства не будет соответствовать текущему управляющему файлу резервной БД.

MANAGED STANDBY DATABASE

Включает режим управляемого восстановления резервной БД.

NODELAY

Подменяет любые имеющиеся настройки DELAY для незамедлительного применения архивного журнального файла к резервной БД.

TIMEOUT целое

Задает период времени (в минутах), в течение которого операция управляемого восстановления ожидает появления архивного журнального файла. По истечении этого периода времени операция восстановления завершается, и вам придется заново выдать команду RECOVER для повторного запуска операции управляемого восстановления.

CANCEL

Завершает операцию управляемого восстановления и возвращает управление сеансу после завершения процесса восстановления.

CANCEL IMMEDIATE

Завершает операцию управляемого восстановления после применения всей журнальной информации текущего журнала или после чтения следующего журнального файла. Возвращает управление сеансу после завершения процесса восстановления.

CANCEL IMMEDIATE NOWAIT

Аналогично CANCEL IMMEDIATE, с тем лишь отличием, что управление возвращается незамедлительно.

CANCEL NOWAIT

Завершает операцию управляемого восстановления после чтения следующего журнального файла и незамедлительно возвращает управление сеансу.

DISCONNECT [*FROM SESSION*]

Запускает управляемое восстановление как фоновый процесс.

FINISH

Приводит к тому, что управляемый процесс восстанавливает текущие журнальные файлы резервной БД. Если применяется в сочетании с ключевым словом NOWAIT, то управление незамедлительно возвращается сеансу.

STARTUP MOUNT

```
STARTUP MOUNT имя_бд
```

Команда SQL*Plus. Предназначена для монтирования базы данных без ее открытия. Эту команду следует применять для операций пользовательского восстановления.

Ключевые слова

имя_бд Имя монтируемой базы данных.

Recovery Manager (RMAN)

Утилита Recovery Manager (RMAN), появившаяся в версии Oracle8, позволяет создать резервную копию БД Oracle и восстановить БД Oracle из резервной копии. Многие операции RMAN можно выполнить и другими методами, описанными в этой главе, но RMAN обладает и рядом существенных преимуществ, а именно поддерживает следующие функции (в отличие от пользовательского резервного копирования и восстановления):

- Выполнение резервного копирования без перевода базы данных в режим резервного копирования.
- Осуществление инкрементного резервного копирования только для изменившихся блоков данных
- Автоматическое отслеживание и журналирование операций резервного копирования
- Автоматическое выявление повреждений блоков в процессе резервного копирования
- Ведение протокола операции резервного копирования
- Выполнение восстановления носителя на уровне блоков

Бесспорно, что именно RMAN представляет собой стратегию будущего для резервного копирования и восстановления. Недавнее расширение корпорацией Oracle возможностей резервного копирования, а именно восстановление носителя на уровне блоков и восстановление табличного пространства на момент времени, коснулось только RMAN.

Последующие разделы посвящены применению RMAN. Мы вкратце рассмотрим основные принципы работы, параметры командной строки, применяемые для запуска RMAN и работы с ней, а также пример командного сценария, с помощью которого можно выполнить резервное копирование БД Oracle. Затем будут описаны команды RMAN.

Основы RMAN

Для того чтобы лучше понимать последующие разделы, необходимо поближе познакомиться с несколькими новыми концепциями RMAN:

Серверные процессы

RMAN в своей работе использует ряд серверных процессов. Они стартуют при каждом запуске RMAN и обращении к одной из его функций, например, при подключении к каталогу или при выделении канала для резервного копирования.

Канал

Канал (*channel*) – это серверный процесс RMAN, взаимодействующий с определенным устройством резервного копирования. Выделение канала должно предшествовать любой операции резервного копирования.

Целевая БД и каталог

RMAN может взаимодействовать с двумя разными БД Oracle. *Целевая БД* (*target DB*) – это база данных, для которой выполняются операции резервного копирования или восстановления. *Каталог* (*catalog*) – это база данных, выполняющая функции хранилища информации о действиях RMAN. Обычно в роли каталога выступает другая БД Oracle, хранящаяся на другом хосте. Выделение для этой цели другого компьютера позволяет избежать неприятностей, связанных с разрушением той информации из каталога, которая необходима для восстановления после сбоя.

Каталог содержит специальную схему для хранения нужной информации. Для создания каталога RMAN необходимо сначала подключиться к базе данных каталога средствами RMAN и выполнить команду CREATE CATALOG. Если надо работать с каталогом для RMAN, то необходимо зарегистрировать целевые БД при помощи команды RMAN REGISTER, для того чтобы эти БД получили доступ к каталогу.

Использовать каталог не обязательно, т. к. RMAN включает информацию о любом виде резервного копирования в управляющие файлы целевой базы данных. Однако каталог позволяет сделать управление более гибким, благодаря хранению большего объема исторических данных обо всех операциях резервного копирования, произведенных для БД.

Media Management Layer

Если при резервном копировании с помощью RMAN вы используете накопитель на магнитной ленте, то для взаимодействия с таким устройством вам придется воспользоваться дополнительным программным обеспечением Media Management Layer (MML), которое обычно поставляется продавцом оборудования.

Наборы резервных копий

При выполнении резервного копирования средствами RMAN вы фактически создаете один или несколько *наборов резервных копий* (*recovery sets*). Набор резервных копий – это логическая сущность, состоящая из одного или нескольких физических элементов резервного копирования – физических файлов. Файлы данных и управляющие файлы могут входить в один набор резервных копий, но архивные журнальные файлы должны относиться к другому набору. Восстановить БД на основе набора резервных копий можно только при помощи RMAN.

Запуск RMAN

Для того чтобы запустить RMAN, необходимо обладать привилегией SYSDBA, полученной при аутентификации в операционной системе или посредством файла паролей. Если аутентификация выполняется средствами ОС, то необходимо также корректно задать две переменные окружения:

ORACLE_SID
ORACLE_HOME

Для запуска Recovery Manager следует вызвать исполняемый файл RMAN в командной строке:

```
> RMAN
```

RMAN выведет приглашение на ввод:

```
RMAN>
```

Подключиться к целевой БД и к каталогу восстановления позволяет команда CONNECT, описанная ниже в этом разделе.

При вызове исполняемого файла RMAN можно указать следующие параметры:

TARGET

Задает строку соединения с целевой БД. Если строка соединения не указана, то RMAN подключается к БД Oracle, содержащейся в переменной окружения ORACLE_SID.

CATALOG

Задает строку соединения с базой данных, служащей каталогом восстановления. Если не указать каталог восстановления при подключении к RMAN или внутри самой утилиты, то каталог не будет участвовать в операциях резервного копирования.

NOCATALOG

Указывает, что RMAN будет запускаться без использования определенного каталога восстановления. В Oracle9*i* это поведение по умолчанию.

AUXILARY

Задает строку соединения с базой данных каталога восстановления. Вспомогательные БД служат для создания дублирующих резервных БД (см. описание команды DUPLICATE далее в этом же разделе) или для реализации восстановления табличного пространства на момент времени.

CMDFILE

Задает строку, содержащую имя файла с командами RMAN. Когда все команды файла выполнены, RMAN завершает свою работу.

LOG

Задает строку, содержащую путь и имя файла для журнала выходных сообщений.

MSGNO

Означает, что любые выводимые командой данные должны включать в себя номер сообщения RMAN.

APPEND

Указывает, что RMAN должен добавлять новые сообщения в конец существующего файла сообщений. Если данное ключевое слово не указано, а имя журнального файла совпадает с именем уже существующего файла, то существующий файл перезаписывается.

DEBUG

Активирует режим отладки для RMAN, в котором для каждой команды RMAN выводятся поясняющие сообщения. В Oracle9*i* пользователь имеет возможность указать тип и степень детализации отладочных сообщений.

TRACE

Задает строку, содержащую путь и имя файла для вывода отладочных сообщений.

SEND

Задает строку, содержащую команду, отправляемую по всем выделенным каналам. Такая функциональность поддерживается службой управления носителями MML для определенного устройства.

PIPE

Задает строку, содержащую имя канала, который может применяться для передачи команд в RMAN. Каналы можно использовать в сочетании со встроенным пакетом PL/SQL DBMS_PIPE. Параметр появился в Oracle9*i*.

TIMEOUT

Определяет период времени (в секундах), в течение которого канал ожидает поступления входных данных. По истечении данного промежутка времени RMAN завершает работу. Параметр появился в Oracle9*i*.

Если в составе команды RMAN указано несколько приведенных выше ключевых слов, то их следует разделять пробелами, например:

```
> RMAN TARGET SYS/syspassword NOCATALOG
> RMAN TARGET / CATALOG SYS/syspassword
> RMAN TARGET SYS/syspassword CATALOG CAT/meow APPEND
```

Применение сценариев RMAN

В процессе задания операций резервного копирования или восстановления, которые должны быть выполнены посредством RMAN, часто приходится применять различные команды. Если при вводе этих команд допущена ошибка, то RMAN прервет выполнение всего задания. Поэтому многие администраторы баз данных предпочитают работать со сценариями, содержащими готовые корректные наборы команд RMAN.

Можно создать сценарии операционной системы, содержащие все команды, необходимые для запуска задания RMAN, начиная от установки переменных окружения и заканчивая собственно резервным копированием или восстановлением БД. Также могут понадобиться сценарии ОС для передачи переменных в RMAN. Синтаксис таких сценариев соответствует стандартным правилам для сценариев операционной системы.

Кроме того, можно создавать сценарии непосредственно внутри RMAN. Команда @ вызывает сценарий ОС из RMAN. Команда @@ вызывает другой сценарий из уже вызванного сценария. Команда @@ должна ссылаться на сценарий, который хранится в том же каталоге, что и первоначально вызванный сценарий.

Для сохранения сценария в каталоге восстановления предназначена команда:

```
RMAN> CREATE SCRIPT имя_сценария {тело_сценария}
```

Для редактирования существующего сценария применяется аналогичный синтаксис с ключевыми словами REPLACE SCRIPT.

Для запуска сценария, хранящегося в каталоге, следует выполнить команду:

```
RMAN> RUN {EXECUTE SCRIPT имя_сценария;}
```

Удалить сценарий из каталога можно такой командой:

```
RMAN> DELETE SCRIPT имя_сценария;
```

Есть два способа, позволяющих просмотреть сценарий, сохраненный в каталоге. Можно использовать в RMAN команду PRINT, указав имя сценария, а можно выпол-

нить запрос к таблицам, в которых хранится сценарий. Приведем пример SQL-запроса к БД каталога, который извлекает все хранящиеся в БД сценарии:

```
SELECT a.script_name, a.text
    FROM rc_stored_script_line a, rc_stored_script b
    WHERE a.db_key = b.db_key
```

RMAN включает в себя команды, управляющие работой БД Oracle, но имеется также возможность вызвать любую команду SQL, указав внутри RMAN ключевое слово SQL, а за ним – строку с SQL-выражением, заключенную в кавычки.

Применение команд RMAN

Основное предназначение утилиты Recovery Manager состоит в том, чтобы упростить выполнение операций резервного копирования и восстановления. Если применять Oracle9*i* Recovery Manager, то для выполнения полного оперативного резервного копирования БД потребуется совсем несложная команда. Предположим, что вы уже подключились к целевой БД и хотите применить для всех параметров значения по умолчанию, тогда команда будет такой:

```
RMAN> BACKUP DATABASE FORMAT
2> '/d99/rmanback/brdstn/rman_%d_%t_%U.bus';
```

В Oracle8*i* RMAN необходимо использовать команду RUN и явно выделить канал. Такое же резервное копирование, как и в предыдущем примере, в Oracle8*i* выполняется посредством более сложного набора команд:

```
RMAN> RUN {
2> ALLOCATE CHANNEL d1 TYPE DISK;
3> BACKUP DATABASE FORMAT
4> '/d99/rmanback/brdstn/rman_%d_%t_%U.bus';
5> }
```

Обычное восстановление при помощи RMAN в Oracle9*i* осуществляется следующими командами (предполагается, что БД уже закрыта):

```
RMAN> RESTORE DATABASE;
RMAN> RECOVER DATABASE;
RMAN> ALTER DATABASE OPEN;
```

Такое же восстановление посредством RMAN в Oracle8*i* выполняется такими командами:

```
RMAN> RUN{
2> ALLOCATE CHANNEL d1 TYPE DISK;
3> RESTORE DATABASE;
4> RECOVER DATABASE;
5> ALTER DATABASE OPEN; }
```

Как будет видно из представленных ниже описаний команд, RMAN обеспечивает чрезвычайную гибкость при выборе способа резервного копирования и восстановления.

Общие параметры и идентификаторы RMAN

Некоторые ключевые слова и параметры используются многими командами RMAN. В последующих разделах при рассмотрении синтаксиса команд общие ключевые слова будут упоминаться, но подробное их описание будет дано только в данном разделе.

id_канала

Чувствительное к регистру имя канала, следующее за ключевым словом CHANNEL.

спецификация_строки_соединения

Выполняет соединение с БД Oracle при помощи RMAN; имеет такой вид:

```
['][имя_пользователя][/[пароль]][@имя_сетевой_службы]['] 
```

спецификация_файла_данных

Определяет файл данных по его полному уточненному имени или номеру, которые задаются следующим образом: '*имя_файла*' | *номер_файла*

устройство

Тип хранилища, в которое будет записываться результат резервирования или копирования. Значением может быть DISK или же имя устройства-носителя, указанное поставщиком применяемого ленточного устройства резервирования.

имя_файла

Имя файла, включаемое в команды RMAN. Может содержать в себе полный путь к файлу. Если имя не включает в себя путь, то RMAN считает, что файл хранится в каталоге запуска RMAN.

строка_формата

Строка (заключенная в кавычки), задающая формат некоторого множества имен файлов. Так как большинство имен файлов для элементов резервирования должно быть уникальными, *строка_формата* обычно включает в себя путь к файлу и один или несколько групповых символов. Групповые символы предваряются символом % и обозначают уникальное сформированное значение (%u) или нечто более значащее, например номер копии или значение, формируемое на основе даты. За более подробной информацией о строках формата обратитесь к документации RMAN.

указатель_носителя

Полное имя фрагмента резервной копии.

PARMS

Параметры для выделенного недискового устройства в формате PARMS '*параметры_канала*' или PARMS='*параметры_канала*' (см. также документацию RMAN).

первичный_ключ

Значение уникального первичного ключа, присвоенное элементу резервирования.

имя_табличного_пространства

Одно или несколько имен табличных пространств рассматриваемой БД Oracle.

имя_тега

Одно или несколько имен тегов, которые идентифицируют копии файлов данных. Если существует несколько копий с тегами, то RMAN использует самую «свежую». Имя тега назначается в процессе резервного копирования.

Общие инструкции RMAN

Рассмотренные далее инструкции могут присутствовать в различных командах RMAN. Впоследствии при описании конкретных команд подробное описание общих инструкций из данного раздела не будет приводиться повторно.

Инструкция ПараметрыКанала

```
ПараметрыКанала :=
{ PARMS [=] 'параметры_канала'
| CONNECT [=]спецификация_строки_соединения
| DEBUG [=] целое
| FORMAT [=] 'строка_формата' [,'строка_формата' ...]
| TRACE [=] целое
| { MAXPIECESIZE [=] целое
  | RATE [=] целое } [ K | M | G ]
| MAXOPENFILES [=] целое
| SEND 'команда'
}
```

Позволяет изменить характеристики канала.

Ключевые слова

спецификация_строки_соединения

Указывает альтернативную целевую БД для резервного копирования и восстановления. Синтаксис инструкции описан в предыдущем разделе.

DEBUG целое

Записывает отладочную информацию в выходной файл для команд резервирования, копирования или восстановления, использующих данный канал. Степень детализации записываемой информации определяется значением *целое*.

TRACE целое

Определяет уровень детализации трассировочной информации, записываемой в выходной файл. Точный смысл значения *целое* определяется программным обеспечением MML.

MAXPIECESIZE целое

Указывает максимальный размер элемента резервной копии. Появилось в Oracle9*i*.

RATE целое

Указывает максимальный объем, который RMAN может считывать из канала в байтах, килобайтах (K), мегабайтах (M) или гигабайтах (G). Появилось в Oracle9*i*.

MAXOPENFILES целое

Определяет максимальное количество файлов, которые RMAN может держать открытыми одновременно. Появилось в Oracle9*i*.

SEND команда

Отправляет команду (зависящую от производителя) всем выделенным каналам. Появилось в Oracle9*i*.

Инструкция СпецификацияАрхЖурнала

```
ARCHIVELOG
   { ALL
   | LIKE 'строка-образец'
   | диапазонАрхЖурналов
     [LIKE 'строка-образец' [THREAD [=] целое]]
   }
```

```
диапазонАрхЖурналов для Oracle9i :=
{ { { UNTIL TIME | FROM TIME } [=] 'строка_даты'
  | { TIME BETWEEN 'строка_даты' AND
    | FROM TIME [=] 'строка_даты' UNTIL TIME [=]
    }
    'строка_даты'
  | UNTIL SCN [=] целое
  | SCN BETWEEN целое AND целое
  | FROM SCN [=] целое [UNTIL SCN [=] целое]
  }
  [THREAD [=] целое]
  | { UNTIL SEQUENCE [=] целое
  | FROM SEQUENCE [=] целое [UNTIL SEQUENCE [=] целое]
  | SEQUENCE [BETWEEN целое AND] целое
  }
  [THREAD [=] целое]
}

диапазонАрхЖурналов для Oracle8i :=
{ { UNTIL TIME | FROM TIME } [=]'строка_даты'
  | FROM TIME [=]'строка_даты' UNTIL TIME [=]'строка_даты'
  | UNTIL SCN [=]целое
  | FROM SCN [=] целое [UNTIL SCN [=] целое]
  | UNTIL LOGSEQ [=] целое [THREAD [=] целое]
  | FROM LOGSEQ [=] целое [UNTIL LOGSEQ [=] целое]
    [THREAD [=] целое]
}
```

Обеспечивает гибкость при указании архивных журнальных файлов, которые будут вовлечены в операции резервного копирования, восстановления и хранения.

Ключевые слова

ALL

Указывает, что в операции будут участвовать все журналы.

LIKE 'строка-образец'

Выбирает только журналы, соответствующие указанному образцу.

диапазонАрхЖурналов

Выбирает журналы по времени, номеру системного изменения или номеру журнала.

THREAD целое

Определяет поток журнала и применяется только в рамках компонентов Oracle Parallel Server и Real Application Clusters.

Инструкция СпецификацияВремени

```
СпецификацияВремени :=
COMPLETED
{ AFTER [=]
| BETWEEN 'строка_даты' AND
| BEFORE [=] } 'строка_даты'
```

Указывает момент или период времени, в которые была выполнена операция резервирования или копирования.

Ключевые слова

AFTER 'строка_даты'

Задает время, после которого произошло копирование или резервирование.

BETWEEN 'строка_даты' AND 'строка_даты'

Задает промежуток времени, в течение которого произошло копирование или резервирование.

BEFORE 'строка_даты'

Задает время, перед которым произошло копирование или резервирование.

Инструкция ПараметрХранения

```
{ KEEP { UNTIL TIME [=] 'строка_даты' | FOREVER } { LOGS |
    NOLOGS }
| NOKEEP
}
```

Означает, что резервирование и копирование будут рассматриваться вне рамок имеющейся политики относительно времени хранения.

Ключевые слова

UNTIL TIME 'строка_даты'

Указывает, что резервная копия должна храниться до указанной даты.

FOREVER

Указывает, что резервная копия будет храниться бесконечно. Требует использования каталога восстановления.

LOGS

Сохраняет все архивные журнальные файлы, связанные с резервированием или копированием.

NOLOGS

Указывает, что архивные журнальные файлы, связанные с резервированием или копированием, не будут храниться. Соответственно БД можно будет восстановить только на тот момент времени, в который было выполнено резервирование или копирование.

Инструкция СписокОбъектов

```
СписокОбъектов :=
{ DATAFILE спецификация_файла_данных [,спецификация_файла_данных ...]
| TABLESPACE ['']имя_табличного_пространства['']
  [, ['']имя_табличного_пространства[''] ...]
| СпецификацияАрхЖурнала
| DATABASE [SKIP TABLESPACE
  ['']имя_табличного_пространства[''] [, ['']имя_табличного_пространства['']]...]
| CONTROLFILE
}
[ DATAFILE спецификация_файла_данных [, спецификация_файла_данных]...
| TABLESPACE ['']имя_табличного_пространства['']
  [, ['']имя_табличного_пространства[''] ...]
```

```
| СпецификацияАрхЖурнала
| DATABASE [SKIP TABLESPACE
  ['имя_табличного_пространства'] [, ['имя_табличного_пространства']]...]
| CONTROLFILE
]...
```

Определяет, какие элементы БД будут включены в операции резервного копирования, восстановления и хранения.

Ключевые слова и инструкции были описаны в предыдущих разделах.

Инструкция СлужебныеПараметры

```
Служебныепараметры :=
{ TAG [=] ['имя_тега']
| СпецификацияВремени
| LIKE 'строка-образец'
| DEVICE TYPE устройство [, устройство ...]
}
```

Позволяет применять дополнительные параметры при выполнении служебных операций над файлами БД и архивными журнальными файлами. Действует только в Oracle9*i*.

Ключевые слова и инструкции были описаны в предыдущих разделах.

Инструкция УстаревшиеОперации

```
УстаревшиеОперации :=
{ REDUNDANCY [=] целое
| BACKUP WINDOW OF целое DAYS | ORPHAN }
[ REDUNDANCY [=] целое
| BACKUP WINDOW OF целое DAYS | ORPHAN }...
```

Указывает, что некоторая резервная копия может считаться устаревшей. Действует только в Oracle9*i*.

Ключевые слова

REDUNDANCY целое

Определяет минимальный уровень избыточности, после которого резервные копии считаются устаревшими.

ORPHAN

Указывает, что резервная копия устарела.

Инструкция СпецификацияЗаписи

```
СпецификацияЗаписи :=
{ { BACKUPPIECE | PROXY }
  { 'указатель_носителя' ['указатель_носителя' ...]
  | первичный_ключ [, первичный_ключ ...]
  | TAG [=] ['имя_тега']
  }
| BACKUPSET первичный_ключ [, первичный_ключ ...]
| { CONTROLFILECOPY | DATAFILECOPY }
```

```
  { { первичный_ключ [, первичный_ключ ...]
    | 'имя_файла' [, 'имя_файла' ...]
    }
  | TAG [=] ['']имя_тега['] [, ['']имя_тега['] ...]
  }
| ARCHIVELOG
  { первичный_ключ [, первичный_ключ ...]
  | 'имя_файла' [, 'имя_файла' ...]
  }
| СпецификацияАрхЖурнала
}
```

Указывает объекты, над которыми выполняют операции команды CHANGE, CROSSCHECK, DELETE и LIST. Действует только в Oracle9*i*.

Ключевые слова

BACKUPPIECE

Определяет физический элемент резервной копии, указывая имя файла, первичный ключ или имя тега.

PROXY

Определяет прокси-копию, указывая имя файла, первичный ключ или имя тега.

CONTROLFILECOPY

Определяет копию управляющего файла, указывая имя файла, первичный ключ или имя тега.

DATAFILECOPY

Определяет копию файла БД, указывая имя файла, первичный ключ или имя тега.

Инструкция ВерхняяГраница

```
ВерхняяГраница :=
{ UNTIL TIME [=]'строка_даты'
| UNTIL SCN [=] целое
| UNTIL SEQUENCE [=] целое THREAD [=] целое
}
```

Применяется в различных командах RMAN при определении верхней границы для времени, системного номера изменения или номера журнала.

Команды RMAN

В этом разделе приведен полный перечень команд RMAN в алфавитном порядке. Эти команды пользователь может вводить после приглашения на ввод команды RMAN. Выполнение команды можно прервать в любой момент, нажав комбинацию клавиш Ctrl-C.

Подраздел для каждой команды содержит ее краткое описание, формат, описание специфических ключевых слов и параметров, а также ссылки на описанные ранее общие ключевые слова, параметры и/или инструкции.

@ (Знак At)

```
@ имя_файла
```

Применяется для вызова файла сценария RMAN.

Общие ключевые слова и инструкции: *имя_файла*.

@@ (Двойной знак At)

```
@@ имя_файла
```

Образует ссылку на командный файл из другого командного файла. Предполагается, что файлы находятся в одном каталоге.

Общие ключевые слова и инструкции: *имя_файла*.

ALLOCATE CHANNEL

```
ALLOCATE [AUXILIARY] CHANNEL ['] id_канала ['']
    { DEVICE TYPE [=] устройство | NAME [=] 'имя_канала'}
    [ ПараметрыКанала [ПараметрыКанала . . .]];
```

Создает канал связи между RMAN и устройством вывода. Команду ALLOCATE CHANNEL можно выполнить только из команды RUN.

Ключевые слова

AUXILIARY Указывает, что создается канал к вспомогательной БД. Указывается в команде DUPLICATE и при восстановлении табличного пространства на момент времени.

NAME Предшествует имени конкретного устройства. Имейте в виду, что в Oracle9*i* эта конструкция поддерживается не всеми платформами.

Общие ключевые слова и инструкции: *ПараметрыКанала*, *id_канала*, *устройство*.

ALLOCATE CHANNEL FOR MAINTENANCE

```
ALLOCATE CHANNEL FOR MAINTENANCE
    { DEVICE TYPE [=] устройство | NAME [=] 'имя_канала'}
    [ ПараметрыКанала [ПараметрыКанала . . .]];
```

Создает канал связи между RMAN и устройством вывода при подготовке к выполнению команды CHANGE, CROSSCHECK или DELETE.

Ключевые слова

NAME Предшествует имени конкретного устройства. Не поддерживается в Oracle9*i*.

Общие ключевые слова и инструкции: *ПараметрыКанала*, *устройство*.

ALTER DATABASE

```
ALTER DATABASE { MOUNT | OPEN [RESETLOGS] } |
     { MOUNT | OPEN [RESETLOGS] DATABASE };
```

Аналог команды ALTER DATABASE в SQL. Команду можно применять для монтирования и открытия БД из командной строки RMAN. Команду можно включать внутрь команды RUN или применять независимо.

Ключевые слова

MOUNT Монтирует, но не открывает базу данных.

OPEN Открывает базу данных.

RESETLOGS Возвращает нумерацию оперативных журналов к исходному состоянию – она после этого начинается с 1. Если выполнить команду RMAN ALTER DATABASE, то целевая БД автоматически будет возвращена в исходное состояние внутри каталога восстановления. Задавая SQL*Plus-версию, необходимо выполнить команду RMAN RESET DATABASE, чтобы вернуть целевую БД в исходное состояние внутри каталога восстановления.

BACKUP

Синтаксис Oracle9*i*:

```
BACKUP [ FULL | INCREMENTAL LEVEL [=] целое ]
[ОперандРезервирования [ОперандРезервирования ...]]
спецификацияРезервирования [спецификацияРезервирования]...
[PLUS ARCHIVELOG
[операндСпецификацииРезервирования  [операндСпецификацииРезервирования  ...]]];

ОперандРезервирования :=
{ FORMAT [=] 'строка_формата' [,'строка_формата' ...]
| CHANNEL ['']id_канала['']
| CUMULATIVE
| MAXSETSIZE [=] целое [ K | M | G ]
| FILESPERSET [=] целое
| PARMS [=] 'параметры_канала'
| POOL [=] целое
| TAG [=] ['']имя_тега['']
| ПараметрХранения
| SKIP { OFFLINE | READONLY | INACCESSIBLE }
| NOEXCLUDE
| PROXY [ONLY]
| VALIDATE
| FORCE
| DISKRATIO [=] целое
| NOT BACKED UP [SINCE TIME [=] 'строка_даты']
| NOCHECKSUM
| CHECK LOGICAL
| COPIES [=] целое
| DEVICE TYPE устройство
}

спецификацияРезервирования :=
[(]
{ BACKUPSET
  { { ALL | СпецификацияВремени }
  | первичный_ключ [,первичный_ключ ...]
  }
| DATAFILE спецификация_файла_данных [, спецификация_файла_данных ...]
```

```
| DATAFILECOPY 'имя_файла' ['имя_файла' ...]
| DATAFILECOPY TAG [=] ['']имя_тега[''] [,
    ['']имя_тега[''] ...]
| TABLESPACE ['']имя_табличного_пространства['']
    [, ['']имя_табличного_пространства[''] ...]
| DATABASE
| СпецификацияАрхЖурнала
| CURRENT CONTROLFILE [FOR STANDBY]
| CONTROLFILECOPY 'имя_файла'
}
[операндСпецификацииРезервирования [операндСпецификацииРезервирования ]...]
[)]

операндСпецификацииРезервирования :=
{ FORMAT [=] 'строка_формата' [, 'строка_формата' ...]
| CHANNEL ['']id_канала['']
| MAXSETSIZE [=] целое [ K | M | G ]
| FILESPERSET [=] целое
| PARMS [=]'параметры_канала'
| POOL [=] целое
| TAG [=] ['']имя_тега['']
| ОпцияХранения
| SKIP { OFFLINE | READONLY | INACCESSIBLE }
| NOEXCLUDE
| FORCE
| DISKRATIO [=] целое
| NOT BACKED UP [SINCE TIME [=] 'строка_даты']
| INCLUDE CURRENT CONTROLFILE [FOR STANDBY]
| DELETE [ALL] INPUT
}
```

Синтаксис Oracle8*i*:

```
BACKUP [ FULL | INCREMENTAL LEVEL [=] целое ]
[ОперандРезервирования [ОперандРезервирования]...]
спецификацияРезервирования [спецификацияРезервирования];

ОперандРезервирования :=
{ FORMAT [=] 'строка_формата' [, 'строка_формата' ...]
| CHANNEL ['']id_канала['']
| CUMULATIVE
| FILESPERSET [=] целое
| PARMS [=] 'параметры_канала'
| POOL [=] целое
| TAG [=] ['']имя_тега['']
| ОпцияХранения
| SKIP { OFFLINE | READONLY | INACCESSIBLE }
| PROXY [ONLY]
| DISKRATIO [=] целое
| NOCHECKSUM
| CHECK LOGICAL
| SETSIZE [=] целое
}

спецификацияРезервирования :=
[(]
{ DATAFILE спецификация_файла_данных [, спецификация_файла_данных ...]
| DATAFILECOPY 'имя_файла' [, 'имя_файла' ...]
```

```
| DATAFILECOPY TAG [=] ['']имя_тега['] [,\
  ['']имя_тега['] ...]
| TABLESPACE ['']имя_табличного_пространства['] [,\
  ['']имя_табличного_пространства['] ...]
| DATABASE
| СпецификацияАрхЖурнала
| CURRENT CONTROLFILE
| CONTROLFILECOPY 'имя_файла'
}
[операндСпецификацииРезервирования [операндСпецификацииРезервирования ]...]
[)]

операндСпецификацииРезервирования :=
{ FORMAT [=] 'строка_формата' [,'строка_формата' ...]
| CHANNEL ['']id_канала['']
| FILESPERSET [=] целое
| PARMS [=] 'параметры_канала'
| POOL [=] целое
| TAG [=] ['']имя_тега['']
| SKIP { OFFLINE | READONLY | INACCESSIBLE }
| DISKRATIO [=] целое
| INCLUDE CURRENT CONTROLFILE
| DELETE INPUT
| SETSIZE [=] целое
}
```

Создает резервную копию БД, физической составляющей (например, файла БД), архивного журнального файла или логической составляющей (например, табличного пространства). В Oracle8*i* команду можно выполнять только из команды RUN.

Ключевые слова

INCREMENTAL

Следующее за данным ключевым словом целое число определяет уровень инкрементного резервного копирования. Возможны уровни от 0 (полное резервное копирование) до 4. Процесс инкрементного резервного копирования сохраняет все изменения, выполненные после последнего резервного копирования того же уровня. Так, при выполнении инкрементного резервного копирования 2-го уровня будут скопированы все изменения, внесенные после последнего резервного копирования уровня 2, 1 или 0; в резервную копию войдут все изменения, сохраненные при резервном копировании уровня 3, если таковое выполнялось после данного резервного копирования 2-го уровня.

CURRENT CONTROLFILE [*FOR STANDBY*]

Указывает, что следует включить в резервную копию текущий управляющий файл. Если указать параметр FOR STANDBY, то будет создан управляющий файл, который можно использовать для создания резервной БД.

CUMULATIVE

Означает, что RMAN будет резервировать только те блоки, которые изменились с момента последнего резервного копирования предыдущего уровня.

MAXSETSIZE целое

Ограничивает размер набора резервных копий. Размер можно указывать в байтах, килобайтах (K), мегабайтах (M) или гигабайтах (G).

FILESPERSET целое

Ограничивает количество файлов в наборе резервных копий.

PARMS 'парам_канала'

Посылает уровню операционной системы специфическую для ОС информацию при каждом создании элемента резервирования.

POOL

Задает пул носителей, где будут храниться резервные копии. Данный параметр применяется в сочетании с программным обеспечением MML.

SKIP

Пропускает файлы данных, которые доступны только для чтения, находятся в автономном режиме или недоступны.

NOEXCLUDE

Изменяет настройки по умолчанию, заданные параметром CONFIGURE EXCLUDE.

PROXY

Позволяет программному обеспечению MML принимать решения о том, куда и как передавать данные.

VALIDATE

Проверяет файлы на физические и логические ошибки.

FORCE

Позволяет RMAN игнорировать оптимизационные настройки CONFIGURE.

DISKRATIO целое

Распределяет нагрузку резервного копирования между дисками.

NOT BACKED UP [*SINCE TIME 'строка_даты'*]

Резервирует те файлы, которые не были зарезервированы вообще или после указанной даты и времени.

NO CHECKSUM

Отключает вычисление контрольных сумм блоков.

COPIES целое

Задает количество резервных копий, которые должны быть созданы. Значение по умолчанию равно 1.

DELETE

Удаляет в случае успешного завершения резервного копирования исходные файлы резервных копий, копий файлов данных или архивного журнального файла.

Общие ключевые слова и инструкции: *СпецификацияАрхЖурнала*, *id_канала*, *спецификация_файла_данных*, *устройство*, *имя_файла*, *строка_формата*, *параметрХранения*, *параметры_канала*, *первичный_ключ*, *имя_табличного_пространства*, *имя_тега*.

BLOCKRECOVER

```
BLOCKRECOVER
[DEVICE TYPE устройство [, устройство]...]
спецификацияБлока [спецификацияБлока]... [параметрБлока [параметрБлока]...];
```

```
спецификацияБлока :=
{ DATAFILE спецификация_файла_данных BLOCK целое [, целое ...]
| TABLESPACE имя_табличного_пространства DBA целое [, целое ...]
| CORRUPTION LIST
}
параметрБлока :=
{ FROM { BACKUPSET | DATAFILECOPY }
| FROM TAG [=] ['']имя_тега['']
| RESTORE ВерхняяГраница
}
```

Обеспечивает выборочное восстановление указанного перечня поврежденных блоков, а не целого файла данных. Данный вариант восстановления доступен только в Oracle9*i*.

Ключевые слова

BLOCK целое

Указывает блоки, требующие восстановления.

DBA

Указывает адрес блока данных, подлежащего восстановлению.

CORRUPTION LIST

Восстанавливает блоки, перечисленные в представлениях словаря данных V$COPY_CORRUPTION и V$BACKUP_CORRUPTION.

FROM

Указывает, что восстановление следует проводить на основе набора резервных копий или же копии файла данных.

FROM TAG имя_тега

Определяет набор резервных копий, который необходимо восстановить и который будет использован для восстановления на уровне блоков.

RESTORE верхняяГраница

Указывает, что следует восстановить наборы резервных копий, созданные до заданного момента времени.

Общие ключевые слова и инструкции: *спецификация_файла_данных*, *устройство*, *имя_табличного_пространства*, *имя_тега*, *верхняяГраница*.

CATALOG

```
CATALOG { CONTROLFILECOPY | DATAFILECOPY | ARCHIVELOG }
   'имя_файла' [, 'имя_файла' ...]
[TAG [=] ['']имя_тега['] | LEVEL [=] целое ]
[ TAG [=] ['']имя_тега['] | LEVEL [=] целое ]...;
```

Добавляет в каталог восстановления информацию, относящуюся к пользовательским командам COPY. Данную команду следует применять, если копия была сделана прежде, чем было установлено соединение с каталогом восстановления.

Ключевые слова

CONTROLFILECOPY 'имя_файла'

Определяет имя скопированного управляющего файла, который следует добавить в каталог восстановления.

DATEFILECOPY 'имя_файла'

Определяет имя скопированного файла данных, который следует добавить в каталог восстановления.

ARCHIVELOG 'имя_файла'

Определяет имя архивного журнального файла, который следует добавить в каталог восстановления.

TAG имя_тега

Задает имя тега, которое должно быть присвоено файлу в каталоге восстановления.

LEVEL целое

Определяет, что копия файла должна быть записана в каталог восстановления с указанным уровнем инкрементного резервного копирования.

Общие ключевые слова и инструкции: *имя_файла*, *имя_тега*.

CHANGE

Синтаксис Oracle9*i*:

```
CHANGE
   { { BACKUP | COPY } [OF СписокОбъектов]
     [СлужебныеПараметры [СлужебныеПараметры]...}
   | СпецификацияЗаписи
     [DEVICE TYPE устройство [, устройство]...]
   }
   { AVAILABLE | UNAVAILABLE | UNCATALOG | ПараметрХранения }
   [DEVICE TYPE устройство [, устройство]...];
```

Синтаксис Oracle8*i*:

```
CHANGE
   { ARCHIVELOG
      { первичный_ключ [, первичный_ключ...]
      | 'имя_файла' [,'имя_файла' ...] }
   | СпецификацияАрхЖурнала
   | BACKUPPIECE { 'указатель_носителя' [,'указатель_носителя' ...]
                 | первичный_ключ [, первичный_ключ ...]
                 | TAG [=] ['']имя_тега[''] }
   | BACKUPSET первичный_ключ [, первичный_ключ ...]
   | { CONTROLFILECOPY | DATAFILECOPY }
     { первичный_ключ [, первичный_ключ ...]
     | 'имя_файла' [,'имя_файла' ...]
     | TAG [=] ['']имя_тега[''] [, ['']имя_тега ['']  ... ] }
   | PROXY { 'указатель_носителя' [,'указатель_носителя' ...]
           | первичный_ключ [, первичный_ключ ...]
           | TAG [=] ['']имя_тега[''] }
   }
   { DELETE | AVAILABLE | UNAVAILABLE
   | UNCATAOG | CROSSCHECK };
```

Позволяет обрабатывать наборы резервных копий или сопоставленных им элементов резервирования. Поддерживаются такие операции, как изменение статуса хранимой записи для наборов резервных копий и элементов резервирования или удаление элементов резервирования с диска или магнитной ленты; в последнем случае будут также удалены соответствующие записи из управляющих файлов целевой БД и из необязательного каталога восстановления.

Ключевые слова

AVAILABLE

Изменяет статус резервных копий или копий на AVAILABLE.

UNAVAILABLE

Изменяет статус резервных копий или копий на UNAVAILABLE. Помеченные как UNAVAILABLE резервные копии не будут использованы RMAN. Поэтому последующие операции восстановления будут искать предыдущую доступную резервную копию.

UNCATALOG

Удаляет из каталога записи, сопоставленные копиям файлов данных и архивным журнальным файлам, и помечает такую информацию признаком DELETED в управляющем файле целевой БД.

Общие ключевые слова и инструкции: *СпецификацияАрхЖурнала*, *имя_файла*, *списокОбъектов*, *указатель_носителя*, *служебныеПараметры*, *первичный_ключ*, *спецификацияЗаписи*, *имя_тега*.

CONFIGURE

```
CONFIGURE
{ конфигурацияУстройства
| конфигурацияРезервирования
| { AUXNAME FOR DATAFILE спецификация_файла_данных
  | SNAPSHOT CONTROLFILE NAME
  }
  { TO 'имя_файла' | CLEAR }
| конфигурацияУпрФайла
};

конфигурацияУстройства :=
{ DEFAULT DEVICE TYPE { TO устройство | CLEAR }
| DEVICE TYPE устройство
  { PARALLELISM целое | CLEAR }
| [AUXILIARY] CHANNEL [целое]
  DEVICE TYPE устройство { ПараметрыКанала | CLEAR }
}

конфигурацияРезервирования :=
{ RETENTION POLICY { TO { BACKUP WINDOW OF целое DAYS
                        | REDUNDANCY [=] целое
                        | NONE
                        }
                   | CLEAR
                   }
| MAXSETSIZE { TO { целое [ K | M | G ]
                  | UNLIMITED
                  }
               | CLEAR
             }
| { ARCHIVELOG | DATAFILE }
  BACKUP COPIES FOR DEVICE TYPE устройство
  { TO целое | CLEAR }
| BACKUP OPTIMIZATION { ON | OFF | CLEAR }
```

```
| EXCLUDE FOR TABLESPACE имя_табличного_пространства [CLEAR]
}

конфигурацияУпрФайла :=
CONTROLFILE AUTOBACKUP
{ ON
| OFF
| CLEAR
| FORMAT FOR DEVICE TYPE
   устройство { TO 'строка_формата' | CLEAR }
}
```

Задает параметры конфигурации, значения которых сохраняются в рамках сеанса RMAN. Данная команда доступна только в Oracle9*i*.

Ключевые слова

SNAPSHOT CONTROLFILE NAME

Определяет имя файла и путь для моментальной копии управляющего файла. Значение по умолчанию зависит от платформы. В Unix это *$ORACLE_HOME/dbs*.

DEFAULT DEVICE TYPE

Определяет тип носителя по умолчанию (например, диск или магнитная лента).

PARALLELISM целое

Указывает количество автоматических каналов для устройства. Значение по умолчанию равно 1.

RETENTION POLICY

Определяет политику хранения, в рамках которой RMAN помечает наборы резервных копий как устаревшие, чтобы их можно было впоследствии удалить вручную.

RECOVERY WINDOW OF

Определяет размер окна резервирования как разность SYSDATE и указанного количества дней. Резервные копии, сделанные до рассчитанной даты, помечаются как устаревшие.

REDUNDANCY

Определяет избыточное количество резервных копий или копий, которые RMAN не должен рассматривать как устаревшие. Значение по умолчанию равно 1.

MAXSETSIZE целое

Определяет максимальный размер элемента резервирования для каждого канала в байтах, килобайтах (K), мегабайтах (M) или гигабайтах (G). По умолчанию задается значение UNLIMITED.

CONTROLFILE AUTOBACKUP

Указывает, разрешено ли автоматическое резервирование управляющего файла. По умолчанию задается значение OFF.

FORMAT FOR DEVICE TYPE

Определяет имя и местоположение управляющего файла. По умолчанию устанавливается значение %F.

CLEAR

Переустанавливает значения по умолчанию.

Общие ключевые слова и инструкции: *ПараметрыКанала*, *спецификация_файла_данных*, *устройство*, *имя_файла*, *строка_формата*, *имя_табличного_пространства*.

CONNECT

```
{ CONNECT TARGET | CATALOG | AUXILIARY [спецификация_строки_соединения] [;]
| { CONNECT CATALOG | CONNECT AUXILIARY } спецификация_строки_соединения [;]
}
```

Устанавливает из RMAN соединение с определенной базой данных, каталогом восстановления или вспомогательной БД.

Ключевые слова

TARGET

Определяет соединение с целевой БД.

CATALOG

Определяет соединение с каталогом восстановления.

AUXILIARY

Определяет соединение со вспомогательной БД.

Общие ключевые слова и инструкции: *спецификация_строки_соединения*.

COPY

```
COPY [параметрКопирования [параметрКопирования]...]
{ входнойФайл TO
  { AUXNAME | 'имя_файла' [параметрКопирования [параметрКопирования]...] }
  [,входнойФайл TO
   { AUXNAME | 'имя_файла' [параметрКопирования [опцияКопирования]...] }
  ]...
| (входнойФайл TO
   { AUXNAME | 'имя_файла' [параметрКопирования [параметрКопирования]...] }
  )
  [(входнойФайл TO
    { AUXNAME | 'имя_файла' [параметрКопирования [параметрКопирования]...] }
   )
  ]...
};
```

Синтаксис Oracle9*i*:

```
параметрКопирования :=
{ TAG [=] ['']имя_тега['']
| LEVEL [=] целое
| NOCHECKSUM
| CHECK LOGICAL
| ПараметрХранения
}

входнойФайл :=
{ DATAFILE спецификация_файла_данных
| DATAFILECOPY
  { 'имя_файла' | TAG [=] ['']имя_тега[''] }
| ARCHIVELOG 'имя_файла'
```

```
| CURRENT CONTROLFILE [FOR STANDBY]
| CONTROLFILECOPY
  { 'имя_файла' | TAG [=] ['']имя_тега['] }
}
```

Синтаксис Oracle8*i*:

```
параметрКопирования :=
{ TAG [=] ['']имя_тега['']
| LEVEL [=] целое
| NOCHECKSUM
| CHECK LOGICAL
}

входнойФайл :=
{ DATAFILE спецификация_файла_данных
| DATAFILECOPY
  { 'имя_файла' | TAG [=] ['']имя_тега['] }
| ARCHIVELOG 'имя_файла'
| CURRENT CONTROLFILE
| CONTROLFILECOPY
  { 'имя_файла' | TAG [=] ['']имя_тега['] }
}
```

Создает точные копии для файлов БД. В Oracle8*i* эту команду можно выполнить только из команды RUN.

Ключевые слова

AUXNAME

Указывает, что команда COPY создает файл с именем, ранее определенным в команде CONFIGURE.

NO CHECKSUM

Отключает вычисление контрольных сумм для блоков.

Общие ключевые слова и инструкции: *спецификация_файла_данных*, *имя_файла*, *ПараметрХранения*, *имя_тега*.

CREATE CATALOG

```
CREATE CATALOG [TABLESPACE ['']имя_табличного_пространства[']][;]
```

Создает запрошенную схему для каталога восстановления. Прежде чем выполнять данную команду, необходимо установить соединение с каталогом восстановления.

Общие ключевые слова и инструкции: *имя_табличного_пространства*.

CREATE SCRIPT

```
CREATE SCRIPT ['']имя_сценария[']
{
   { командыРезервирования
    | командыВосстановления
    | командыСлужебные
    | командыДругие
   }
   [ командыРезервирования
```

```
    | командыВосстановления
    | командыСлужебные
    | командыДругие
    ...
    ]
}

командыРезервирования :=
{ BACKUP
| COPY
}

командыВосстановления :=
{ REPLICATE
| RESTORE
| RECOVER
| BLOCKRECOVER
| DUPLICATE
| SWITCH
}

командыСлужебные :=
{ CATALOG
| CHANGE
| CONFIGURE
| CROSSCHECK
| DELETE
| VALIDATE
| REPORT
| DELETE
| SHOW
}

командыДругие :=
{ ALLOCATE
| ALTERDATABASE
| BEGINLINE
| DEBUG
| EXECUTESCRIPT
| HOST
| RELEASE
| RESYNC
| SEND
| SET
| SHUTDOWN
| SQL
| STARTUP
}
```

Создает сценарий, который затем сохраняется в каталоге восстановления.

CROSSCHECK

Синтаксис Oracle9*i*:

```
CROSSCHECK
   { { BACKUP | COPY }
      [OF СписокОбъектов] [СлужебныеПараметры [СлужебныеПараметры]...]
```

```
    | СпецификацияЗаписи
      [DEVICE TYPE устройство [, устройство]...]
    };
```

Синтаксис Oracle8*i*:

```
CROSSCHECK BACKUP
   [ OF СписокОбъектов ]
   [ параметрыПроверки [ параметрыПроверки...]];

параметрыПроверки :=
{ TAG [=] ['']имя_тега['']
| СпецификацияВремени }
```

Определяет, существует ли еще набор резервных копий и соответствующие элементы на носителе. Если элемент резервирования расположен там, где это указано в управляющем файле целевой БД или необязательном каталоге восстановления, он получает статус AVAILABLE. Если элемент отсутствует в указанном месте, то он помечается как EXPIRED.

Общие ключевые слова и инструкции: *СпецификацияВремени*, *СписокОбъектов*, *СлужебныеПараметры*, *имя_тега*.

DELETE

Синтаксис Oracle9*i*:

```
DELETE [NOPROMPT]
   { [EXPIRED]
     { { BACKUP | COPY }
        [OF СписокОбъектов] [СлужебныеПараметры [СлужебныеПараметры]...]
     | СпецификацияЗаписи
       [DEVICE TYPE устройство [, устройство]...]
     }
   | OBSOLETE [УстаревшиеОперации]
     [DEVICE TYPE [устройство [, устройство]...]
   };
```

Синтаксис Oracle8*i*:

```
DELETE EXPIRED BACKUP
   [ OF СписокОбъектов ]
   [ параметрыУдаления [ параметрыУдаления...]];

параметрыУдаления :=
   { TAG [=] ['']имя_тега['']
   | СпецификацияВремени }
```

В Oracle9*i* удаляет физические файлы, которым сопоставлены наборы резервных копий и копии файлов данных, обновляет их статус в управляющем файле и удаляет их информацию из необязательного каталога восстановления (если он используется).

В Oracle8*i* и Oracle9*i* резервные копии помечаются как EXPIRED, если они не найдены в указанном месте. Удаление резервной копии, помеченной как EXPIRED, приводит к удалению информации о ней из управляющего файла и из необязательного каталога восстановления (если он используется).

Ключевые слова

NOPROMPT

Удаляет указанные файлы без запроса подтверждения операции.

EXPIRED Удаляет файлы, которые были помечены как EXPIRED в каталоге.

OBSOLETE Удаляет файлы, которые больше не нужны для восстановления.

Общие ключевые слова и инструкции: *СпецификацияВремени*, *СписокОбъектов*, *УстаревшиеОперации*, *имя_тега*.

DELETE SCRIPT

```
DELETE SCRIPT ['']имя_сценария['];
```

Удаляет сценарий из каталога восстановления.

DROP CATALOG

```
DROP CATALOG;
```

Удаляет все объекты, связанные с каталогом восстановления.

DUPLICATE

Синтаксис Oracle9*i*:

```
DUPLICATE TARGET DATABASE
   { TO ['']имя_бд['']
     [ [параметрыДублирования] [параметрыДублирования ...] ]
   | FOR STANDBY [ [параметрыДублированияРезервнойБД]
      [параметрыДублирования ...] ]
   };

параметрыДублирования :=
{ LOGFILE спецификацияЖурнала [, спецификацияЖурнала]...
| NOFILENAMECHECK
| SKIP READONLY
| DEVICE TYPE устройство [, устройство ...]
| PFILE [=] ['']имя_файла['']
}

спецификацияЖурнала :=
{ 'имя_файла' [SIZE целое [ K | M ] [REUSE]]
| GROUP целое ('имя_файла' [, 'имя_файла' ...])
   [SIZE целое [ K | M ] [REUSE]] }

параметрыДублированияРезервнойБД :=
{ DORECOVER | NOFILENAMECHECK }
```

Синтаксис Oracle8*i*:

```
DUPLICATE TARGET DATABASE TO ['']имя_бд['']
   [ LOGFILE спецификацияЖурнала [, спецификацияЖурнала ...]]
   [ NOFILENAMECHECK ]
   [ SKIP READONLY ];

спецификацияЖурнала :=
   { 'имя_файла' [SIZE целое [ K | M ] [REUSE]]
   | GROUP целое ('имя_файла' [, 'имя_файла' ...])
      [SIZE целое [ K | M ] [REUSE]] }
```

Создает дублирующую (резервную) базу данных из резервной копии целевой БД. В Oracle8*i* данную команду можно выполнить только из команды RUN.

Ключевые слова

NOFILENAMECHECK

Приводит к тому, что RMAN не проверяет, совпадают ли имена файлов целевой БД с именами в дублирующей БД. Данный параметр следует применять, только если дублирующая БД находится на другом хосте (в этом случае допускается давать одинаковые имена каталогам и файлам).

PFILE

Указывает местоположение файла параметров для дублирующей БД. RMAN использует файл параметров, расположенный в указанном месте, для запуска дублирующей БД (файл необходимо предварительно создать). Параметр необходимо задавать, если местоположение файла параметров не совпадает со значением по умолчанию (*$ORACLE_HOME/dbs*).

DORECOVER

Означает, что после создания резервной БД RMAN выполнит ее восстановление. При необходимости будут применяться архивные журнальные файлы.

Общие ключевые слова и инструкции: *устройство*, *имя_файла*.

EXECUTE SCRIPT

```
EXECUTE SCRIPT ['']имя_сценария['];
```

Исполняет сценарий, хранящийся в каталоге восстановления.

EXIT

```
EXIT[;]
```

Выход из RMAN.

HOST

```
HOST [{' | "}команда{' | "}];
```

Исполняет команду операционной системы. По завершении команды управление возвращается среде RMAN. Если выдать такую команду без указания команды ОС, то будет открыт сеанс операционной системы.

LIST

Синтаксис Oracle9*i*:

```
LIST
   { INCARNATION [OF DATABASE [['']имя_бд['']]]
   | [EXPIRED]
     { списокСпецификацийОбъектов
       [ СлужебныеПараметры | RECOVERABLE [ВерхняяГраница] ]
       [ СлужебныеПараметры | RECOVERABLE [ВерхняяГраница] ]...
     | СпецификацияЗаписи
     }
   };

списокСпецификацийОбъектов :=
```

```
{ BACKUP [OF СписокОбъектов] [списокПараметровРезервирования]
| COPY [OF СписокОбъектов]
}

списокПараметровРезервирования :=
[ [BY BACKUP] [VERBOSE]
| SUMMARY
| BY { BACKUP SUMMARY | FILE }
]
```

Синтаксис Oracle8*i*:

```
LIST
   { INCARNATION [OF DATABASE [ ['']имя_бд[''] ] ]
   | { BACKUP | COPY } [OF СписокОбъектов]
     [ списокПараметров [списокПараметров ...] ]
   };

списокПараметров :=
{ { TAG [=] ['']имя_тега['']
  | СпецификацияВремени }
| RECOVERABLE [ ВерхняяГраница ]
| DEVICE TYPE устройство [, устройство ...]
| LIKE 'строка-образец' }
```

Формирует список резервных копий или точных копий.

Ключевые слова

EXPIRED

Перечисляет наборы резервных копий, которые были помечены как EXPIRED в управляющем файле целевой БД в каталоге восстановления.

BY BACKUP

Приводит перечень наборов резервных копий и их содержимого.

VERBOSE

Приводит подробную информацию о наборе резервных копий.

SUMMARY

Выводит итоговую строку для каждого набора резервных копий или файла.

Общие ключевые слова и инструкции: *СпецификацияВремени*, *устройство*, *СписокОбъектов*, *СлужебныеПараметры*, *СпецификацияЗаписи*, *имя_тега*, *ВерхняяГраница*.

PRINT SCRIPT

```
PRINT SCRIPT ['']имя_сценария[''];
```

Выводит сценарий, хранящийся в каталоге восстановления.

QUIT

```
QUIT[;]
```

Выход из RMAN.

RECOVER

Синтаксис Oracle9*i*:

```
RECOVER [DEVICE TYPE устройство
    [, устройство]...]
восстанавливаемыйОбъект [списокПараметровВосстановления];

восстанавливаемыйОбъект :=
{ DATABASE
  [ ВерхняяГраница
  | [ВерхняяГраница] SKIP [FOREVER] TABLESPACE
    ['']имя_табличного_пространства['] [, ['']имя_табличного_пространства['] ...]
  ]
| TABLESPACE ['']имя_табличного_пространства['']
  [, ['']имя_табличного_пространства['] ...]
| DATAFILE спецификация_файла_данных [, спецификация_файла_данных ...]
}

списокПараметровВосстановления :=
{ DELETE ARCHIVELOG
| CHECK READONLY
| NOREDO
| CHECK LOGICAL
| { FROM TAG | ARCHIVELOG TAG } [=] ['']имя_тега['']
}
[, { DELETE ARCHIVELOG
   | CHECK READONLY
   | NOREDO
   | CHECK LOGICAL
   | { FROM TAG | ARCHIVELOG TAG } [=] ['']имя_тега['']
   }
]...
```

Синтаксис Oracle8*i*:

```
RECOVER восстанавливаемыйОбъект [списокПараметровВосстановления];

списокПараметровВосстановления :=
{ DELETE ARCHIVELOG
| CHECK READONLY
| NOREDO
| CHECK LOGICAL
}
[, { DELETE ARCHIVELOG
   | CHECK READONLY
   | NOREDO
   | CHECK LOGICAL
   }
] ...
```

Восстанавливает БД или одну из ее физических составляющих. Команда восстанавливает файлы данных, основываясь на инкрементных резервных копиях, если такие имеются, или же архивные журнальные файлы. В Oracle8*i* команда может быть выполнена только из команды RUN.

Ключевые слова

DELETE ARCHIVELOG

Удаляет ставшие ненужными архивные журнальные файлы.

CHECK READONLY

Исключает из процесса восстановления табличные пространства, доступные только для чтения, если к ним относятся текущие файлы.

NOREDO

Вынуждает использовать при восстановлении только инкрементные резервные копии. Архивные журнальные файлы не применяются. Данный параметр действует при восстановлении БД, работающей в режиме NOARCHIVELOG.

CHECK LOGICAL

Проверяет БД на предмет логического повреждения и записывает результаты в сигнальный файл ALERT и серверные файлы трассировки сеанса.

Общие ключевые слова и инструкции: *спецификация_файла_данных*, *устройство*, *имя_табличного_пространства*, *имя_тега*, *ВерхняяГраница*.

REGISTER

```
REGISTER DATABASE;
```

Регистрирует целевую БД в каталоге восстановления. Прежде чем выдать такую команду, необходимо подключиться к каталогу и к целевой БД, которую следует зарегистрировать.

RELEASE CHANNEL

```
RELEASE CHANNEL [']id_канала['];
```

Освобождает канал, который был выделен для ввода/вывода. После выделения каналы остаются открытыми до завершения задания или до их явного освобождения. Данную команду следует применять для закрытия канала в случае, если RMAN еще поддерживает соединение с целевой БД. В Oracle8*i* команда может быть выполнена только из команды RUN.

Если выдать команду без указания идентификатора канала, то будет закрыт служебный канал, созданный командой ALLOCATE CHANNEL FOR MAINTENANCE.

Общие ключевые слова и инструкции: *id_канала*.

REPLACE SCRIPT

```
REPLACE SCRIPT [']имя_сценария['] 
{
 { командыРезервирования
 | командыВосстановления
 | командыСлужебные
 | командыДругие
 }
 [ командыРезервирования
 | командыВосстановления
 | командыСлужебные
 | командыДругие
 ]...
}
```

Заменяет существующий сценарий в каталоге восстановления. Если сценарий с указанным именем не существует, то он будет создан.

Состав команд, включенных в различные группы, был приведен в разделе для команды CREATE SCRIPT.

REPLICATE

```
REPLICATE CONTROLFILE FROM 'имя_файла';
```

Реплицирует управляющие файлы в те места, которые указаны в параметре инициализации CONTROL_FILES для целевой БД. Команда может быть выполнена только из команды RUN.

Общие ключевые слова и инструкции: *имя_файла*.

REPORT

Синтаксис Oracle9*i*:

```
REPORT
{ { NEED BACKUP [ { INCREMENTAL | DAYS } [=] целое
                  | REDUNDANCY [=] целое
                  | BACKUP WINDOW OF целое DAYS
                ] ]
  | UNRECOVERABLE
  }
  объектОтчета
| SCHEMA [атИнструкция]
| OBSOLETE [УстаревшиеОперации]
}
[ DEVICE TYPE устройство [, устройство ...]];

объектОтчета :=
[ DATAFILE спецификация_файла_данных [, спецификация_файла_данных]...
| TABLESPACE ['']имя_табличного_пространства['']
  [, ['']имя_табличного_пространства[''] ...]
| DATABASE [SKIP TABLESPACE
   ['']имя_табличного_пространства[''] [, ['']имя_табличного_пространства['']]...]
]

атИнструкция :=
{ AT TIME [=] 'строка_даты'
| AT SCN [=] целое
| AT SEQUENCE [=] целое THREAD [=] целое
}
```

Синтаксис Oracle8*i*:

```
REPORT
{ { NEED BACKUP [ { INCREMENTAL | DAYS } [=] целое
                  | REDUNDANCY [=] целое ]
  | UNRECOVERABLE
  }
  объектОтчета
| SCHEMA [атИнструкция]
| OBSOLETE [УстаревшиеОперации]
}

объектОтчета :=
[ DATAFILE спецификация_файла_данных [, спецификация_файла_данных]...
| TABLESPACE ['']имя_табличного_пространства['']
  [, ['']имя_табличного_пространства[''] ...]
```

```
| DATABASE [SKIP TABLESPACE
   ['']имя_табличного_пространства['] [, ['']имя_табличного_пространства['']]...]
]

atИнструкция :=
{ AT TIME [=] 'строка_даты'
| AT SCN [=] целое
| AT LOGSEQ [=] целое THREAD [=] целое
}
```

Выводит подробный отчет о ходе резервного копирования БД. Подобные отчеты могут предоставлять данные о резервных копиях, необходимых для восстановления, которые должны применяться для восстановления на основе определенного количества инкрементных резервных копирований или которые подлежат удалению.

Ключевые слова

NEED BACKUP

Выводит отчет о файлах данных, которым необходимо резервное копирование на основе показателя избыточности, окна восстановления или же инкрементного резервного копирования.

UNRECOVERABLE

Выводит отчет обо всех невосстанавливаемых файлах данных.

SCHEMA [atИнструкция]

Выводит отчет о табличных пространствах и файлах данных на определенный момент времени.

OBSOLETE [УстаревшиеОперации]

Выводит список копий, которые более не требуются и могут быть удалены.

Общие ключевые слова и инструкции: *спецификация_файла_данных*, *устройство*, *УстаревшиеОперации*, *имя_табличного_пространства*.

RESET DATABASE

```
RESET DATABASE [TO INCARNATION ключ_инкарнации];
```

Создает новую инкарнацию целевой БД или возвращает целевую БД к предыдущей инкарнации внутри каталога восстановления. Эта команда необходима только в случае, если при открытии целевой БД в команду SQL ALTER DATABASE был включен параметр RESET_LOGS. Если для сброса журналов использовалась команда RMAN ALTER DATABASE, то операция для целевой БД будет произведена автоматически.

Ключевое слово

INCARNATION ключ_инкарнации

Указывает, что более ранняя инкарнация БД должна стать текущей.

RESTORE

Синтаксис Oracle9*i*:

```
RESTORE
[(] восстанавливаемыйОбъект [(операндСпецификацииВосстановления
                              [операндСпецификацииВосстановления]...]) [)]
[(] восстанавливаемыйОбъект [(операндСпецификацииВосстановления
```

```
                              [операндСпецификацииВосстановления]...]) [)]...
[ CHANNEL ['']id_канала['']
| PARMS [=] 'параметры_канала'
| FROM { BACKUPET | DATAFILECOPY }
| ВерхняяГраница
| FROM TAG [=] ['']имя_тега['']
| VALIDATE
| CHECK LOGICAL
| CHECK READONLY
| DEVICE TYPE устройство [, устройство ...]
| FORCE
]
[ CHANNEL ['']id_канала['']
| PARMS [=] 'параметры_канала'
| FROM { BACKUPET | DATAFILECOPY }
| ВерхняяГраница
| FROM TAG [=] ['']имя_тега['']
| VALIDATE
| CHECK LOGICAL
| CHECK READONLY
| DEVICE TYPE устройство [, устройство ...]
| FORCE
]...;

восстанавливаемыйОбъект :=
{ CONTROLFILE [TO 'имя_файла']
| DATABASE
  [SKIP [FOREVER] TABLESPACE
   ['']имя_табличного_пространства[''] [, ['']имя_табличного_пространства[''] ...]
  ]
| DATAFILE спецификация_файла_данных [, спецификация_файла_данных ...]
| TABLESPACE ['']имя_табличного_пространства['']
   [, ['']имя_табличного_пространства['']]...
| СпецификацияАрхЖурнала
}

операндСпецификацииВосстановления :=
{ CHANNEL ['']id_канала['']
| FROM TAG [=] ['']имя_тега['']
| PARMS [=] 'параметры_канала'
| FROM
  { AUTOBACKUP
    [{ MAXSEQ | MAXDAYS } [=] целое]
    [{ MAXSEQ | MAXDAYS } [=] целое] ...]]
  | 'указатель_носителя'
  }
}
```

Синтаксис Oracle8*i*:

```
RESTORE
[() восстанавливаемыйОбъект [(операндСпецификацииВосстановления
                              [операндСпецификацииВосстановления]...]) [)]
[() восстанавливаемыйОбъект [(операндСпецификацииВосстановления
                              [операндСпецификацииВосстановления]...]) [)]...
[ CHANNEL ['']id_канала['']
| PARMS [=] 'параметры_канала'
| FROM { BACKUPET | DATAFILECOPY }
| ВерхняяГраница
```

```
| FROM TAG [=] ['']имя_тега['']
| VALIDATE
| CHECK LOGICAL
| CHECK READONLY
]
[ CHANNEL ['']id_канала['']
| PARMS [=] 'параметры_канала'
| FROM { BACKUPET | DATAFILECOPY }
| ВерхняяГраница
| FROM TAG [=] ['']имя_тега['']
| VALIDATE
| CHECK LOGICAL
| CHECK READONLY
]...;

операндСпецификацииВосстановления :=
{ CHANNEL ['']id_канала['']
| FROM TAG [=] ['']имя_тега['']
| PARMS [=] 'параметры_канала'
}
```

Восстанавливает целевую БД, основываясь на полной или частичной резервной копии. В Oracle8*i* команду можно выполнять только из команды RUN.

Ключевые слова

PARMS

Задает параметры для выделенного недискового устройства.

VALIDATE

Инициирует обращение к RMAN для принятия решения о том, какую резервную копию следует восстановить, и для проверки содержимого такой копии. Этот параметр не приводит к выполнению каких-либо действий по восстановлению, а предназначен для проверки наличия резервной копии.

CHECK LOGICAL

Проверяет БД на предмет логического повреждения и записывает результаты в сигнальный файл *alert.log* и серверные файлы трассировки сеанса.

CHECK READONLY

Обычно RESTORE не восстанавливает базы данных, доступные только для чтения. Данное ключевое слово означает, что команда RESTORE будет проверять файлы данных, доступные только для чтения, на существование, читаемость и содержание соответствующих контрольных точек. Если какое-либо из условий не будет выполнено, БД будет восстановлена.

FORCE

Обычно RMAN восстанавливает файлы только в том случае, если их заголовочная информация не совпадает с данными управляющего файла. Ключевое слово FORCE приводит к восстановлению всех файлов данных.

SKIP

Указывает RMAN, что следует пропустить определенное табличное пространство.

FROM AUTOBACKUP

Восстанавливает управляющий файл на основе копии автоматического резервирования. RMAN определяет текущую дату, а затем отсчитывает назад количество дней, указанное в параметре MAXDAYS.

Общие ключевые слова и инструкции: *СпецификацияАрхЖурнала*, *id_канала*, *указатель_носителя*, *параметры_каналаимя_тега*, *ВерхняяГраница*.

RESYNC

```
RESYNC CATALOG [FROM CONTROLFILECOPY 'имя_файла'];
```

Выполняет повторную синхронизацию каталога с управляющими файлами целевой БД. Так, можно применять RESYNC в случае, если каталог восстановления был отключен для выполнения служебных операций или по каким-то другим причинам и требует синхронизации с целевой БД. Большая часть операций RMAN синхронизирует каталог с целевой БД. В Oracle8*i* команду можно выполнять только из команды RUN.

Общие ключевые слова и инструкции: *имя_файла*.

RUN

```
RUN {
    RMAN_команды . . .
}
```

Выполняет последовательность команд. В Oracle8*i* большинство команд можно выполнить, только поместив их в фигурные скобки после команды RUN. Во многих случаях в Oracle9*i* это требование уже не действует, однако перечисленные ниже команды даже в этой версии должны исполняться только внутри команды RUN:

ALLOCATE CHANNEL
EXECUTE SCRIPT
REPLICATE
SWITCH

SEND

```
SEND
[ DEVICE TYPE устройство [, устройство]...]
| CHANNEL ['']id_канала[''] [, ['']id_канала[''] ...]
]
'команда' [PARMS [=] 'параметры_канала'];
```

Отправляет специфичные для производителя команды одному или нескольким каналам при использовании Media Management Layer. В Oracle8*i* команду можно выполнять только из команды RUN.

Ключевые слова

Общие ключевые слова и инструкции: *id_канала*, *устройство*, *параметры_канала*.

SET

Синтаксис Oracle9*i*:

```
SET { параметрыRman [;] | параметрыRun; }

параметрыRman :=
{ ECHO { ON | OFF }
```

```
| DBID [=] целое
| CONTROLFILE AUTOBACKUP FORMAT
  FOR DEVICE TYPE устройство TO 'строка_формата' }

параметрыRun :=
{ NEWNAME FOR DATAFILE спецификация_файла_данных TO { 'имя_файла' | NEW }
| MAXCORRUPT FOR DATAFILE спецификация_файла_данных
   [, спецификация_файла_данных ...] TO целое
| ARCHIVELOG DESTINATION TO 'направление_архивации'
| ВерхняяГраница
| BACKUP COPIES [=] целое
| COMMAND ID TO 'строка'
| AUTOLOCATE { ON | OFF }
| CONTROLFILE AUTOBACKUP
   FORMAT FOR DEVICE TYPE устройство TO
       'строка_формата'
}
```

Синтаксис Oracle8*i*:

```
SET { параметрыRman [;] | параметрыRun;}

параметрыRman :=
 {{ AUXNAME FOR DATAFILE спецификация_файла_данных TO { 'имя_файла' | NULL }
    | DBID [=] целое
    | SNAPSHOT CONTROLFILE NAME TO 'имя_файла'
    }
  | ECHO { ON | OFF }
  };

параметрыRun :=
{ { NEWNAME FOR DATAFILE спецификация_файла_данных TO 'имя_файла'
  | MAXCORRUPT FOR DATAFILE спецификация_файла_данных
      [, спецификация_файла_данных ...]
    TO целое
  | ARCHIVELOG DESTINATION TO 'направление_архивации'
  | ВерхняяГраница
  | DUPLEX [=] { ON | OFF | целое }
  | COMMAND ID TO 'строка'
  | AUTOLOCATE { ON | OFF }
  }
| LIMIT CHANNEL ['']id_канала['']
 параметрыОграничения [параметрыОграничения ...]
}

параметрыОграничения :=
{ KBYTES [=] целое
| READRATE [=] целое
| MAXOPENFILES [=] целое
}
```

Задает значения параметров для текущего сеанса RMAN.

Ключевые слова

ECHO

Определяет, будут ли команды RMAN отображаться в журнале сообщений. По умолчанию команды, вводимые в ответ на приглашение на ввод, не отображаются в журнале, а команды сценария отображаются.

DBID

Задает уникальный 32-битный идентификационный номер, создаваемый при инсталляции БД. Обычно применяется только в отсутствие соединения с целевой БД.

CONTROLFILE AUTOBACKUP FORMAT

Изменяет формат по умолчанию для имени управляющего файла автоматического резервирования.

NEWNAME FOR DATAFILE

Задает имя файла данных по умолчанию для всех последующих команд RESTORE или SWITCH существующего сеанса RMAN. В Oracle8*i* этот параметр действует только в рамках текущей команды RUN.

MAXCORRUPT FOR DATAFILE

Устанавливает ограничение для впервые выявляемых физических повреждений блоков при выполнении команды BACKUP или RESTORE. Если заданное количество превышено, то задание завершается. В Oracle8*i* этот параметр действует только в рамках текущей команды RUN.

ARCHIVELOG DESTINATION

Изменяет для всех операций RESTORE и RECOVER сеанса принятый по умолчанию файл хранения журнала, который указан в параметре инициализации LOG_ARCHIVE_DEST_1. В Oracle8*i* этот параметр действует только в рамках текущей команды RUN.

BACKUP COPIES

Задает количество копий каждого элемента резервирования при резервировании БД. Значением может быть целое число в диапазоне от 1 до 4.

COMMAND ID

Вводит указанную строку в столбец V$SESSION.CLIENT_INFO, чтобы определить, к каким каналам RMAN подключены те или иные серверные сеансы. В Oracle8*i* этот параметр действует только в рамках текущей команды RUN.

AUTOLOCATE

Указывает RMAN, что следует автоматически определять, на каких узлах базы данных Real Application Clusters имеются резервные копии для восстановления. Если параметр не задан, то RMAN будет пытаться восстановить каждый узел.

AUXNAME FOR DATAFILE

Устанавливает имя целевой БД в имя вспомогательной БД для применения в команде DUPLICATE или при восстановлении табличного пространства на момент времени.

SNAPSHOT CONTROLFILE NAME TO

Указывает имя для моментальной копии управляющего файла.

DUPLEX

В Oracle8*i* этот параметр действует только в рамках текущей команды RUN.

LIMIT CHANNEL

Задает ограничения для всех команд BACKUP и COPY, использующих данный канал. Ограничения накладываются на размер (KBYTES), максимальное количество буферов в секунду (READRATE) или максимальное количество открытых файлов для операции BACKUP. В Oracle8*i* этот параметр действует только в рамках текущей команды RUN.

Общие ключевые слова и инструкции: *id_канала*, *спецификация_файла_данных*, *устройство*, *имя_файла*, *строка_формата*, *верхняяГраница*.

SHOW

```
SHOW
{ RETENTION POLICY
| [DEFAULT] DEVICE TYPE
| [AUXILIARY] CHANNEL [FOR DEVICE TYPE устройство]
| MAXSETSIZE
| { DATAFILE | ARCHIVELOG } BACKUP COPIES
| BACKUP OPTIMIZATION
| SNAPSHOT CONTROLFILE NAME
| AUXNAME
| EXCLUDE
| CONTROLFILE AUTOBACKUP [FORMAT]
| ALL
};
```

Выводит текущие параметры конфигурации. Действует только в Oracle9*i*.

Ключевые слова

Ключевые слова был описаны ранее в разделе для команды CONFIGURE.

Общие ключевые слова и инструкции: *устройство*.

SHUTDOWN

```
SHUTDOWN [ NORMAL | ABORT | IMMEDIATE | TRANSACTIONAL ][;]
```

Останавливает целевую БД. NORMAL – это значение по умолчанию.

Ключевые слова

ABORT

Незамедлительно останавливает БД и при следующем запуске указывает на необходимость восстановления после аварийной ситуации.

IMMEDIATE

Незамедлительно останавливает БД.

TRANSACTIONAL

Позволяет всем клиентским приложениям завершить текущие транзакции; запрещает открытие новых транзакций, отключает пользователей после завершения всех транзакций. Выполнив все эти действия, останавливает БД.

SPOOL

```
SPOOL LOG { OFF | TO имя_файла } [APPEND][;]
```

Отправляет выводимые RMAN сообщения в файл. Действует только в Oracle9*i*.

Общие ключевые слова и инструкции: *имя_файла*.

SQL

```
SQL {' | "}команда{' | "};
```

Исполняет команду SQL.

STARTUP

```
STARTUP
[ FORCE
| { NOMOUNT | MOUNT }
| DBA
| PFILE [=] ['']имя_файла['']
]
[ FORCE
| { NOMOUNT | MOUNT }
| DBA
| PFILE [=] ['']имя_файла['']
]... [;]
```

Запускает БД из RMAN, подобно SQL-команде STARTUP.

Ключевые слова

FORCE

Если БД открыта, то вызывает ее останов, а затем перезапуск.

MOUNT

Запускает и монтирует экземпляр, не открывая его.

NOMOUNT

Запускает экземпляр, не монтируя и не открывая его.

DBA

Разрешает доступ только пользователям с привилегией RESTRICTED SESSION.

PFILE

Указывает конкретный файл инициализации (INIT.ORA).

Общие ключевые слова и инструкции: *имя_файла*.

SWITCH

```
SWITCH
{ DATAFILE спецификация_файла_данных
  [ TO DATAFILECOPY { 'имя_файла' | TAG [=] ['']имя_тега[''] }]
| DATAFILE ALL
};
```

Указывает, что копия файла данных становится текущим файлом данных. Команда должна запускаться из команды RUN. Данная команда аналогична SQL-команде ALTER DATABASE RENAME *имя_файла*.

Общие ключевые слова и инструкции: *имя_файла*.

UPGRADE CATALOG

```
UPGRADE CATALOG [TABLESPACE ['']имя_табличного_пространства['']] [;]
```

Обновляет схему каталога восстановления новой версией в соответствии с версией RMAN. Утилиту RMAN нельзя использовать для каталога, который был создан более старой версией. Например, нельзя применять утилиту RMAN 9.0.1 для подключения к каталогу Oracle 8.1.7.

Общие ключевые слова и инструкции: *имя_табличного_пространства*.

VALIDATE

Синтаксис Oracle9i:

```
VALIDATE BACKUPET первичный_ключ [, первичный_ключ ...]
[ CHECK LOGICAL
| DEVICE TYPE устройство [, устройство ...]
]
[ CHECK LOGICAL
| DEVICE TYPE устройство [, устройство ...]
]...;
```

Синтаксис Oracle8i:

```
VALIDATE BACKUPET первичный_ключ [, первичный_ключ ...]
[ CHECK LOGICAL ];
```

Проверяет целостность резервной копии. RMAN просматривает все соответствующие элементы резервирования, чтобы определить, можно ли при необходимости восстановить их содержимое и не были ли они физически повреждены. В Oracle8*i* команду можно выполнять только из команды RUN.

Ключевые слова

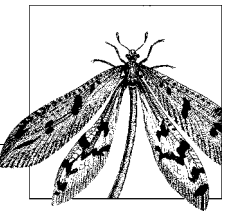
CHECK LOGICAL

Проверяет отсутствие логических повреждений, таких как повреждение записи индекса.

Общие ключевые слова и инструкции: *устройство*, *первичный_ключ*.

16

Enterprise Manager



Консоль с графическим интерфейсом пользователя Oracle Enterprise Manager (EM) предназначена для управления базами данных Oracle. Кроме того, EM предоставляет ряд сервисов, помогающих контролировать и отслеживать экземпляры баз данных, в том числе:

Discovery Wizard

Позволяет обнаружить все активные экземпляры, доступные для Enterprise Manager.

Система заданий

Позволяет задать расписание выполнения заданий технического обслуживания или запуска сценариев SQL. Система отслеживает результат работы заданий и уведомляет пользователя о статусе их завершения или о возникших ошибках. Задание может быть реализовано посредством любых команд SQL или операционной системы.

Система событий

Контролирует экземпляры БД на предмет наступления таких событий, как выход из строя или проблемы с ресурсами.

Система предупреждений

Уведомляет вас о статусе заданий или событий через консоль Enterprise Manager, электронную почту или пейджер. EM позволяет отключить систему предупреждений, чтобы не выводить ненужные сообщения о запланированных событиях.

Безопасность

Реализованы функции обеспечения безопасности для администраторов, отслеживаемых баз данных и объектов.

Системная отчетность

Выводит информацию о состоянии, использовании и конфигурации отслеживаемых систем, а также о содержимом хранилища Enterprise Manager.

В данной главе представлен обзор этой важной утилиты Oracle. По большей части интерфейс EM достаточно понятен. Поэтому мы не ставили перед собой цель привести все его элементы и подробно описать работу с ними. Вместо этого сосредоточимся на архитектуре и структуре среды EM. Более полную информацию об интерфейсе и параметрах EM можно найти в документации Oracle.

Архитектура

Начиная с версии 2.0 (которая соответствует Oracle8*i*) Enterprise Manager может работать как в трехуровневой, так и в двухуровневой архитектуре.

Трехуровневая архитектура

Компонентами трехуровневой архитектуры Enterprise Manager являются:

Консоль Enterprise Manager

Может работать из броузера, с персонального компьютера или рабочей станции Unix. Консоль EM запускается из броузера на Windows-компьютерах при обращении к странице по адресу:

http://<имя_хоста_вебсервера>.<номер_порта>

где номер порта по умолчанию равен 3339. Имейте в виду, что через броузер можно получить доступ не ко всем функциям консоли Enterprise Manager.

Oracle Management Server

Представляет собой среднее звено EM и включает в себя хранилище. Oracle Management Server обеспечивает реализацию системы заданий, системы событий и групп. В целях резервирования и выравнивания нагрузки могут применяться несколько серверов Oracle Management Server. Oracle Management Server использует в качестве хранилища базу данных Oracle.

Агенты Intelligent Agents

Работают на каждом сервере БД, управляемом Oracle Management Server. Выполняют локальные задания, осуществляют мониторинг состояния и операций экземпляров БД на сервере. Агенты Intelligent Agents работают независимо по отношению к Oracle Management Server. Например, если клиент ставит задание в расписание, то Oracle Management Server передает это задание агенту Intelligent Agent для целевой БД. Даже в случае сетевых проблем между компьютерами Oracle Management Server и целевой БД задание все равно будет запущено на целевом сервере.

Для запуска заданий на целевых узлах посредством агентов Intelligent Agents применяется механизм Preferred Credentials, предоставляющий Enterprise Manager привилегии на выполнение задач управления на удаленных узлах, при этом не приходится выдавать соответствующие привилегии высокого уровня отдельным пользователям Enterprise Manager.

В Oracle9*i* агенты Intelligent Agents автоматически перезапускаются в случае сбоя.

Двухуровневая архитектура

Если запустить Enterprise Manager в двухуровневой архитектуре, то будет получена автономно работающая на клиентской машине версия консоли Enterprise Manager с прямым доступом к серверу БД. И в двухзвенной конфигурации Enterprise Manager может выполнять массу разнообразных технологических операций, но система заданий, система событий и группы требуют применения сервера Oracle Management Server, который доступен только в трехуровневой архитектуре.

Двухуровневая архитектура не требует применения хранилища, но для некоторых приложений (таких как диспетчер изменений Change Manager, модуль автоматической настройки Expert, модуль настройки SQL – SQL Analyze, мастер настройки ин-

дексов Index Tuning Wizard и интерфейс анализа журнальных файлов LogMiner Viewer) автономное хранилище необходимо.

Запуск Enterprise Manager

Для запуска консоли Enterprise Manager введите команду

```
oemapp console
```

на компьютере, который будет выступать в качестве консоли, или же выберите ее из перечня программ в меню Programs при работе в Windows. Система запросит имя пользователя и пароль, а также задаст вопрос о том, следует ли запустить консоль EM автономно или же вместе с сервером Oracle Management.

По умолчанию для главного администратора действуют имя и пароль SYSMAN/OEM_TEMP. После запуска консоли можно создать пары *имя_пользователя/пароль* для других пользователей.

При работе в трехуровневой конфигурации необходимо также запустить сервер Oracle Management Server. Такой запуск может входить в последовательность начальной загрузки сервера среднего звена. Сервер Management Server также можно запустить, вызвав утилиту OEMCTRL командой

```
oemctrl start oms
```

на том сервере, гда работает Management Server.

Утилита OEMCTRL также предлагает пользователю несколько команд:

`oemctrl ping oms`

Проверяет, работает ли Management Server.

`oemctrl status oms`

Возвращает состояние Management Server.

`oemctrl stop oms`

Останавливает работу Management Server.

Интерфейс Enterprise Manager

Консоль Enterprise Manager предоставляет графический интерфейс для функций Enterprise Manager. Стандартный вид консоли для Oracle8*i* изображен на рис. 16.1.

Области консоли

Консоль EM содержит четыре основные области:

Панель инструментов

Расположена в левой верхней части окна. Обеспечивает доступ к общим функциям для объектов из находящейся справа области навигатора. Шесть значков представляют (сверху вниз) просмотр отчетов Enterprise Manager, обновление окна навигатора, создание нового объекта, создание нового объекта на основе существующего, удаление объекта и вывод главной страницы справочной системы. Эти кнопки доступны не всегда, а в зависимости от того, какой именно объект выбран в области навигации.

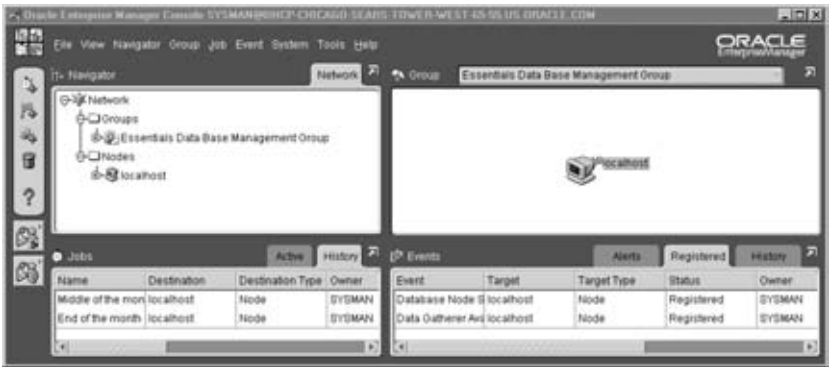


Рис. 16.1. *Консоль Enterprise Manager для Oracle8i*

Панели запуска

Расположены непосредственно под панелью инструментов. Применяются для запуска различных приложений. Каждая панель может содержать несколько приложений. Из них могут быть запущены (если установлены):

*SQL*Plus Worksheet*

Интерфейс для SQL*Plus с окном ввода в верхней части экрана и окном результатов в нижней части. Одновременно может быть открыто несколько копий SQL*Plus Worksheet.

SQL Scratchpad

Этот инструмент появился в Oracle9*i*; он позволяет вводить код SQL в верхнем окне и просматривать результаты в нижнем окне. Кроме этого он может отобразить в графическом виде вывод EXPLAIN PLAN и статистику возвращаемых значений выполнения команды SQL.

Oracle Enterprise Security Manager

Применяется совместно с компонентом Oracle Advanced Security.

Oracle Text

Помогает управлять текстом и осуществлять поиск текста в базе данных Oracle.

Oracle Spatial Index Advisor

Применяется для компонента Oracle Spatial.

Oracle Directory Manager

Применяется для Oracle Internet Directory.

Oracle Forms Server Manager

Управляет и следит за различными серверами, которые работают с Oracle Forms.

Oracle Policy Manager

Управляет метками безопасности.

Oracle Net Manager

Настраивает службы Oracle Net Services и управляет ими.

Oracle Data Guard Manager

Реализует подсистему высокой готовности Data Guard и управляет ей.

Oracle LogMiner Viewer

Предоставляет интерфейс для исследования активности базы данных.

Область навигатора

Расположена справа от панели инструментов и панелей запуска. Предлагает подобный проводнику интерфейс для объектов и представлений консоли EM. Выделив объект и нажав правую кнопку мыши, можно получить перечень действий, доступных для данного объекта. При необходимости можно также копировать объекты, перетаскивая их с одного места на другое в иерархии навигатора.

Область детализации

Для каждого объекта-листа, выбранного в области навигатора, в правой части консоли EM выводится подробная информация. Она может быть представлена в виде списка или перечня свойств, а также как страница с закладками для дополнительной информации.

Кроме того, в верхней части консоли Enterprise Manager представлена строка меню, содержащая элементы File (Файл), Navigator (Навигатор), Object (Объект), Event (Событие), Job (Задание), Tools (Сервис), Configuration (Конфигурация) и Help (Справка). Некоторые из пунктов этих меню могут быть неактивны в каждый конкретный момент времени в зависимости от того, какой объект выбран в области навигатора.

В версии Enterprise Manager, поставляемой вместе с Oracle8*i*, существовало представление по умолчанию, включавшее в себя четыре области детализации: Navigator, Group, Event и Job. Объекты можно было просматривать в окне навигатора, создав разделенное представление – в области навигатора возникала страница с закладками. В Oracle9*i* это представление по умолчанию было изменено; теперь окно навигатора расположено слева, а все остальные окна могут быть включены в единую область детализации, расположенную справа (для этого необходимо выбрать их в иерархии навигатора). В результате получается представление, аналогичное тому, которое предлагалось функцией DBA Studio в предыдущей версии Enterprise Manager.

Представления

Существуют четыре основных способа просмотра информации в области навигатора:

Стандартное представление

Это иерархическое представление, называемое Databases и расположенное в верхней части области навигатора. Для того чтобы показать или скрыть дочерние объекты в данном представлении, необходимо щелкнуть по значку слева от родительского объекта.

Представление групп

Это представление, находящееся под иерархическим представлением Databases, предоставляет возможность организации коллекций объектов в группы и перемещения между получившимися группами. Группы помогают упростить имеющееся окружение или повысить производительность. Например, можно запустить задание для группы, и оно будет выполнено для всех ее элементов. Группы также можно применять для совместного отображения состояний всех элементов группы. Объекты в группу добавляются путем перетаскивания их из навигатора в область детализации Group.

Представление заданий

Это представление служит интерфейсом для системы заданий, применяемой для запуска служебных заданий и управления ими. Область детализации для представления Job содержит страницу с закладками, включающую в себя список активных заданий и страницу истории, на которой приведены выполненные задания и их статус. С любой страницы можно проверить журнал, содержащий подробные данные о ходе выполнения задания.

Представление событий

Данное представление выводит события, зарегистрированные для отслеживания в Enterprise Manager. Область детализации для представления Event содержит страницу с закладками, на которую могут быть выведены еще не удаленные предупреждения, список зарегистрированных событий и страницу истории – события, уже произошедшие и еще не удаленные.

В табл. 16.1 приведен перечень страниц, появляющихся в области детализации при выборе в области навигатора различных узлов, относящихся к базе данных. Если страница с закладками не указана, то при выборе данного узла будет выведен список объектов. При выборе объекта выдается перечень его свойств.

Таблица 16.1. Варианты области детализации

Папка	Узел в панели Навигатора	Страницы с закладками
Instance (Экземпляр)	Configuration (Конфигурация)	General (Общее) Memory (Память) Recovery (Восстановление) Resource Managers (Диспетчеры ресурсов) Undo (Отмена)
	Stored Configurations (Хранимые конфигурации) – доступна только при использовании сервера Management Server)	
	Sessions (Сеансы)	
	Resource Consumer Groups (Группы потребителей ресурсов)	
	Resource Plans (Планы ресурсов)	
Schemas (Схемы)	Tables (Таблицы)	
	Indexes (Индексы)	
	Views (Представления)	
	Synonyms (Синонимы)	
	Sequences (Последовательности)	
	Clusters (Кластеры)	
	Source types (Исходные типы)	
	User-defined types (Пользовательские типы)	Array types (Регулярные типы) Object types (Объектные типы) Table types (Табличные типы)

Таблица 16.1 (продолжение)

Папка	Узел в панели Навигатора	Страницы с закладками
Security (Безопасность)	Users (Пользователи)	
	Roles (Роли)	
	Profiles (Профили)	
Storage (Хранение)	Control file (Управляющий файл)	
	Tablespace (Табличное пространство)	
	Datafile (Файл данных)	General (Общее) Storage (Хранение)
	Rollback segments (Сегменты отката)	
	Redo logs (Журналы)	
	Archive logs (Архивные журналы)	
Distributed (Распределенные)	In-doubt transactions (Сомнительные транзакции)	
	Database links (Каналы БД)	
	Streams (потоки)	Доступна только начиная с Oracle9*i* Release 2
	Queue tables (Таблицы очередей)	
	Advanced replication (Расширенное тиражирование)	Topology (Топология) Errors (Ошибки) Transactions (Транзакции) Schedule (Расписание) Configurations (Конфигурации) DBMS jobs (Задания СУБД)
Warehouse (Хранилище)	OLAP	Measure folders (Каталоги показателей) Cubes folders (Каталоги кубов) Dimensions folders (Каталоги измерений)
	Summaries (Итоги)	Dimensions (Измерения) Materialized view folders (Каталоги материализованных представлений) Materialized view logs folders (Каталоги журналов материализованных представлений) Refresh group folders (Каталоги групп обновления)

Папка	Узел в панели Навигатора	Страницы с закладками
Workspace (Рабочее пространство)	Version-enabled tables (Версионные таблицы) Workspaces (Рабочие пространства)	
XML Data base (БД XML)	Configuration (Конфигурация) Resources (Ресурсы) XML Schema Links (Ссылки схемы XML)	

В дополнение к перечисленным объектам БД область навигатора может включать в себя записи для групп, прослушивающих процессов, узлов и веб-серверов.

Администрирование Enterprise Manager

Основное назначение Enterprise Manager состоит в выполнении административных задач для БД Oracle и в их контроле. Enterprise Manager включает в себя ряд программ-мастеров, помогающих управлять базой данных. Enterprise Manager может выводить отчеты для БД, отслеживать изменения в БД или искать объекты БД.

Enterprise Manager может определять несколько административных пользователей. Для каждого пользователя можно указать ряд атрибутов, в том числе определить, какими узлами, схемами или объектами они могут управлять, а также как и о чем их следует предупреждать.

Enterprise Manager содержит стандартный набор отчетов, но также можно определить свои собственные отчеты. Доступ к отчетам можно получить с автоматически создаваемого и поддерживаемого веб-сайта.

Списки свойств Enterprise Manager и мастера резервного копирования и восстановления позволяют вызвать утилиту Recovery Manager (RMAN, см. главу 15).

Задания

Enterprise Manager позволяет создавать задания, управлять ими и определять для них расписание выполнения. Задание (*job*) – это любая операция, которая может быть выполнена посредством команд SQL, команд операционной системы или сценариев.

При помощи консоли Enterprise Manager можно создать задание и назначить расписание его выполнения для различных узлов. Кроме того можно создать задание для серверов, работающих на разных платформах, т. к. во внутренних сценариях Enterprise Manager применяется не зависящий от платформы язык Tcl (Tool Command Language).

Enterprise Manager содержит несколько предопределенных задач, в основном связанных с управлением базами данных. Эти задачи можно использовать при создании заданий наряду со сценариями SQL*Plus, операционной системы или Tcl.

Сервер Oracle Management Server передает определенное задание определенному агенту Intelligent Agent в предусмотренное расписанием время. Если агент Intelligent Agent недоступен, то сервер Oracle Management Server продолжает пытаться передать задание то тех пор, пока агент не возобновит работу.

Enterprise Manager также отслеживает состояние заданий как в процессе их выполнения, так и после завершения работы.

События

Событие (*event*) происходит в результате выполнения некоторой проверки, инициированной Enterprise Manager. Enterprise Manager включает в себя ряд предопределенных событий, способных отслеживать условия для баз данных, прослушивающих процессов, HTTP-серверов, диспетчера соединений и каждого конкретного узла.

Базовый продукт Enterprise Manager содержит небольшой набор событий «UpDown», применяемых для проверки доступности базы данных, прослушивающего процесса или узла. Дополнительные события можно добавлять из пакетов расширения, которые будут подробно рассмотрены далее в соответствующем разделе. Кроме того, можно добавить собственные дополнительные события; такие события происходят вне области видимости системы Enterprise Manager и посылают ей уведомления о себе.

Можно создавать свои собственные события, которые будут впоследствии реализованы сервером Oracle Management Server. Определяя событие, вы определяете для него критерии и расписание, а также параметры и настройки безопасности, указывающие, каким образом администраторы могут обращаться к событию и изменять его. При создании события, выявляющего какой-то сбой, можно сопоставить ему «ремонтный» сценарий, который будет автоматически запущен и попытается устранить сбой.

При наступлении события изменяется его статус, который может принимать одно из четырех значений:

- Не определен (Unknown) – означает, что невозможно проверить критерий наступления события.
- Критический (Critical)
- Предупреждение (Warning)
- Ошибка (Error) – означает, что проверка завершилась с ошибкой.

При определении события вы указываете, какие из возвращаемых значений приведут к изменению статуса на Critical или Warning.

Страница Event содержит закладки для предупреждений (Alerts), истории событий (History) и зарегистрированных событий (Registered). Со страниц Alerts и History можно вызвать средство просмотра событий Event Viewer для вывода подробной информации о выбранном событии. Каждое событие на страницах Alerts и History помечено флагом определенного цвета, отражающим его статус: серый соответствует статусу Unknown, желтый – Warning, красный – Critical, а желтый шестиугольник с восклицательным знаком – статусу Error.

Enterprise Manager включает в себя обработчик Event Handler (для Oracle8*i* вышла тестовая версия, а для Oracle9*i* появилась уже полномасштабная версия). Event Handler регистрирует события в журнале и может автоматически предпринимать действия по результатам проверки критериев событий. Применение фильтров для Event Handler позволяет ограничить объем регистрируемых и обрабатываемых событий.

Мастера

Мастер (*wizard*) – это автоматическое средство, применяемое для выполнения конкретной задачи и требующее минимального участия пользователя. Enterprise Manager включает в себя следующие мастера:

Мастер анализа (Analyze Wizard)

Собирает и обрабатывает статистику.

Мастера управления резервным копированием и восстановлением (Backup and Recovery Management Wizards)

Помогают осуществить резервное копирование и восстановление баз данных и их элементов, таких как табличные пространства, файлы данных и архивные журналы. Эти мастера предоставляют широкий спектр возможностей и действуют как интерфейс для RMAN (утилите Recovery Manager посвящена глава 15).

Мастер создания таблиц (Create Table Wizard)

Помогает в создании таблиц и столбцов.

Мастер построения кубов (Cube Wizard)

Помогает при создании многомерных кубов данных.

Мастер управления данными (Data Management Wizard)

Выполняет операции импорта, экспорта и загрузку данных.

Мастер создания измерений (Dimension Creation Wizard)

Применяется для создания измерений для хранилища данных.

Мастер распределения ресурсов (Resource Plan Wizard)

Группирует пользовательские сеансы для создания групп распределения ресурсов.

Мастер работы с материализованными представлениями (Summary Advisor Wizard)

Помогает создавать и удалять материализованные представления.

Мастер представлений (View Wizard)

Помогает при создании представлений.

Пакеты расширения

Возможности Enterprise Manager могут быть расширены с помощью специальных дополнительных модулей. Пакеты расширения Oracle – это компоненты, поставляемые за отдельную плату и предназначенные для обеспечения специфических функций управления для определенных видов работ.

Следующие приложения были объединены с версией Enterprise Manager для Oracle8*i*:

Oracle Applications Manager

Oracle Enterprise Security Manager

Oracle8I *interMedia* Text Manager

Oracle Parallel Server Manager

Oracle Spatial Index Advisor

Oracle Directory Manager

Oracle Distributed Access Manager

Oracle Developer Server Forms Manager

Эта версия Enterprise Manager также поставлялась вместе с пакетом DBA Management Pack, описываемым ниже. Функции большинства из перечисленных приложений включены в стандартную версию Enterprise Manager для Oracle9*i*.

Пакет диагностики Diagnostics Pack

Данный пакет включает в себя такие приложения:

Advanced Events

Набор предопределенных событий, предназначенный для проверки таких условий, как избыточное расходование ресурсов и снижение производительности БД, прослушивающих процессов, узлов и HTTP-серверов. Существуют даже события для проверки работы Microsoft SQL Server.

Performance Manager

Создает графики и диаграммы, помогающие следить за производительностью БД в режиме реального времени.

Capacity Planner

Прогнозирует будущие требования к емкости системы на основе сведений, собранных приложением Performance Manager.

TopSessions

Выводит подробную информацию о текущих сеансах и расходуемых ими ресурсах.

Trace Data Viewer

Обеспечивает просмотр данных, собранных Oracle Trace.

В Oracle8*i* существовал отдельный инструмент для управления веб-серверами Apache и Oracle iAS Version 1.0.

Пакет настройки Tuning Pack

Этот пакет включает в себя следующие приложения:

Expert

Помогает администратору выполнить начальное конфигурирование БД Oracle и дает рекомендации по настройке базы данных с целью повышения ее производительности. Expert также генерирует сценарии для реализации таких рекомендаций.

Index Tuning Wizard

Формирует предложения по построению дополнительных индексов для таблиц и готовит сценарии для реализации таких рекомендаций.

SQL Analyze

Анализирует и сравнивает планы оптимизации для команд SQL, а также способствует повышению производительности при помощи подсказок.

Tablespace Map and Analysis

Выводит информацию о табличных пространствах и выполняет анализ на предмет выявления возможных проблем управления пространством.

Reorg Wizard

Может реорганизовать отдельные объекты схемы или табличные пространства или восстанавливать строки при миграции. Выделено в независимое приложение в версии Oracle9*i*.

Outline Management

Помогает управлять хранимыми планами выполнения. Выделено в независимое приложение в версии Oracle9*i*.

Outline Editor

Обеспечивает простое и удобное редактирование хранимых планов выполнения. Выделено в независимое приложение в версии Oracle9*i*.

Auto Analyze

Автоматически поддерживает статистику для БД Oracle8*i*. Это приложение не вошло в версию Oracle9*i*.

Пакет управления изменениями Change Management Pack

Данный пакет представляет собой набор приложений, применяемых для отслеживания и реализации изменений в одной или нескольких базах данных. Пакет включает в себя диспетчер изменений Change Manager, позволяющий выполнять следующие операции:

- Сгенерировать определения объектов БД
- Сравнить два различных набора определений БД
- Синхронизировать два набора определений БД
- Быстро внести изменения в объект, используя таблицу свойств
- Внести изменения в один или несколько объектов БД
- Распространить изменения одной БД на другую
- Отредактировать сформированные планы внесения изменений любого типа

Стандартный пакет управления Standard Management Pack

Данный пакет предназначен для небольших установок и включает в себя некоторые из приложений трех пакетов, описанных ранее, а именно:

Performance Manager

Обеспечивает графическое представление в реальном времени производительности БД Oracle и хостов.

Index Tuning Wizard

Выполняет упреждающую оптимизацию индексов.

Create Baseline

Собирает определения существующих объектов схемы.

Compare Database Objects

Сравнивает две базы данных, две схемы или два набора определений объектов.

Advanced Events for Databases and Nodes

Содержит подмножество событий Advanced Events

Пакет управления для Oracle Applications

Этот пакет включает в себя приложения, предназначенные специально для управления Oracle Applications:

Performance Manager

Выводит информацию о производительности для нескольких экземпляров Oracle Applications.

Capacity Planner

Использует архивные данные для прогнозирования будущих требований к емкости системы.

Concurrent Processing Tuning Assistant

Анализирует записи о выполнении пакетных заданий и вносит предложения по настройке.

Oracle Applications Advanced Events

Определяет события для Oracle Applications

Пакет управления для SAP R/3

Этот пакет включает в себя приложения, предназначенные специально для управления SAP R/3:

Performance Manager

Выводит информацию о производительности для нескольких экземпляров серверов приложений R/3.

Capacity Planner

Использует архивные данные для прогнозирования будущих требований к емкости системы.

Oracle Applications Advanced Events

Определяет события для SAP R/3

DBA Management Pack и DBA Studio

Пакет DBA Management Pack – это группа стандартных приложений, которая вместе с Enterprise Manager была включена в Oracle8*i*. Все эти приложения были интегрированы в Enterprise Manager в версии Oracle9*i*.

DBA Studio входил в состав пакета DBA Management Pack, который поставлялся вместе с версией Enterprise Manager для Oracle8*i*. DBA Studio представлял собой автономный интерфейс для выполнения следующих задач, входивших в состав пакета DBA Management Pack:

- Управление экземплярами
- Управление схемами
- Управление безопасностью
- Управление хранением
- Управление репликацией
- Управление JServer
- Управление кэшами
- Блокнот SQL*Plus

А также ряда мастеров для резервного копирования, восстановления, создания таблиц и представлений, и анализа.

До указанной версии Enterprise Manager данная функциональность обеспечивалась несколькими отдельными инструментами. В версии Enterprise Manager для Oracle9*i* функции DBA Studio интегрированы в основной продукт.

OEMUTIL

OEMUTIL – это утилита с интерфейсом командной строки, появившаяся в Oracle9*i*, обеспечивающая выполнение различных действий с заданиями и событиями. Утилиту можно включать в состав пакетного задания.

Запуск OEMUTIL

Для выполнения отдельной команды при помощи OEMUTIL применяется команда:

```
oemapp oemutil имя_пользователя/пароль@oms команда параметры
```

где *oms* – это имя сервера, на котором работает Oracle Management Server. Для запуска в командной строке любой из команд, описанных в следующем разделе, необходимо указать префикс *имя_пользователя*/*пароль*@*oms*.

Для запуска утилиты OEMUTIL со списком команд выполните следующую команду:

```
oemapp oemutil -cmdfile командный_файл
```

где *командный_файл* – это файл, содержащий необходимые команды OEMUTIL, в том числе и информацию для регистрации пользователя.

Команды OEMUTIL

В разделе будут рассмотрены команды OEMUTIL. Имейте в виду, что если команда выполняется из командной строки, то при наличии пробела или любого специального символа в составе одного из параметров необходимо заключать такой параметр в кавычки.

---

changeCredentials

```
changeCredentials пользовательEM имя_цели пользователь пароль роль
```

Изменяет *реквизиты доступа* (*credentials*) для баз данных в хранилище Enterprise Manager.

Ключевые слова

пользовательEM

Администратор Enterprise Manage, чьи реквизиты доступа будут изменены.

имя_цели

База данных, представляющая собой цель изменения.

пользователь

Пользователь БД.

пароль

Пароль БД.

роль

Роль, сопоставленная пользователю БД: NORMAL, SYSDBA или SYSOPER.

deregisterEvent

```
deregisterEvent событие владелец имя_цели тип_цели
```

Отменяет регистрацию события.

Ключевые слова

событие

Имя события.

владелец

Владелец события.

имя_цели

Узел или группа, для которых отменяется регистрация события.

тип_цели

Тип цели для отмены регистрации события. Допустимы следующие значения:

```
oracle_sysman_database
oracle_sysman_node
oracle_sysman_listener
oracle_sysman_cmanager
oracle_sysman_ops
oracle_sysman_webserver
oracle_sysman_hotstandby
```

omsCredentials

```
omsCredentials имя_пользователя/пароль
```

Указывает имя пользователя и пароль для пользователя Enterprise Manager. Данная команда необходима при использовании командного файла.

registerEventFromLibrary

```
registerEventFromLibrary событие владелец имя_цели [администратор]
```

Регистрирует событие, которое определено в библиотеке событий Event Library.

Ключевые слова

событие

Имя события

владелец

Владелец события.

имя_цели

Узел или группа, для которых производится регистрация события.

администратор

Необязательный параметр – имя администратора Enterprise Manager, который будет получать уведомления из задания.

submitJob

```
submitJob задание узел командаОС параметрыОС
```

Незамедлительно отправляет задание узлу.

Ключевые слова

задание

Имя, присвоенное заданию.

узел

Целевой узел или группа. Указывается в формате

владелец_группы:имя_группы | *oracle_sysman_group*

при использовании в качестве параметра командной строки заключается в двойные кавычки.

командаОС

Запускаемая команда операционной системы.

параметрыОС

Параметры для команды операционной системы.

submitJobFromLibrary

```
submitJobFromLibrary заданиее владелец имя_цели [администратор]
```

Передает задание, которое было сохранено в библиотеке заданий. Это задание будет запущено так, как это определено в библиотеке, с учетом заданного расписания.

Ключевые слова

задание

Имя, присвоенное заданию.

владелец

Владелец задания.

имя_цели

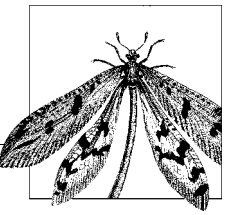
Цель задания. Это может быть имя узла или группы, аналогично команде submitJob.

администратор

Необязательный параметр – имя администратора Enterprise Manager, который будет получать уведомления из задания.

17

Производительность



Достижение оптимальной производительности при работе с БД Oracle является скорее искусством, чем наукой. Применяя подходящие структуры данных, обеспечивая доступность соответствующих решаемой задаче ресурсов и учитывая особенности СУБД Oracle, можно обойти узкие места, ограничивающие производительность системы.

Эта глава предлагает обзор некоторых инструментов из состава Oracle, который поможет понять, как СУБД оптимизирует производительность выполняемых ею операций. В частности в главе описаны:

- Оптимизация SQL, включая типы оптимизаторов и подсказки для них.
- Средства EXPLAIN PLAN, TKPROF и AUTOTRACE, помогающие понять, каким образом оптимизируется SQL.
- Пакеты сбора статистики UTLBSTAT, UTLESTAT и Statspack, предоставляющие статистику производительности и помогающие найти узкие места.

Разумеется, эта глава дает лишь начальные сведения о перечисленных инструментах и их применении; достижение наивысшей производительности требует более глубоких знаний. Дополнительную информацию по этой теме можно получить из документации Oracle и ряда книг о внутренней структуре и настройке производительности Oracle (некоторые из рекомендуемых изданий приведены в приложении E).

Оптимизация SQL

Одно из основных достоинств реляционной СУБД заключается в ее способности к извлечению данных без указания путей доступа к ним. Когда сервер Oracle получает SQL-запрос, ему необходимо выбрать способ доступа к данным. Процесс принятия такого решения называется оптимизацией *запроса*, поскольку Oracle ищет оптимальный способ извлечения данных. Задача поиска наиболее эффективного способа осложняется наличием множества альтернативных вариантов доступа к информации.

Оптимизатор должен определить, какой из путей выполнения окажется наиболее быстрым. При обработке сложных запросов, содержащих, например, соединения по нескольким таблицам или сложные условия отбора и несколько уровней сортировки, оптимизатор решает очень непростую задачу.

Оптимизаторы запросов играют ключевую роль в достижении наивысшей производительности SQL-запросов. До выхода Oracle7 в состав Oracle входил только *оптими-*

затор по синтаксису (*rule-based optimizer*). *Оптимизатор по стоимости* (*cost-based optimizer*) появился в Oracle7.

Оптимизатор по синтаксису

Оптимизатор по синтаксису в полном соответствии со своим названием применяет набор предопределенных правил.

Выбирая путь выполнения запроса, оптимизатор по синтаксису применяет набор правил, приведенных в табл. 17.1, в том же порядке, в котором они расположены в таблице.

Таблица 17.1. Приоритет правил для оптимизатора по синтаксису

Правило	Описание
Единственная строка по ROWID	Применять ROWID в инструкции WHERE или CURRENT OF CURSOR.
Единственная строка по кластерному соединению	Применять кластерный ключ в инструкции WHERE, запрос возвращает только одну строку.
Единственная строка по ключу хэшированного кластера с уникальным или первичным ключом	Использовать все столбцы ключа хешированного кластера в условии WHERE, запрос возвращает только одну строку.
Единственная строка по уникальному или первичному ключу	Применять уникальный или первичный ключ в инструкции WHERE, запрос возвращает только одну строку.
Кластеризованное соединение	Использовать, если все таблицы входят в кластер и все столбцы кластерного ключа входят в инструкцию WHERE в условиях равенства.
Ключ хеш-кластера	Использовать, если все столбцы ключа хешированного кластера входят в инструкцию WHERE в условиях равенства.
Ключ индексного кластера	Применять, если все столбцы ключа индексного кластера входят в инструкцию WHERE в условиях равенства.
Составной индекс	Применять, если все столбцы составного индекса входят в инструкцию WHERE в условиях равенства.
Одностолбцовый индекс	Применять, когда столбцы одностолбцовых индексов входят в инструкцию WHERE в условиях равенства.
Поиск в ограниченном диапазоне по индексированному столбцу	Применять, когда индексированный столбец входит в инструкцию WHERE в условии принадлежности ограниченному диапазону.
Поиск в неограниченном диапазоне по индексированному столбцу	Применять, когда индексированный столбец входит в инструкцию WHERE в условии принадлежности неограниченному диапазону.
Сортировка соединения слиянием	Применять, когда столбцы соединяемых таблиц входят в инструкцию WHERE в условии равенства.

Таблица 17.1 (продолжение)

Правило	Описание
Максимум или минимум в индексированном столбце	Использовать, если вычисляются максимум и минимум индексированного столбца и отсутствуют инструкции WHERE и ORDER BY.
Сортировка по индексированному столбцу	Использовать одностолбцовый индекс или начальную часть многостолбцового в инструкции ORDER BY, если в нее входит хотя бы один столбец с ограничениями PRIMARY KEY или NOT NULL, гарантирующими отсутствие значений NULL.
Полный просмотр таблицы	Если ни одно из предыдущих правил не применимо, выполнить полный просмотр таблицы.

Оптимизация по синтаксису дает лучшие результаты, чем случайная оптимизация[1] SQL-запросов, но и у нее есть слабые стороны. Одна из них заключается в слишком упрощенном наборе правил. В сложных базах данных запросы часто строятся по нескольким таблицам, имеющим по несколько индексов, с применением сложных условий и сортировок. Результатом такой сложности становится обилие путей выполнения, и слишком простой набор правил не позволяет сделать наилучший выбор.

Оптимизатор выбирает, какие из правил, приведенных в таблице, применимы к запросу. Из выбранных правил применяется то, которое имеет наибольший приоритет. Но в сложных запросах может потребоваться делать выбор из нескольких таблиц с одностолбцовыми индексами. В этом случае оптимизатор учитывает синтаксис SQL-запроса для разрешения противоречия. Выбор окончательного пути определяется тем, в каком порядке таблицы упоминаются в команде SQL. Таким образом, две функционально идентичных команды могут порождать разные планы выполнения в зависимости от порядка вхождения таблиц.

Оптимизатор по стоимости

Для улучшения качества оптимизации SQL-запросов в состав СУБД Oracle начиная с версии 7 входит оптимизатор по стоимости. В соответствии с названием в его задачу входит не только простой выбор правил из заданного набора. Он выбирает тот план выполнения, который требует наименьшего количества операций логического ввода/вывода и, следовательно, имеет наименьшую стоимость выполнения. Такой подход позволяет избежать потенциальных проблем, рассмотренных выше.

Оптимизация по стоимости в Oracle8 и последующих версиях возможна только при наличии статистики хотя бы по одной таблице или индексу.

Статистика

Стоимость любого фрагмента плана выполнения зависит от используемых в нем структур данных. При поиске оптимального плана выполнения оптимизатор по

[1] Случайная оптимизация (random optimization) – название класса алгоритмов, предназначенных для решения задач оптимизации. Случайная оптимизация сравнительно малоизвестна и может быть сравнена с генетическими алгоритмами. Часто случайная оптимизация приводит к лучшим результатам, чем другие методы с более быстрой сходимостью. Более подробно о случайной оптимизации можно прочитать в J. Matyas «Random optimization», Automation and remote control, vol 26, pp. 246–253. – *Примеч. науч. ред.*

стоимости основывается на статистической информации о данных[1] (например, на сведениях о размере данных в таблицах и индексах и их уникальности).

Типы статистик, применяемых оптимизатором по стоимости, приведены в табл. 17.2.

Таблица 17.2. Статистики, применяемые при анализе стоимости выполнения

Сущность	Тип статистики
Таблица	Количество строк
	Количество блоков
	Количество неиспользованных блоков
	Среднее доступное свободное пространство в блоке
	Количество расщепленных строк
	Средняя длина строки
	Приблизительная средняя длина строки
Столбец	Количество различающихся значений в столбце (кардинальность)
	Приблизительная кардинальность
	Второе по меньшинству значение в столбце
	Второе по величине значение в столбце
	Коэффициент плотности столбца
	Количество значений NULL в столбце
	Коэффициент распределения данных
Индекс	Глубина B*-дерева индекса
	Количество листовых блоков
	Количество различающихся значений
	Среднее количество листовых блоков для ключа
	Среднее количество блоков данных для ключа
	Фактор кластеризации
Система	Производительность и загруженность ввода/вывода (появилась в Oracle9*i*)
	Производительность и загруженность процессора (появилась в Oracle9*i*)

Сбор статистики

Качество любой оптимизации по стоимости определяется достоверностью лежащей в ее основе статистики. А для того чтобы она была достоверной в БД Oracle, необходимо периодически выполнять операцию сбора статистики.

В версиях, предшествующих Oracle8*i*, сбор статистики выполнялся командой ANALYZE. В Oracle8*i* появился пакет DBMS_STATS, предоставляющий дополнительные возможности, такие как параллельная работа и сбор статистики для глобальных и секционированных объектов. Более подробно этот пакет рассмотрен в главе 10.

Можно собрать статистику для всей базы данных или для отдельной структуры данных, а можно воспользоваться выборочным методом и оценить статистику для некоторой структуры данных по ее фрагментам.

---

[1] Если какая-либо из статистик отсутствует, оптимизатор по стоимости возьмет фиксированное значение по умолчанию. – *Примеч. науч. ред.*

В общем случае необходимость в сборе статистики возникает при значительных изменениях в объеме или организации данных, например после загрузки данных или создания нового индекса. Обновление статистики должно стать частью общей процедуры сопровождения БД. С помощью процедур пакета DBMS_STATS можно выполнять автоматический сбор статистики для определенной таблицы.

Хранимые статистики

Оптимизатор по стоимости может, основываясь на разных статистиках, выбирать разные планы выполнения. Если вы считаете, что оптимизация выполняется правильно, и не хотите, чтобы новая статистика изменила планы выполнения, то можете перед обновлением статистики для базы данных сохранить ее с помощью процедуры DBMS_STATS.EXPORT_SCHEMA_STATS.

Если новый набор статистик не привел к желаемому увеличению производительности, можно реимпортировать сохраненные ранее статистики процедурой DBMS_STATS.IMPORT_SCHEMA_STATS.

Кроме того, в таблице можно хранить несколько версий статистик.

Гистограммы

Оптимизатор по стоимости предполагает, что значения столбца имеют равномерное распределение. Например, если в столбце присутствуют два различных значения, оптимизатор считает, что на каждое из них приходится по 50% записей.

Для столбцов с распределением, отличным от равномерного, такое предположение может оказаться ошибочным, что приведет к неверному выбору плана выполнения. Построение гистограмм поможет избежать этой проблемы.

Гистограммы, появившиеся в Oracle8*i*, дают оптимизатору более детальное представление о распределении значений данных в столбце. Для их создания существуют процедуры в пакете DBMS_STATS. Построение гистограмм требует определенных дополнительных ресурсов, поэтому не следует применять их по умолчанию для всех столбцов. В то же время гистограммы могут помочь в создании более точных планов выполнения, включающих столбцы с низкой селективностью.

Гистограммы появились в Oracle8*i*.

Режимы оптимизатора

Можно указать, какой из оптимизаторов применять и как именно использовать оптимизатор по стоимости, выбрав один из режимов:

RULE

Применять оптимизатор по синтаксису.

CHOOSE

Если ни для одной из таблиц в запросе нет статистики, применяется оптимизатор по синтаксису. Если для некоторых таблиц статистики есть, то оптимизатор основывается на них и делает предположения о статистиках для остальных таблиц.

ALL_ROWS

Оптимизатор по стоимости выбирает план выполнения, возвращающий все строки запроса быстрее остальных. Такой режим более всего подходит для запросов к хранилищам данных.

FIRST_ROWS[(*n*)]

Оптимизатор по стоимости выбирает план выполнения, быстрее всего возвращающий первые *n* строк запроса. Параметр *n* в этом ключевом слове, появившемся в Oracle9*i*, может принимать значения 1, 10, 100 или 1000. Этот режим оптимален для OLTP-запросов, когда после извлечения нескольких строк происходит взаимодействие с пользователем.

Перечисленные режимы могут быть установлены различными способами. Можно задать параметр инициализации OPTIMIZER_MODE. Можно установить режим интерактивно, выполнив команду

```
ALTER SESSION SET OPTIMIZER_GOAL = режим_оптимизатора
```

Можно, наконец, дать *подсказку* (*hint*) для отдельной команды SQL. Этот способ рассмотрен в следующем разделе.

Вы можете сами выбрать оптимизатор, но мы рекомендуем всегда применять оптимизатор по стоимости. Несмотря на то что этот оптимизатор вызывал нарекания в первых выпусках, за прошедшие с тех пор годы он был значительно улучшен. Дело не только в том, что оптимизатор по стоимости имеет теоретические и практические преимущества, но и в том, что он учитывает новые возможности Oracle, такие как материализованные представления, не доступные оптимизатору по синтаксису.

При определенных условиях, например, при запросе к секционированным таблицам, оптимизатор по стоимости будет использован, даже если режим оптимизации явно установлен в RULE или в CHOOSE, и при этом отсутствует статистика. Когда сервер Oracle вызывает оптимизатор по стоимости, не имея статистики, за основу берется статистика по умолчанию.

Подсказки

Подсказки применяются для того, чтобы сообщить оптимизатору, какой метод следует использовать для оптимизации отдельного запроса.

Подсказки включаются в SQL-запрос как комментарии, соответствующие следующему синтаксису:

```
sql_action /*+ подсказка */
```

или

```
sql_action --+ подсказка
```

где *sql_action* – это команда SELECT, INSERT, UPDATE или DELETE. Подсказка может также содержать в себе текст комментария для целей документирования. В любой команде может быть несколько подсказок. Если подсказка некорректна или ошибочна, Oracle игнорирует ее, не сообщая об ошибке.

В следующих разделах приведено описание подсказок по категориям: подсказки режима оптимизатора, подсказки пути доступа, подсказки преобразования запросов, подсказки соединений, подсказки параллельного выполнения, прочие подсказки.

Подсказки режима оптимизации

В качестве таких подсказок можно указывать приведенные выше названия режимов оптимизации: RULE, CHOOSE, ALL_ROWS и FIRST_ROWS.

Подсказки пути доступа

Следующие подсказки указывают оптимизатору запросов определенный путь доступа к данным:

FULL(*таблица*)

Задает полный просмотр для *таблицы*.

ROWID(*таблица*)

Задает просмотр по ROWID для *таблицы*.

CLUSTER(*таблица*)

Задает кластерное сканирование для *таблицы*.

HASH(*таблица*)

Задает хеш-сканирование для *таблицы*.

INDEX(*таблица* [*индекс* . . .])

Задает просмотр по индексу. Если в подсказке указан один индекс, именно он и будет использован. Если же подсказка содержит имена нескольких индексов, то SQL-оптимизатор выберет тот из них, который обеспечивает наименьшую стоимость. Если название индекса не указано, то SQL-оптимизатор использует тот индекс *таблицы*, который обеспечивает наименьшую стоимость.

INDEX_ASC(*таблица* [*индекс* . . .])

Задает просмотр по индексу в порядке возрастания индексированных значений. Синтаксис аналогичен подсказке INDEX.

INDEX_COMBINE(*таблица* [*индекс* . . .])

Означает использование указанных битовых индексов. Синтаксис аналогичен подсказке INDEX.

INDEX_JOIN(*таблица* [*индекс* . . .])

Задает применение индексного соединения. Синтаксис аналогичен подсказке INDEX. Появилась в Oracle8*i*.

INDEX_DESC(*таблица* [*индекс*. . .])

Задает просмотр по индексу в порядке убывания индексированных значений. Синтаксис аналогичен подсказке INDEX.

INDEX_FFS(*таблица* [*индекс* . . .])

Означает, что вместо полного сканирования таблицы следует выполнить *быстрое полное сканирование индекса* (*fast full index scan*). Синтаксис аналогичен подсказке INDEX.

NO_INDEX(*таблица* [*индекс* . . .])

Явно запрещает использование перечисленных индексов. Появилась в Oracle8*i*.

AND_EQUAL(*таблица индекс индекс* [*индекс* . . .])

Задает просмотр по слиянию указанных одностолбцовых индексов.

Подсказки преобразования запросов

Следующие подсказки сообщают оптимизатору, как следует преобразовать переданный запрос:

USE_CONCAT

Вызывает преобразование списка инструкции OR в составной запрос UNION ALL.

NO_EXPAND

Запрещает расширение условия OR для запроса. Появилась в Oracle8*i*.

REWRITE

Заставляет использовать материализованные представления. Появилась в Oracle8*i*.

EXPAND_GSET_TO_UNION

Вызывает преобразование запроса, основывающегося на *наборах группировок* (*grouping sets*), в составной запрос UNION ALL. Появилась в Oracle9*i*.

NOREWRITE

Запрещает перезапись запроса в материализованное представление. Появилась в Oracle8*i*.

MERGE(*таблица*)

Означает слияние представления с запросом. Эту подсказку можно использовать, только если разрешено слияние сложных представлений.

NO_MERGE(*таблица*)

Запрещает слияние сложных представлений.

STAR_TRANSFORMATION

Предлагает, но не заставляет выполнить преобразование запроса к типу «звезда».

FACT(*таблица*)

Означает, что таблица будет использована как таблица фактов при преобразовании типа «звезда». Появилась в Oracle9*i*.

NO_FACT(*таблица*)

Означает, что таблица не должна использоваться как таблица фактов при преобразовании типа «звезда». Появилась в Oracle9*i*.

Подсказки соединений

Подсказки данного раздела предлагают определенный тип операции соединения или порядок ее выполнения:

ORDERED

Означает, что соединение таблиц выполняется в том же порядке, в каком они приведены в инструкции FROM.

STAR

Заставляет использовать план запроса типа «звезда».

USE_NL(*таблица* [*таблица* . . .])

Вынуждает использовать соединения типа вложенного цикла, при этом указанные таблицы будут выступать в качестве внутренних.

USE_MERGE(*таблица* [*таблица* . . .])

Вынуждает использовать соединения типа сортировка-слияние, при этом указанные таблицы соединяются со строками, полученными в результате соединения предыдущих таблиц.

USE_HASH(*таблица* [*таблица* . . .])

Означает, что для данной таблицы будет применяться хеш-соединение.

DRIVING_SITE(*таблица*)

Означает, что соединение должно выполняться на том компьютере, где расположена указанная таблица.

LEADING(*таблица*)

Указывает, что данная таблица будет первой в соединении. Появилась в Oracle9*i*.

HASH_AJ, MERGE_AJ, NL_AJ

Применяется в подзапросе NOT IN для задания антисоединения хешированием, слиянием или вложенными циклами. Подсказка NL_AJ появилась в Oracle9*i*.

HASH_SJ, MERGE_SJ, NL_SJ

Задает полусоединение хешированием, слиянием или вложенными циклами. Подсказка NL_SJ появилась в Oracle9*i*.

Подсказки параллельного выполнения

Следующие подсказки управляют параллельным выполнением запросов, характерных для хранилищ данных:

PARALLEL(*таблица*, [*целое* | *DEFAULT*][,{*целое* | *DEFAULT*}])

Определяет степень параллелизма операций SQL для указанной таблицы. Первый параметр указывает степень параллелизма для операций над таблицей, а второй (необязательный параметр, появившийся в версии Oracle9*i*) определяет, каким образом таблица должна быть разделена между экземплярами БД в реализации Real Application Clusters. Значение DEFAULT означает, что запрос должен использовать ту степень параллелизма, которая установлена соответствующим параметром инициализации.

NOPARALLEL(*таблица*)

Отменяет все установленные параметры параллельной работы и запрещает параллельные операции с таблицей.

PQ_DISTRIBUTE(*таблица*[,] *внешнее_распределение, внутреннее_распределение*)

Определяет, каким образом строки соединенных таблиц должны быть распределены между *серверами запросов производителей* (*producer query servers*) и *серверами запросов потребителей* (*consumer query servers*). Пара ключевых слов *внешнее_распределение*, *внутреннее_распределение* может иметь шесть комбинаций значений:

Hash, Hash
Broadcast, None
None, Broadcast
Partition, None
None, Partition
None, None

PARALLEL_INDEX(*таблица*, [*целое* | *DEFAULT*][,{*целое* | *DEFAULT*}])

Определяет количество серверов, которые могут одновременно применяться для параллельного диапазонного сканирования индексов. Ключевые слова имеют те же значения, что и для подсказки PARALLEL.

NOPARALLEL_INDEX(*таблица*)

Перекрывает любые параметры параллельного исполнения и запрещает параллельный просмотр индексов.

Прочие подсказки

Данные подсказки относятся к прочим возможностям оптимизации:

APPEND

Разрешает применение режима прямой вставки в команде INSERT при работе в последовательном режиме (т. е. если не используется Enterprise Edition).

NOAPPEND

Запрещает прямую вставку для команды INSERT.

CACHE(*таблица*)

При выполнении полного просмотра помещает таблицу в кэш буферов в тот конец списка LRU, который соответствует наименее давно использованным элементам (как правило, для небольших справочных таблиц).

NOCACHE

При выполнении полного просмотра помещает таблицу в кэш буферов в тот конец списка LRU, который соответствует наиболее давно использованным элементам. В версии Oracle9*i* Release 2 небольшие таблицы кэшируются автоматически.

UNNEST

Указывает на необходимость слияния подзапроса с родительским запросом. Эта подсказка появилась в Oracle9*i*.

NO_UNNEST

Отключает подсказку UNNEST для определенных блоков запроса. Эта подсказка появилась в Oracle9*i*.

PUSH_PRED(*таблица*)

Преобразует предикат соединения в представление. В версии Oracle8 эта подсказка называлась PUSH_JOIN_PRED.

NO_PUSH_PRED(*таблица*)

Запрещает преобразование предиката соединения в представление. В версии Oracle8 эта подсказка называлась NO_PUSH_JOIN_PRED.

PUSH_SUBQ

Перемещает не участвующие в соединении подзапросы в начало плана выполнения. Применение имеет смысл, если стоимость подзапроса невелика и он заметно ограничивает количество извлекаемых строк.

NO_PUSH_SUBQ

Перемещает не участвующие в соединении подзапросы в конец плана выполнения. Появилась в Oracle9*i*.

ORDERED_PREDICATES

Указывает оптимизатору, что необходимо сохранить порядок предикатов в инструкции WHERE. Эта подсказка применяется в инструкции WHERE. Появилась в Oracle8*i*.

CURSOR_SHARING_EXACT

Запрещает замену литералов переменными связывания. Такой тип оптимизации регулируется параметром инициализации CURSOR_SHARING. Появилась в Oracle9*i*.

DYNAMIC_SAMPLING([*таблица*] *целое*)

Позволяет более точно определить оценки для избирательности и кардинальности. Для команды или для отдельной таблицы в рамках команды можно задать уровень динамической выборки от 0 до 10 (подробную информацию об этих уровнях вы найдете в документации Oracle). Появилась в Oracle9*i*.

Хранимые планы выполнения и их стабильность

Оптимизатор по стоимости оценивает команды SQL на основе текущей стоимости операций. Вычисление этой стоимости основывается на имеющейся статистике относительно базы данных и содержащихся в ней данных.

Вы можете решить, что ваши SQL-команды достаточно оптимизированы и вам не хотелось бы в дальнейшем изменять их. В этом случае можно создать *хранимый план выполнения* (*stored outline*), основывающийся на подсказках для управления процессом оптимизации команды.

При работе с хранимым планом выполнения изменения в БД, которые могли бы привести к выбору другого оптимального плана выполнения, не повлияют на оптимизацию команды. Хранимый план выполнения может гарантировать, что во всех инсталляциях системы будет применяться одна и та же схема оптимизации. Можно создать хранимые планы выполнения перед переходом на более новую версию Oracle, чтобы убедиться, что новая версия оптимизатора обеспечивает хотя бы не меньшую производительность.

Корпорация Oracle также связывает применение хранимых планов выполнения со *стабильностью планов выполнения* (*plan stability*).

Создание хранимых планов выполнения

Хранимые планы выполнения автоматически создаются для всех команд SQL, если значение параметра CREATE_STORED_OUTLINE равно TRUE.

Хранимый план выполнения для отдельной команды можно создать следующим образом:

```
CREATE [OR REPLACE] [PUBLIC | PRIVATE] OUTLINE имя_плана
    [FROM {PUBLIC | PRIVATE} исходный_план]
    FOR CATEGORY категория
    ON SQL_команда
```

Инструкция CATEGORY позволяет сгруппировать планы выполнения по категориям. Если никакая категория не указана, то планы выполнения помещаются в категорию по умолчанию. Планы выполнения категории PUBLIC (категория по умолчанию) доступны всем пользователям, входящим в группу PUBLIC. Планы выполнения категории PRIVATE будут кратко описаны далее.

Планы выполнения создаются и редактируются средствами пакетов DBMS_OUTLN и DBMS_OUTLIN_EDIT (пакеты были подробно рассмотрены в главе 10). В Oracle9*i* для создания и редактирования планов выполнения можно также применять Enterprise Manager. Планы выполнения хранятся в таблицах схемы SYS и доступны в представлениях *_OUTLINES и *_OUTLINE_HINTS (за информацией об этих представлениях обратитесь к главе 6).

Применение планов выполнения

Для работы с хранимыми планами выполнения необходимо установить параметр USE_STORED_OUTLINES в TRUE. После параметра может быть указано имя категории, тогда сервер Oracle будет искать хранимый план выполнения сначала в названной категории, а затем в категории по умолчанию. Если имя категории не указано, то сервер Oracle будет просматривать только категорию по умолчанию.

Закрытые планы выполнения

Закрытый план выполнения (*private outline*) доступен только текущему сеансу. Применение закрытых планов выполнения регулируется параметром USE_PRIVATE_OUTLINE.

Для работы с такими планами выполнения необходимо, чтобы в текущей схеме присутствовали таблицы планов выполнения. Для создания этих таблиц применяется сценарий *UTLEDITOL.SQL* или процедура DBMS_OUTLN_EDIT.CREATE_EDIT_TABLES.

EXPLAIN PLAN

Оптимизатор Oracle должен выбрать для команд SQL наилучший план выполнения. В случае возникновения проблем с производительностью какой-то команды у пользователя может возникнуть желание посмотреть, какой именно путь выполнения был выбран оптимизатором. Такую информацию может предоставить команда EXPLAIN PLAN.

План выполнения, выведенный командой EXPLAIN PLAN, может не полностью соответствовать реальному плану, примененному в процессе выполнения, из-за различий двух окружений. Появившиеся в версии Oracle9*i* представления словаря данных SQL_PLAN содержат информацию о фактическом плане, примененном при выполнении запроса. Сведения о таких представлениях можно найти в главе 6.

Подготовка к запуску EXPLAIN PLAN

Команде EXPLAIN PLAN необходима таблица для хранения собираемой информации. По умолчанию эта таблица называется PLAN_TABLE и создается сценарием *UTLXPLAN.SQL* в схеме пользователя.

Для того чтобы поместить сведения, формируемые командой EXPLAIN PLAN, в другую таблицу, обратитесь к рекомендованному корпорацией Oracle сценарию *UTLXPLAN.SQL*, а затем переименуйте таблицу.

В следующих версиях выводимая командой EXPLAIN PLAN информация может измениться, поэтому корпорация Oracle рекомендует удалять и заново создавать таблицу PLAN_TABLE при каждом переходе на новую версию ПО.

Выполнение EXPLAIN PLAN

Команду EXPLAIN PLAN можно выполнить следующим образом:

```
EXPLAIN PLAN [SET STATEMENT_ID = 'ид_команды']
    [INTO [схема.]таблица [@dblink]] FOR команда
```

Инструкция STATEMENT_ID означает вывод плана для одной конкретной команды. Инструкция INTO указывает, что информации плана будет храниться в таблице, отличной от PLAN_TABLE; эта таблица должна существовать на момент выполнения команды. Параметр *команда* определяет, для какой команды SQL будут выданы разъяснения.

Для вывода планов выполнения можно написать собственную команду SQL, обращающуюся к таблице PLAN_TABLE, или же использовать сценарий *UTLXPLS.SQL* при последовательном выполнении или *UTLXPLP.SQL* – при параллельном.

Столбцы PLAN_TABLE

Информация, полученная командой EXPLAIN PLAN, сохраняется в таблице PLAN_TABLE или какой-то другой таблице, названной в команде. Такая таблица состоит из следующих столбцов (описание каждого столбца включает также его тип данных, например, VARCHAR2, DATE и т. д.):

STATEMENT_ID

Идентификатор команды. Тип VARCHAR2(30).

TIMESTAMP

Время запуска команды EXPLAIN PLAN. Тип DATE.

REMARKS

Содержит любые примечания, которые предполагается добавить к строке. Тип VARCHAR2(80).

OPERATION

Первая строка таблицы PLAN_TABLE содержит одно из перечисленных значений:

```
DELETE STATEMENT
INSERT STATEMENT
SELECT STATEMENT
UPDATE STATEMENT
```

Последующие строки таблицы PLAN_TABLE содержат комбинацию значений данного столбца и столбца OPTION, возможные значения приведены в табл. 17.3. Тип VARCHAR2(30).

OPTION

Описывает возможности для операций из столбца OPERATION. Тип VARCHAR2(225). Пары значений двух последних столбцов перечислены в табл. 17.3.

Таблица 17.3. Значения OPERATION и OPTION

OPERATION	OPTION	Описание
AND-EQUAL		Пересечение наборов данных.
BITMAP	CONVERSION (TO ROWIDS | FROM ROWIDS | COUNT) INDEX (SINGLE VALUE | RANGE SCAN | FULL SCAN) MERGE MINUS OR AND KEY ITERATION	Использует битовые индексы. AND и KEY ITERATION появились в Oracle9*i*.
CONCATENATION		Объединение нескольких наборов данных.
CONNECT BY		Использует инструкцию CONNECT BY для иерархического извлечения.
COUNT		Подсчитывает количество строк.
	STOPKEY	Подсчет ограничен выражением ROWNUM в инструкции WHERE.
DOMAIN INDEX		Использует домен для извлечения ROWID. Появилась в Oracle8*i*.
FILTER		Удаляет некоторые строки из набора.
FIRST ROW		Возвращает первую строку запроса.
FOR UPDATE		Извлекает и блокирует строки.
HASH JOIN	ANTI SEMI	Тип хэш-соединения.
INDEX	RANGE SCAN [DESCENDING] FULL SCAN [DESCENDING] FAST FULL SCAN SKIP SCAN UNIQUE SCAN	Использует индекс. FULL SCAN, FAST FULL SCAN и SKIP SCAN появились в Oracle9*i*. UNIQUE SCAN отсутствует в Oracle9*i*.
INLIST ITERATOR		Последовательно использует значения из списка IN.
INTERSECTION		Удаляет дубликаты из двух наборов строк.
MERGE JOIN	OUTER ANTI SEMI CARTESIAN	Типы соединений слиянием. CARTESIAN появилась в Oracle9*i*.

Таблица 17.3 (продолжение)

OPERATION	OPTION	Описание
MINUS		Возвращает строки, входящие в первое, но не входящие во второе множество.
NESTED LOOPS	OUTER	Сравнивает каждый элемент внешнего набора с каждым элементом внутреннего и возвращает те из них, которые соответствуют условию.
PARTITION	SINGLE ITERATOR ALL INLIST INVALID CONCATENATED	Обращается к разделам. В Oracle8 единственной доступной опцией была CONCATENATED.
PROJECTION		Внутренняя операция, которая не включена в Oracle9*i*.
REMOTE		Извлекает данные из удаленной базы данных.
SEQUENCE		Обращается к значениям последовательности.
SORT	AGGREGATE UNIQUE GROUP BY JOIN ORDER BY	Типы операции сортировки.
TABLE ACCESS	FULL SAMPLE CLUSTER HASH BY ROWID BY ROWID RANGE SAMPLE BY ROWID RANGE BY USER ROWID BY INDEX ROWID BY GLOBAL INDEX ROWID BY LOCAL INDEX ROWID	Типы извлечения данных из таблицы. SAMPLE и SAMPLE BY ROWID RANGE появились в Oracle9*i*. BY ROWID, BY ROWID RANGE, BY USER ROWID, BY INDEX ROWID, BY GLOBAL INDEX ROWID и BY LOCAL INDEX ROWID появились в Oracle8*i*. BY ROWID не поддерживается после Oracle8.
UNION		Использует UNION.
VIEW		Использует представление.

OBJECT_NODE

Имя связи базы данных, используемой для ссылки на объект, или порядок использования результатов данной операции для локальных параллельных запросов. Тип VARCHAR2(128).

OBJECT_OWNER

Имя владельца схемы, содержащей таблицу или индекс. Тип VARCHAR2(30).

OBJECT_NAME

Имя объекта. Тип VARCHAR2(30).

OBJECT_INSTANCE

Порядковый номер объекта в исходной команде. Тип NUMERIC.

OBJECT_TYPE

Описательная информация об объекте. Тип VARCHAR2(30).

OPTIMIZER

Текущий режим оптимизации. Тип VARCHAR2(255).

SEARCH_COLUMNS

Не используется в Oracle9*i*. Тип NUMERIC.

ID

Идентификатор этапа в плане выполнения. Тип NUMERIC.

PARENT_ID

Идентификатор этапа, работающего с выводимой данным этапом информацией. Тип NUMERIC.

POSITION

Для первой строки вывода EXPLAIN PLAN – оценка стоимости выполнения операции. Для последующих строк – позиция по отношению к другим дочерним элементам этого же этапа плана. Тип NUMERIC.

COST

Внутренняя стоимость операции по расчетам оптимизатора. Применяется для относительного сравнения с другими стоимостями. Тип NUMERIC.

CARDINALITY

Оценивает количество строк, к которым обращается операция. Тип NUMERIC.

BYTES

Оценивает количество байт, к которым обращается операция. Тип NUMERIC.

OTHER_TAG

Может содержать одно из перечисленных значений:

```
пустое
SERIAL_FROM_REMOTE (S ? R)
SERIAL_TO_PARALLEL (S ? P)
PARALLEL_TO_PARALLEL (P ? P)
PARALLEL_TO_SERIAL (P ? S)
PARALLEL_COMBINED_WITH_PARENT (PWP)
PARALLEL_COMBINED_WITH_CHILD (PWC)
```

Подробно об этих значениях написано в документации Oracle. Тип VARCHAR2(255).

PARTITION_START

Начальная позиция диапазона разделов, к которым происходит обращение. Возможны следующие значения: целое число, представляющее собой номер раздела; ключевое слово KEY, означающее, что ключевое значение будет определено в процессе выполнения; ROW_REMOVE_LOCATION, указывающее, что будет использовано местоположение записи в момент выполнения; или же ключевое слово INVALID, указывающее, что диапазон разделов пуст. Тип VARCHAR2(255).

PARTITION_STOP

Конечная позиция диапазона разделов, к которым происходит обращение. Ключевые слова те же, что и для PARTITION_START. Тип VARCHAR2(255).

PARTITION_ID

Идентификатор этапа, на котором вычисляются PARTITION_START и PARTITION_STOP. Тип NUMERIC. В Oracle8*i* этот столбец назывался PID.

OTHER

Дополнительная информация об этапе. Тип LONG.

DISTRIBUTION

Метод, применяемый для распределения строк от серверов запросов производителей между серверами запросов потребителей. Тип VARCHAR2(30). Может содержать одно из перечисленных значений:

```
PARTITION (ROWID)
PARTITION (KEY)
HASH
RANGE
ROUND-ROBIN
BROADCAST
QC (ORDER)
QC (RANDOM)
```

За дополнительной информацией об этих значениях обращайтесь к документации Oracle.

CPU_COST

Стоимость в терминах тактов процессора. Тип NUMERIC. Появился в Oracle9*i*.

IO_COST

Стоимость в терминах операций ввода/вывода. Тип NUMERIC. Появился в Oracle9*i*.

TEMP_SPACE

Оценивает объем временного пространства, необходимый операции. Тип NUMERIC. Появился в Oracle9*i*.

TKPROF

Команда EXPLAIN PLAN помогает пользователю понять, какие планы выполнения были применены для отдельных команд SQL. Утилита TKPROF применяется в сочетании с файлами, создаваемыми утилитой SQL Trace, собирающей информацию о выполненных командах. Утилита TKPROF преобразует информацию, собранную SQL Trace, в удобный для восприятия формат.

Подготовка к трассировке

Прежде чем применять TKPROF, необходимо собрать информацию о командах SQL в файл трассировки. Для работы с утилитой SQL Trace необходимо выполнить следующие изменения в файле инициализации целевой БД Oracle:

- Значение параметра TIMED_STATISTICS должно быть равно TRUE. В версии Oracle9*i* Release 2 установка параметра STATISTICS_LEVEL в значение TYPICAL или ALL приведет к сбору статистики по времени, а значение DB_CACHE_ADVICE.TIMED_STATISTICS или TIMED_OS_STATISTICS подменит значение, заданное для STATISTICS_LEVEL.
- Значение параметра MAX_DUMP_FILE_SIZE должно быть достаточно большим, чтобы обеспечить возможность хранить всю информацию, записанную в файл трассировки.
- Параметр USER_DUMP_DEST должен указывать на существующий каталог.

Утилита SQL Trace записывает все файлы трассировки в каталог *USER_DUMP_DEST*, поэтому необходимо быть уверенным в том, что удастся каким-то способом распознать конкретный файл трассировки.

Все эти параметры представляют собой параметры сеанса и могут динамически изменяться начиная с версии Oracle9*i*.

Сбор трассировочной информации

Для сбора трассировочной информации SQL можно вызвать процедуру DBMS_SESSION.SET_SQL_TRACE или выполнить команду

```
ALTER SESSION SET SQL_TRACE = TRUE;
```

Для отключения трассировки можно установить SQL_TRACE в FALSE.

Для того чтобы включить SQL Trace для сеанса, не являющегося текущим, применяется процедура DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION.

Трассировку SQL для всего экземпляра можно включить, установив параметр инициализации SQL_TRACE в TRUE или выполнив команду

```
ALTER SYSTEM SET SQL_TRACE = TRUE;
```

Последний способ следует применять с осторожностью, т. к. сбор трассировочной информации приводит к увеличению накладных расходов.

За подробными сведениями об утилите Oracle Trace, собираемой ею статистике и способах ее применения, обращайтесь к документации Oracle.

Запуск TKPROF

Утилита TKPROF вызывается из командной строки.

Синтаксис

```
TKPROF имя_файла1 имя_файла2
    [WAITS = YES | NO ]
    [SORT = параметр (, параметр . . .)]
    [PRINT = целое]
    [AGGREGATE = YES | NO]
```

```
[INSERT = имя_файла3]
[SYS = YES | NO]
[[TABLE = схема.таблица] EXPLAIN = имя_пользователя/пароль]
[RECORD = имя_файла4]
[WIDTH = целое]
```

Ключевые слова

имя_файла1

Входной файл трассировки SQL.

имя_файла2

Файл для вывода TKPROF.

WAITS

Указывает, следует ли записывать итоговую информацию о событиях ожидания. Появилась в Oracle9*i*.

SORT

Сортирует команды SQL по убыванию перечисленных параметров:

PRSCNT – количество раз, которое был проведен синтаксический анализ

PRSCPU – время процессора, потраченное на синтаксический анализ

PRSELA – время, потраченное на синтаксический анализ

PRSDSK – количество считываний диска при синтаксическом анализе

PRSQRY – количество считываний блоков в согласованном режиме при синтаксическом анализе

PRSCU – количество считываний блоков в текущем режиме при синтаксическом анализе

PRSMIS – количество промахов для библиотечных кэшей при синтаксическом анализе

EXECNT – количество раз, которое команда была выполнена

EXECPU – время процессора, потраченное в ходе выполнения

EXEELA – время, потраченное в процессе выполнения

EXEDSK – количество считываний диска в процессе выполнения

EXEQRY – количество считываний блоков в согласованном режиме в процессе выполнения

EXECU – количество считываний блоков в текущем режиме в процессе выполнения

EXEROW – количество строк, обработанных в процессе выполнения

EXESMIS – количество промахов для библиотечных кэшей в процессе выполнения

FCHCNT – количество извлечений

FCHCPU – время процессора, потраченное на извлечение

FCHELA – время, потраченное на извлечение

FCHDSK – количество считываний диска при извлечении

FCHQRY – количество считываний блоков в согласованном режиме при извлечении

FCHCU – количество считываний блоков в текущем режиме при извлечении

FCHROW – количество извлеченных строк

Считывание блоков в согласованном режиме выполняется для команд SELECT и не блокирует данные. Считывание блоков в текущем режиме выполняется при записи в базу данных и вызывает блокировку данных.

PRINT

Выводит только указанное в параметре количество упорядоченных команд SQL.

AGGREGATE

Будучи установлен в NO, не суммирует статистику для нескольких пользователей, работающих с одним текстом SQL.

INSERT

Создает SQL-сценарий с именем *имя_файла3*, запускаемый для сохранения статистики файла трассировки в базе данных. По умолчанию данные сохраняются в таблице TKPROF_TABLE, но это значение можно изменить, отредактировав команду CREATE TABLE в файле *имя_файла3*.

SYS

Будучи установлен в NO, не выводит команды SQL, выданные пользователем SYS, или рекурсивные команды SQL.

TABLE

Указывает временную таблицу *схема.таблица*, в которой будет сохранен план выполнения перед его записью в выходной файл, что позволит нескольким пользователям запустить TKPROF с одним и тем же значением параметра *имя_пользователя*.

EXPLAIN

Записывает план выполнения в выходной файл, выдавая команду EXPLAIN PLAN от имени пользователя *имя_пользователя* с паролем *пароль*. Замедляет работу TKPROF.

RECORD

Создает сценарий SQL со всеми нерекурсивными операциями SQL в файле трассировки, который может использоваться для повторного выполнения пользовательских событий.

WIDTH

Определяет ширину строки вывода TKPROF. Появилась в Oracle9*i*.

Запуск TKPROF без каких-либо параметров приводит к выводу справочной информации.

Вывод TKPROF

Вывод утилиты TKPROF включает в себя следующие элементы:

- Команда SQL.
- Статистика для команды. Для каждой команды SQL выдается три строки статистики: синтаксический анализ, выполнение и извлечение (значения Parse, Execute и Fetch в столбце call). В каждой из этих строк существует по семь дополнительных полей в следующих столбцах:

  count

  Счетчик количества выполнений данного этапа.

cpu

Общее время процессора для этапа.

elapsed

Время, затраченное на этап.

disk

Количество считываний блоков диска.

query

Количество буферов, к которым происходило обращение в согласованном режиме, используемом операциями SELECT и не применяющем блокировки.

current

Количество буферов, к которым происходило обращение в текущем режиме, используемом операциями INSERT, UPDATE и DELETE и применяющем блокировки.

rows

Количество строк, обработанных командой SQL.

За этими тремя строками следует итоговая строка с суммарными данными по всем трем фазам:

- Информация о выполнении команды. Она может включать в себя количество промахов для библиотечных кэшей в каждой части вызова, цель оптимизации команды и идентификатор пользователя, выполняющего команду. Если промахов для библиотечных кэшей не наблюдалось, то такие сведения не выводятся.
- План выполнения, если он запрошен посредством аргумента командной строки.

Отчет TKPROF начинается с информации о командах SQL, сгенерировавших ошибки, и завершается общей итоговой информацией, разбитой на две части: для рекурсивных и нерекурсивных команд SQL. Итоговая информация также содержит сведения о трассировочных файлах, использованных при подготовке отчета.

AUTOTRACE

Появившаяся в Oracle8*i* утилита AUTOTRACE обеспечивает автоматический сбор отчетности для команд, выполняемых в SQL*Plus. Для того чтобы можно было применять AUTOTRACE, должна существовать таблица PLAN_TABLE (для хранения информации EXPLAIN PLAN), а пользователь должен обладать ролью PLUSTRACE. Роль PLUSTRACE создается запуском сценария *PLUSTRACE.SQL* от имени DBA; затем она может быть выдана другим пользователям.

Операциями AUTOTRACE можно управлять при помощи команды

```
SET AUTOTRACE ON | OFF | TRACEONLY параметр
```

где *параметр* может иметь значение EXPLAIN или STATISTICS. Если *параметр* не задан, то AUTOTRACE возвращает информацию об оптимизации (EXPLAIN) и статистику, использованную командой. Если указать TRACEONLY, то команда возвращает только план оптимизации и статистику.

AUTOTRACE создает следующую статистику:

recursive calls – количество рекурсивных вызовов

db block –количество запросов блока CURRENT

consistent – количество запросов на согласованное чтение для блока

physical reads – количество блоков данных, прочитанных с диска

redo size – объем журнальных данных, выраженный в байтах

bytes sent via SQL*Net to client – количество байт, отправленных клиенту посредством Oracle Net Services

bytes received via SQL*Net from client – количество байт, полученных от клиента посредством Oracle Net Services

SQL*Net roundtrips to/from client – количество сообщений, отправленных клиенту и полученных от него посредством Oracle Net Services

sorts (memory) – количество сортировок в памяти

sorts (disk) – количество сортировок, потребовавших обращения к диску

rows processed – количество обработанных строк

Сбор статистики

Статистика о работе БД Oracle собирается в динамические представления словаря данных V$ (о них мы говорили в главе 6). Информация о таких динамических представлениях при остановке экземпляра пропадает, поэтому Oracle предоставляет пакеты для сохранения статистики производительности, собранной в этих представлениях.

До выхода версии Oracle8*i* для этого применялись сценарии UTLBSTAT и UTLESTAT. В Oracle8*i* появился пакет Statspack, расширяющий возможности UTLBSTAT и UTLESTAT.

Кроме специальных пакетов, можно применять Oracle Enterprise Manager и его пакет диагностики для мониторинга статистики, относящейся к производительности (см. главу 16).

Сбор статистики – это первый шаг на пути решения проблем производительности. Обучение тонкому искусству настройки производительности выходит за рамки нашего обсуждения, но об этом уже написано множество прекрасных книг.

UTLBSTAT/UTLESTAT

Эти два сценария собирают данные из представлений словаря данных V$ и сохраняют их в таблицах базы данных.

Сценарий UTLBSTAT собирает начальную статистику в набор таблиц, а также создает таблицы для конечной статистики. Сценарий UTLESTAT собирает конечную статистику и создает набор таблиц с отличиями между начальной и конечной статистиками.

Созданные таблицы получают имена согласно приведенному ниже списку. Если не указано иное, то имена таблиц начальной статистики имеют формат STATS$BEGIN_*, имена таблиц конечной статистики – STATS$END_*, а имена таблиц отличий – STATS$*.

**DC*

Для статистики кэша словаря.

**EVENT*

Для статистики ожидания.

**FILE*

Для статистики файлового ввода/вывода. Таблица отличий называется STATS $FILES.

**LATCH*

Для статистики защелок. Таблица отличий называется STATS$LATCHES.

**LIB*

Для статистики библиотечного кэша.

**ROLL*

Для статистики сегментов отката.

**STATS*

Для системной статистики.

STATS$DATES

После запуска UTLBSTAT содержит начальные дату и время, а после запуска UTLESTAT содержит еще и конечные дату и время.

Сценарии UTLBSTAT и UTLESTAT всегда сохраняют статистики в одних и тех же таблицах, поэтому пользователю в каждый момент времени доступен только один набор отличий. Если необходимо видеть несколько наборов отличий, то можно сохранять информацию таблиц отличий в другой набор таблиц или создавать на их основе отчеты для сравнения с другими наборами статистик.

Statspack

Появившийся в Oracle8*i* пакет Statspack выполняет функции, аналогичные сценариям UTLBSTAT/UTLESTAT. Statspack отличается от этих более ранних инструментов тем, что он собирает больший объем данных, автоматически вычисляет некоторые соотношения между данными и сохраняет данные в БД Oracle.

Кроме того, Statspack принимает в расчет как зафиксированные, так и откаченные транзакции.

Установка Statspack

Объекты, с которыми работает Statspack, принадлежат пользователю PERFSTAT. Можно создать такого пользователя, а затем вызвать сценарий SPCREATE.SQL от имени пользователя с привилегией SYSDBA или же выполнить сценарий как пакетное задание в SQL*Plus после установки переменных DEFAULT_TABLESPACE и TEMPORARY_TABLESPACE.

В любом случае сценарий *SPCREATE.SQL* вызывает сценарии *SPCUSR.SQL*, *SPCTAB.SQL* и *SPCPKG.SQL* и сохраняет результаты их работы в соответствующих файлах *.LIS*.

Применение Statspack

Statspack создает моментальную копию представлений статистики производительности на определенный момент времени. Моментальная копия идентифицируется параметрами SNAP_ID, DBID и INSTANCE_ID.

Для создания моментальной копии необходимо войти в SQL*Plus под именем PERFSTAT/PERFSTAT и выполнить команду

```
EXECUTE STATSPACK.SNAP;
```

Процедуру SNAP можно вызвать для получения идентификатора SNAP_ID самой свежей моментальной копии.

Процедуры встроенного пакета DBMS_JOB позволяют создавать моментальные копии по заданному расписанию через указанные интервалы времени.

Отчеты на основе данных Statspack

Для того чтобы получить отчет о различиях двух моментальных копий, применяется сценарий *SPREPORT.SQL* в SQL*Plus (для этого необходимо зарегистрироваться как PERFSTAT). Сценарий перечисляет существующие моментальные копии текущего экземпляра и запрашивает у пользователя начальную и конечную копии, а также имя файла, в котором будет сохранен полученный отчет.

Если между моментальными копиями в списке есть пустая строка, это означает, что в период между созданием этих двух копий экземпляр был перезапущен, поэтому их нельзя использовать для сравнения.

Можно запустить этот сценарий и в режиме пакетной обработки, указав значения для переменных BEGIN_SNAP, END_SNAP и REPORT_NAME.

Вывод, формируемый в отчете из *SPREPORT.SQL*, содержит хеш-значение для идентификации конкретной команды SQL. Подробный отчет о статистике производительности для данной конкретной команды можно получить, запустив отчет *SPREPSQL.SQL*, который запросит у вас все те же самые значения, а также хеш-номер команды. Этот сценарий можно запустить и в режиме пакетной обработки, указав значения для *SPREPORT.SQL* и переменную HASH_VALUE.

Statspack также включает в себя сценарий *SPREPINS.SQL*, который создает отчет по статистике базы данных и экземпляра.

Уровни и пороги Statspack

Пользователь имеет возможность регулировать объем информации, собираемой Statspack, посредством задания уровня или изменения порога для сбора данных. Статистика будет собираться только для тех команд SQL, параметры которых превышают соответствующий порог.

Statspack поддерживает следующие уровни:

- 0 и выше – собирается общая статистика: системная статистика и статистика ожидания, данные о сегментах отката и кэшировании строк, информация SGA; сведения о системных и фоновых событиях, а также о событиях сеанса; статистика для блокировок, пула буферов и родительских защелок.
- 5 и выше – дополнительно собираются данные о производительности для команд SQL, превышающих установленные пороги.
- 6 и выше – дополнительно собираются сведения о возможных изменениях планов выполнения.
- 10 и выше – дополнительно включается информация о дочерних и родительских защелках.

Для команд можно установить следующие пороги (в скобках приведены значения по умолчанию):

I_EXECUTIONS_TH

Количество выполнений (100)

I_DISK_READS_TH

Количество считываний диска (1000)

I_PARSE_CALLS_TH

Количество вызовов синтаксического анализатора (1000)

I_BUFFER_GETS_TH

Количество извлечений из буфера (10000)

I_SHARABLE_MEM_TH

Объем совместно используемой памяти (1048576)

I_VERSION_COUNT_TH

Количество версий (20)

Для того чтобы изменить параметр для отдельного сеанса, включите в процедуру SNAP в качестве параметра идентификатор порога или уровня, за которым следуют символ => и новое значение. Для того чтобы определить уровень, задайте параметр I_SNAP_LEVEL. Можно также добавить параметр I_UCOMMENT для пользовательского комментария и I_SESSION_ID – для идентификатора сеанса, по которому будет собираться специфическая информация.

Для изменения этих параметров на постоянной основе следует присвоить значение TRUE параметру I_MODIFY_PARAMETER при вызове процедуры SNAP или же применить вызов MODIFY_STATSPACK_PARAMETER в том же формате.

Удаление данных Statspack

Для удаления из БД моментальных копий данных, собранных пакетом Statspack, применяется сценарий *SPPURGE.SQL*. Если запустить этот сценарий интерактивно в SQL*Plus, то будет выведен список существующих моментальных копий и система запросит минимальный и максимальный номера копии, которые определят границы диапазона удаления. Также можно задать значения для переменных LOSNAPID и HISNAPID и запустить сценарий в режиме пакетной обработки.

Для того чтобы удалить из базы данных всю моментальную статистику, применяется сценарий *SPTRUNC.SQL*.

IV

Приложения

В этой части книги содержится сводная и справочная информация.

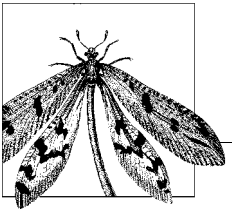
Приложение A «Типы данных» содержит перечень типов данных Oracle и правила их преобразования.

В приложении B «Выражения, операторы и условия» приведен список разрешенных выражений, операторов и условий, которые можно включать в команды SQL, PL/SQL и SQL*Plus.

В приложении C «Числовые форматы» описаны форматы чисел, которые можно использовать в командах SQL, PL/SQL и SQL*Plus.

В приложении D «Форматы даты» описаны форматы дат, допустимые в командах SQL, PL/SQL и SQL*Plus.

Приложение E «Дополнительные ресурсы» содержит перечень книг и сетевых ресурсов, предлагающих дополнительные сведения по вопросам, изучаемым в данной книге.



A

Типы данных

Одним из атрибутов столбца таблицы или переменной в хранимой процедуре является *тип данных*. Тип данных накладывает ограничения на характер хранимой в столбце информации и диапазон допустимых значений, а также на набор допустимых операций.

Типы данных Oracle можно разделить на две категории: скалярные типы (возвращающие единственное значение) и типы-коллекции (содержащие в себе набор однотипных значений или указывающие на него). Кроме того, в языке PL/SQL поддерживается ссылочный тип, содержащий указатели на другие объекты.

Скалярные типы

Имеется пять основных разновидностей скалярных типов: символьные, числовые, дата и время, большие объекты (LOB), прочие. В этом приложении рассмотрены основные свойства каждой из них.

Все скалярные типы данных, описанные ниже, могут применяться как в базе данных, так и в объявлениях переменных PL/SQL, если не оговорено обратное. (Заметьте, однако, что некоторые из типов данных, описанных в разделе «Типы-коллекции», могут применяться только в PL/SQL.)

Символьные типы

Переменные символьного типа могут хранить любые строковые значения, включая и строковые представления числовых данных. Попытка присвоить переменной символьного типа значение, чей размер превышает объявленный размер переменной, приведет к ошибке во время выполнения. Стандартные символьные типы (в отличие от символьных больших объектов, описанных ниже) можно использовать в строковых функциях, таких как UPPER, LOWER, SUBSTR и SOUNDEX.

К символьным относятся следующие типы данных:

CHAR/CHARACTER[*(n* [*CHAR*|*BYTE*])]

Содержит символьные значения фиксированной длины. Значение типа CHAR может иметь длину от 1 до 2000 символов. Если длина не указана явно, то подразумевается 1. Если присваиваемое значение имеет длину меньше, чем указанная, оно будет дополнено пробелами. Начиная с версии Oracle9*i* можно указывать длину *n* с ключевым словом BYTE или CHAR.

VARCHAR2(n [CHAR | BYTE]) и VARCHAR(n [CHAR | BYTE])

Содержит символьные строки переменной длины. Несмотря на то что указание длины обязательно, фактическая длина определяется размером присваиваемого значения. Значения, присваиваемые переменной типа VARCHAR2, не дополняются пробелами. Максимальная длина составляет 4000 символов.

В настоящее время в Oracle8 и более поздних версиях типы VARCHAR и VARCHAR2 идентичны, но корпорация Oracle рекомендует использовать VARCHAR2, так как в будущем между этими типами могут возникнуть различия.

Начиная с Oracle9*i* можно указывать длину *n* с ключевым словом BYTE или CHAR.

В PL/SQL для обозначения этого типа можно использовать ключевое слово STRING.

NCHAR2(n) и NVARCHAR2(n)

Содержат символьные данные фиксированной или переменной длины, представленные в кодировке, отличной от используемой в данной БД. При создании БД указывается, какая кодировка должна применяться для хранения символьных данных. Можно также указать дополнительную кодировку (эта характеристика БД называется NLS – National Language Set). Данная дополнительная кодировка и используется для столбцов типа NCHAR и NVARCHAR2. Например, у вас могут быть поля с описаниями на японском языке, в то время как вся остальная информация в БД хранится в английской кодировке. Для этого при создании БД необходимо в качестве дополнительной указать кодировку, поддерживающую японские символы, а указанным столбцам назначить тип NCHAR или NVARCHAR2.

В Oracle9*i* можно указать, что размер столбцов NCHAR и NVARCHAR2 должен измеряться в символах, а не в байтах. Таким образом, если вы укажете размер столбца равным семи символам, то при условии, что для хранения символа отведено два байта, он будет автоматически преобразован в 14 байт.

LONG

Содержит символьные данные размером до 2 Гбайт. Тип LONG унаследован от ранних версий БД Oracle. Сейчас для хранения больших массивов символьных данных Oracle рекомендует пользоваться типами CLOB и NCLOB. На использование типа LONG в таблицах и запросах накладывается много ограничений, например, нельзя использовать его в инструкциях WHERE, GROUP BY, ORDER BY и CONNECT BY, а также в запросах с квалификатором DISTINCT. Для столбца типа LONG нельзя создать индекс.

Числовые типы

БД Oracle использует для хранения чисел стандартный внутренний формат переменной длины. Этот внутренний формат обеспечивает точность до 38 разрядов.

Единственный числовой тип данных, поддерживаемый Oracle8 и последующими версиями – это NUMBER. (О типах PL/SQL будет рассказано ниже.) Объявление столбца или переменной с типом NUMBER автоматически обеспечивает им точность в 38 разрядов. Объявление типа NUMBER может содержать два квалификатора:

```
column NUMBER(разрядность, масштаб)
```

Разрядность показывает общее количество значащих разрядов в представлении числа и может принимать целые значения вплоть до 38; именно столько принимается по

умолчанию. *Масштаб* представляет собой количество цифр справа от десятичной точки; по умолчанию принимается равным 0. Если *масштаб* отрицателен, Oracle округляет число до указанного разряда слева от десятичной точки.

Для хранения числовых данных в Oracle8 и более поздних версиях используется только тип NUMBER. Все типы данных ANSI DECIMAL/DEC, NUMBER, INTEGER/INT, SMALLINT, FLOAT, DOUBLE PRECISION и REAL представлены в базе данных типом NUMBER. В PL/SQL и других языках все эти типы определены.

В PL/SQL поддерживается тип BINARY_INTEGER, позволяющий хранить целые значения в диапазоне от -2^{31} до 2^{31}. Этот тип данных имеет несколько подтипов:

NATURAL

Неотрицательные целые

NATURALN

Неотрицательные целые и не NULL

POSITIVE

Положительные целые

POSITIVEN

Положительные целые и не NULL

SIGNTYPE

Принимает значения -1, 0 или 1, которые могут быть использованы в троичной логике

Еще один тип, поддерживаемый PL/SQL, – PLS_INTEGER. Этот тип, как и BINARY_INTEGER, может содержать целые значения от -2^{31} до 2^{31}. Его отличие в том, что он требует меньше места для хранения значений и использует машинную арифметику, работающую быстрее, чем арифметическая библиотека BINARY_INTEGER.

Типы даты и времени

Исторически единственным типом данных для хранения дат в Oracle был тип DATE. В Oracle9*i* была добавлена поддержка интервальных типов и типов метки времени.

Тип DATE

Все значения даты и времени Oracle хранит в стандартном внутреннем формате, включающем год, месяц, день, час, минуту и секунду. Для ввода даты по умолчанию используется формат DD-MON-RR, где значение RR из диапазона 50–99 преобразуется в годы 1950–1999, а остальные значения – в годы 2000–2049. Формат ввода даты для экземпляра определяется параметром инициализации NLS_DATE_FORMAT. Для сеанса формат ввода даты устанавливается командой ALTER SESSION. Если требуется изменить формат определенного значения в команде SQL, то следует передать соответствующий параметр функции TO_DATE.

Oracle SQL поддерживает арифметические операции для дат, в которых целая часть представляет дни, а дробная – часы, минуты и секунды. Например, если добавить .5 к значению даты, то результатом будет значение даты и времени, на 12 часов превышающее начальное значение. Приведем несколько примеров арифметических операций с датами:

```
12-DEC-99 + 10 = 22-DEC-99
31-DEC-1999:23:59:59 + .25 = 1-JAN-2000:5:59:59
```

Диапазоны дат и времени

Все описанные в этом приложении типы данных, предназначенные для представления значений даты и времени и интервалов, состоят из следующих полей:

YEAR

Для значений типа DATE может иметь значение в диапазоне от -4712 до 9999 (за исключением года 0). Для интервалов может иметь любое положительное или отрицательное числовое значение.

MONTH

Для значений типа DATE может иметь значение в диапазоне от 01 до 12. Для интервалов может иметь значение в диапазоне от 0 до 11.

DAY

Ограничения зависят от указанного в дате месяца.

HOUR

Для значений типа DATE может иметь значение в диапазоне от 00 до 12 или 23 (в зависимости от установленного для экземпляра формата этого типа данных). Для интервалов может иметь значение в том же диапазоне.

MINUTE

Для значений типа DATE может иметь значение в диапазоне от 00 до 59. Для интервалов может иметь значение в диапазоне от 0 до 59.

SECOND

Для значений типа DATE может иметь значение в диапазоне от 00 до 59,9. Для интервалов может иметь значение в диапазоне от 0 до 59,9.

TIMEZONE_HOUR

Для значений типа DATE может иметь значение в диапазоне от 12 до 13. Для интервалов это поле не используется.

TIMEZONE_MINUTE

Для значений типа DATE может иметь значение в диапазоне от 00 до 59. Для интервалов это поле не используется.

Функция EXTRACT (см. главу 8) позволяет извлечь значение любого из упомянутых полей из значения интервального типа или значения даты и времени.

Типы INTERVAL

Oracle9*i* вводит два новых интервальных типа данных:

INTERVAL YEAR TO MONTH

Хранит период времени, используя поля года и месяца:

```
INTERVAL YEAR (точность_года) TO MONTH
```

где *точность_года* – это количество разрядов, отведенных для задания года (значение по умолчанию равно 2).

INTERVAL DAY TO SECOND

Хранит период времени, используя поля дней, часов, минут и секунд:

```
INTERVAL DAY (точность_дня) TO SECOND (точность_дробной_части_секунд)
```

где *точность_дня* – это количество разрядов для задания числа дней (от 0 до 9, значение по умолчанию равно 2); *точность_дробной_части_секунд* – это количество разрядов в дробной части секунд (от 0 до 9, значение по умолчанию равно 6).

За дополнительной информацией о присваивании значений типам данных INTERVAL обращайтесь к описанию функций INTERVAL в главе 8.

Типы TIMESTAMP

В Oracle9*i* существуют три типа данных TIMESTAMP:

TIMESTAMP

Расширение типа данных DATE; включает в себя значения года, месяца, часов, минут и секунд:

```
TIMESTAMP [(точность_дробной_части_секунд)]
```

где *точность_дробной_части_секунд* – это количество разрядов в дробной части секунд (от 0 до 9, значение по умолчанию равно 2).

TIMESTAMP WITH TIMEZONE

Разновидность типа TIMESTAMP, которая включает в себя смещение часового пояса относительно UTC (Universal Coordinated Time – всеобщее скоординированное время, которое по сути представляет собой время по Гринвичу). Для этого типа данных применяется такой формат:

```
TIMESTAMP [(точность_дробной_части_секунд)] WITH TIMEZONE
```

TIMESTAMP WITH LOCAL TIME ZONE

Аналог TIMESTAMP WITH TIMEZONE, только здесь время нормализовано к локальному часовому поясу БД и смещение относительно UTC не хранится в составе данных столбца. Используется такой синтаксис:

```
TIMESTAMP [(точность_дробной_части_секунд)] WITH LOCAL TIMEZONE
```

Типы больших объектов

Большие объекты (Large Objects – LOBs) обычно служат для хранения больших объемов (до 4 Гбайт) двоичных данных (например, изображений) или символьных текстовых данных. Объекты LOB имеют ряд общих характеристик: их размер обычно превышает разрешенные размеры для других типов данных, или же они имеют форматирование, которое недопустимо для стандартных типов данных.

Oracle поддерживает несколько типов данных для работы с большими объектами:

BLOB

Хранит до 4 Гбайт двоичных данных. BLOB расшифровывается как *binary large object* – большой двоичный объект. Этот тип не преобразует хранимые значения. Может использоваться в транзакции.

CLOB

Хранит до 4 Гбайт символьных данных. CLOB – это *character large object*, или символьный большой объект. Такой тип данных служит для хранения в базе данных большого блока фиксированной длины, содержащего однобайтные символьные данные. Этот тип не преобразует хранимые значения. Может применяться в транзакции.

NCLOB

Хранит до 4 Гбайт символьных данных. NCLOB – это символьный большой объект NLS (National Language Support). Этот тип не преобразует хранимые значения. Может применяться в транзакции.

BFILE

Тип данных BFILE действует как указатель на файл, хранящийся вне БД Oracle. Поэтому столбцы и переменные типа BFILE не участвуют транзакциях, а данные, хранящиеся в таких столбцах, доступны только для чтения. Сервер Oracle воспринимает данные во внешнем файле как двоичные. Объем данных BFILE регулируется ограничениями на размер файла, имеющимися в базовой операционной системе.

Другие скалярные типы данных

Oracle также поддерживает ряд специализированных типов данных:

BOOLEAN

Хранит логическое значение (TRUE, FALSE или NULL).

RAW и LONG RAW

Обычно сервер БД Oracle не только хранит, но и интерпретирует данные. Когда данные запрашиваются или экспортируются из БД, может потребоваться преобразование набора символов или дополнение данных пробелами.

Типы данных RAW и LONG RAW обходят любые попытки интерпретации со стороны БД Oracle. Если указывается один из таких типов, то сервер Oracle сохраняет данные в виде именно той последовательности байтов, которая была ему предъявлена. Типы данных RAW обычно хранят объекты в своем собственном внутреннем формате, как битовые образы. Тип RAW может хранить 2000 байт, а тип LONG RAW – 2 Гбайт.

ROWID

ROWID – это специальный тип столбца, называемый *псевдостолбцом* (*pseudocolumn*). К псевдостолбцу ROWID можно обращаться так же, как и к обычному столбцу, в SQL-команде SELECT. Псевдостолбец ROWID существует для каждой строки БД Oracle и представляет собой ее специальный адрес. Псевдостолбец ROWID можно определить при помощи типа данных ROWID.

ROWID устанавливает связь с конкретным местоположением на дисковом устройстве, следовательно, это самый быстрый способ извлечения отдельной строки. Однако ROWID для строки может измениться в результате записи дампа БД и повторной загрузки. Поэтому мы не рекомендуем использовать значение для псевдостолбца ROWID за пределами одной транзакции.

Задать значение стандартного псевдостолбца ROWID при помощи какой-либо команды SQL невозможно. Можно определить столбец или переменную типа ROWID, но Oracle8 и последующие версии не гарантируют, что любое значение, помещенное в такие переменные или столбцы, будет корректным значением ROWID.

UROWID

Некоторые типы таблиц, такие как индекс-таблицы или внешние таблицы (в базах данных другого типа), имеют идентификаторы строк, которые не являются физическими, постоянными или не генерируются сервером Oracle. Тип UROWID хранит логический, физический или не управляемый Oracle идентификатор стро-

ки для таких таблиц. Псевдостолбец ROWID этих таблиц имеет тип данных UROWID. Тип данных UROWID может хранить до 4000 байт.

XMLType

В рамках поддержки XML версия Oracle9*i* предлагает новый тип данных XMLType. Столбец, определенный с таким типом, может хранить XML-документ. Соответствующие встроенные функции позволяют извлечь отдельные узлы из типа данных (см. описание функций XML в главе 8). Кроме того, можно создать индексы для любого отдельного узла документа XMLType.

SYS.AnyType, SYS.AnyData, SYS.AnyDataSet

Oracle9*i* включает в себя три новых типа данных Any, обеспечивающие возможность явного определения и способные хранить данные любого типа:

SYS.AnyType

Может содержать описание любого типа.

SYS.AnyData

Может содержать любые данные и описание их типа.

SYS.AnyDataSet

Может содержать любой набор данных и описание их типа.

Каждый экземпляр таких типов должен быть определен посредством программных единиц, которые укажут серверу Oracle9*i*, как следует обрабатывать конкретные реализации каждого из типов.

UriType

Oracle9*i* также вводит несколько типов URI, которые используются для хранения универсальных идентификаторов ресурсов. Все эти типы позволяют извлекать информацию с использованием некоторого типа URI. Доступны четыре таких типа данных:

URIType

Родительский объектный тип для остальных трех типов, который может применяться для хранения данных любого другого типа URI.

HTTPURIType

Применяется для хранения идентификаторов URI, указывающих на информацию вне БД. Доступ к таким страницам производится по протоколу HTTP.

XDBURIType

Применяется для хранения иерархий XML БД.

DBURIType

Служит для хранения указателя DBURI REF, который ссылается на данные БД через Xpath-представление.

Пользовательские данные

Oracle предоставляет пользователям возможность создавать собственные сложные типы данных – комбинации описанных выше стандартных типов Oracle. Oracle разрешает создание объектов, состоящих одновременно из элементов как стандартных, так и пользовательских типов данных (дополнительные сведения о создании пользовательских типов данных можно найти в разделе, описывающем SQL-команду CREATE TYPE в главе 7).

Типы-коллекции

В дополнение к рассмотренных скалярным типам данных приведем ряд типов данных, способных хранить коллекцию данных или указывать на такую коллекцию:

RECORD

Применяется только в PL/SQL для определения записей. Состоит из полей некоторых типов данных, аналогично таблице в базе данных. Можно вставлять записи в соответствующие таблицы или обновлять их там, а также применять инструкцию RETURNING для извлечения значений из таблицы после выполнения команды INSERT, UPDATE или DELETE.

TABLE

Вложенная таблица. Не может иметь тип REF CURSOR в PL/SQL или типы данных PL/SQL, такие как BINARY_INTEGER или POSITIVE, когда хранится в базе данных. Отдельные значения в TABLE можно удалять. При сохранении в БД значения сохраняют свой первоначальный порядок.

В PL/SQL можно применять инструкцию INDEX BY, добавляющую индекс, который позволяет организовать доступ к данным таблицы. В инструкции INDEX BY можно указывать спецификатор BINARY_INTEGER, PLS_INTEGER, VARCHAR2(*n*) или тип ключа.

Таблица TABLE может входить в состав объектного типа (объектные типы подробно описаны в главе 7).

VARRAY(*n*)

Массив значений из *n* элементов. К отдельным элементам можно обращаться по индексу. Удалять отдельные значения внутри VARRAY нельзя, поскольку массив должен быть плотным. При сохранении в БД значения не сохраняют первоначальный порядок.

Операторы коллекции

Для коллекций в PL/SQL можно применять следующие операторы:

COUNT

Возвращает количество элементов.

DELETE[([*m*,]*n*)]

Удаляет все элементы (если параметры не указаны), элемент *n* (если указан только *n*) или же элементы между *m* и *n* (если заданы оба параметра).

EXISTS(*n*)

Возвращает TRUE в случае существования элемента с индексом *n*.

EXTEND

Увеличивает размер коллекции.

FIRST

Возвращает первый элемент.

LAST

Возвращает последний элемент.

LIMIT

Для VARRAY возвращает предельно допустимый размер.

NEXT

Возвращает следующий элемент.

PRIOR

Возвращает предыдущий элемент.

TRIM[(*n*)]

Удаляет *n* элементов из конца коллекции. Если *n* не указано, то удаляет один элемент.

За подробной информацией о работе с коллекциями в PL/SQL обратитесь к соответствующим разделам в главе 9. Дополнительные сведения о создании типов-коллекций можно найти в разделе, описывающем SQL-команду CREATE TYPE в главе 7.

Ссылочные типы данных

Эта категория типов данных содержит ссылки на другие наборы данных:

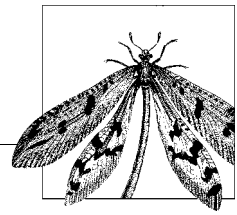
REF CURSOR

В PL/SQL служит указателем на курсор SQL, который может возвратить строку базы данных. После открытия (OPEN) экземпляра этого типа вы можете извлекать с его помощью строки (FETCH). После завершения работы с типом REF CURSOR его необходимо закрыть (CLOSE).

REF объект

Указывает на экземпляр объекта. Служит для организации совместного доступа к объекту для подпрограмм PL/SQL или ссылочных объектов в SQL.

B



Выражения, операторы и условия

Язык SQL (Structured Query Language), которому посвящена глава 7, – это общепринятый язык доступа к базам данных Oracle. Вам довольно часто придется использовать определенную логику для указания того, какие данные должна затрагивать команда SQL. Для реализации этой логики потребуются выражения, операторы и условия.

Выражения

Применяются для определения значений, например:

```
2+2
SYSDATE
TO_CHAR(SYSDATE)
```

Операторы

Применяются для объединения значений, например:

```
a = b
a != b
a AND b
```

Условия

Применяются для указания допустимых значений, например:

```
a > b
a EXISTS
a IS NOT NULL
```

Выражения, операторы и условия можно включать во многие команды SQL, а также в код PL/SQL. В данном приложении собраны разнообразные правила задания выражений, операторов и условий (подробное описание возможностей и применения команд SQL вы найдете в главе 7; в главе 9 описаны выражения PL/SQL).

Выражения

Выражение – это основная единица информации в команде и функции SQL. Oracle поддерживает различные типы выражений:

Простое выражение

Может быть представлено следующим образом:

- Текст в кавычках

- Число
- ROWID для строки, заданной как часть таблицы, таблицы схемы или запроса
- Значение столбца для строки, заданной как часть таблицы, таблицы схемы или запроса
- ROWNUM
- Последовательность, за которой следует .NEXTVAL или .CURRVAL
- NULL

Составное выражение

Состоит из нескольких выражений, соединенных стандартным оператором, таким как +, -, *, / или || (если это возможно). Составное выражение также может быть результатом функции.

CASE выражение

Возвращает единственное значение по результатам логического сравнения. Выражение CASE может иметь две формы:

```
CASE выражение WHEN выражение_сравнения THEN возвращаемое_выражение
    [WHEN выражение_сравнения THEN возвращаемое_выражение . . .]
END
```

или

```
CASE WHEN условие THEN возвращаемое_выражение
    [WHEN условие THEN возвращаемое_выражение . . .]
END
```

Выражения CASE появились в Oracle9*i*. Любая из форм может включать в себя инструкцию ELSE:

```
ELSE else_выражение
```

Эту инструкцию следует поместить перед ключевым словом END; *else_выражение* будет возвращено в том случае, если ни одно из условий или сравнений не возвратит значения.

CURSOR выражение

Возвращает вложенный курсор – эквивалент REF CURSOR в PL/SQL (см. главу 9). Выражение имеет такой формат:

```
CURSOR(подзапрос)
```

Выражение CURSOR появились в Oracle9*i*.

DATETIME выражение

Возвращает значение даты и времени в следующем виде:

```
выражение_даты-и-времени [ AT [LOCAL | TIME ZONE [DBTIMEZONE | SESSIONTIMEZONE
                           | '+/-HH:MM' | 'часовой_пояс' | выражение]]]
```

где '*часовой пояс*' – это имя одного из часовых поясов, перечисленных в табл. 8.2. Выражение DATETIME появились в Oracle9*i*.

Выражение-функция

Функция, которая возвращает значение соответствующего типа данных. Допускает вложенные функции.

INTERVAL выражение

Возвращает интервал времени. Задается в следующем формате:

```
интервальное_выражение [YEAR TO MONTH | DAY TO SECOND]
```

Дополнительную информацию об интервалах можно найти в приложении A. Выражения INTERVAL появились в Oracle9*i*.

Выражение доступа к объекту

Возвращает значение. Можно использовать такую конструкцию:

```
{псевдоним_таблицы.столбец | псевдоним_объектной_таблицы|(выражение) }
{атрибут[.метод ([аргумент[, аргумент . . .]]) |
метод ([.аргумент])}}
```

Скалярное выражение подзапроса

Значение, возвращаемое подзапросом, извлекающим одну или ноль строк. Если возвращена одна строка, то значение представляет собой список выборки подзапроса. Если не возвращено ни одной строки, то значение содержит NULL. Скалярные выражения подзапроса появились в Oracle9*i*.

Выражение конструктора типа

Возвращает объектный тип из метода-конструктора для данного объекта. Применяется такой формат:

```
[NEW] [схема.]имя_типа([выражение[, выражение . . .]])
```

Переменная базового языка

Задает переменную базового языка, перед которой стоит двоеточие (:). За ней может (необязательно) следовать индикаторная переменная базового языка, также предваренная двоеточием. Синтаксис данного выражения таков:

```
: переменная_базового_языка [INDICATOR :индикаторная_переменная_базового_языка]
```

Список выражений

Разделенный запятыми список выражений, который используется для условий сравнения и принадлежности множеству, а также в инструкции GROUP BY запросов.

Операторы

СУБД Oracle поддерживает следующие стандартные операторы:

- Арифметические: + - * /
- Конкатенации: ||
- Операторы над множествами, работающие с группами строк:

  UNION

  Все строки, выбранные любым из запросов.

  UNION ALL

  Все строки, выбранные любым из запросов, включая дубликаты.

  INTERSECT

  Все различные строки, выбранные двумя запросами.

MINUS

Все различные строки, выбранные первым, но не вторым запросом.

- Пользовательские операторы. В Oracle9*i* пользователи получили возможность создавать собственные специальные операторы при помощи команды CREATE OPERATOR.

Операторы вычисляются в соответствии с уровнями приоритетов (приоритеты операторов и условий представлены в табл. B.1). Первыми вычисляются операторы с приоритетом 1. Операторы с одинаковым приоритетом оцениваются слева направо в рамках выражения. Первыми сервер Oracle вычисляет все операторы внутри скобок.

Таблица B.1. Приоритеты операторов и условий

Приоритет	Оператор/условие
1	+ (положительное), - (отрицательное), PRIOR
2	*, /
3	+ (сложение), - (вычитание), ‖ (конкатенация)
4	=, != (не равно), <, >, <=, >=
5	IS [NOT] NULL, [NOT] LIKE, [NOT] BETWEEN, [NOT] IN, EXISTS, IS [NOT] OF *тип*
6	NOT
7	AND
8	OR

Все операторы над множествами и пользовательские операторы имеют одинаковый приоритет и выполняются слева направо после того, как вычислены составляющие их выражения.

Условия

Условие – это логическая операция, возвращающая TRUE, FALSE или UNKNOWN. Условия можно включать в инструкции WHERE таких DML-команд, как SELECT, INSERT, UPDATE или DELETE, а также в инструкции START WITH, CONNECT BY и HAVING команды SELECT.

Простые условия

Простое условие сравнивает выражение с другим выражением или отдельным результатом подзапроса. Oracle поддерживает следующие простые условия:

= Равно
!= Не равно
< Больше
> Меньше
<> Больше или меньше, но не равно
<= Меньше или равно
>= Больше или равно

Групповые условия

Групповое условие сравнивает выражение с группой результатов и имеет такой формат:

```
[()]выражение[, выражение . . .][)] простое_условие ANY | SOME | ALL | IN| NOT IN
   (подзапрос|список_выражений
    [,подзапрос | список_выражений . . .])
```

Если в левой части условия указано несколько выражений, то в правой части должно быть такое же количество значений с соответствующими типами данных.

Если какое-то из значений в правой части условия NOT IN вычисляется как NULL, то условие всегда оценивается как FALSE. Если значения в правой части условия NOT IN представляют подзапрос, не возвращающий ни одной строки, то возвращаются все строки родительского запроса, содержащего условие.

Логические условия

Логическое условие – это условие, которое может возвращать одно из трех значений: TRUE, FALSE или UNKNOWN (последнее в Oracle известно как NULL). Oracle поддерживает три логических условия: NOT, AND и OR.

Условие NOT для TRUE, FALSE и UNKNOWN (или NULL) вычисляется следующим образом:

NOT TRUE = FALSE
NOT FALSE = TRUE
NOT UNKNOWN = UNKNOWN (NULL)

Результаты сравнений для комбинаций логических условий приведены в табл. B.2.

Таблица B.2. Комбинации логических условий

Сравнение	Значение	TRUE	FALSE	UNKNOWN
AND	TRUE	TRUE	FALSE	UNKNOWN
	FALSE	FALSE	FALSE	FALSE
	UNKNOWN	UNKNOWN	FALSE	UNKNOWN
OR	TRUE	TRUE	TRUE	TRUE
	FALSE	TRUE	FALSE	UNKNOWN
	UNKNOWN	TRUE	UNKNOWN	UNKNOWN

Другие условия

Oracle также позволяет указать ряд дополнительных условий:

Условие попадания в диапазон

Возвращаемое значение определяется тем, попадает ли *выражение* в диапазон между двумя указанными значениями. Синтаксис этого условия таков:

```
выражение [NOT] BETWEEN выражение_сравнения1 AND выражение_сравнения2
```

NULL

Возвращаемое значение определяется тем, оценивается ли *выражение* как NULL или же NOT NULL. Задается в следующем формате:

```
выражение IS [NOT] NULL
```

EXISTS

Возвращает TRUE, если подзапрос возвращает хотя бы одну строку. Задается в следующем формате:

```
EXISTS (подзапрос)
```

LIKE

Возвращает TRUE, если *строка1* входит в *строку2*. Поддерживаются групповые символы: символ подчеркивания (_) для представления отдельного символа или знак процента (%) – для нескольких символов. *Экранирующий_символ* должен указываться перед групповыми символами, которые необходимо трактовать буквально (как обычные символы). Условие имеет такой синтаксис:

```
строка1 [NOT] LIKE | LIKEC | LIKE2 | LIKE4 строка2 [ESCAPE экранирующий_символ]
```

Имейте в виду, что при сравнении:

LIKEC соответствует полным символам Unicode.

LIKE2 соответствует кодам UCS2.

LIKE4 соответствует кодам UCS4.

EQUALS_PATH

Определяет, можно ли найти XML-документ в БД по указанному пути. Такое условие появилось в версии Oracle9*i* Release 2 и применяется для представлений RESOURCE_VIEW и PATH_VIEW. Задается в следующем формате:

```
EQUALS_PATH (столбец, строка_пути[, корреляционное_целое])
```

Параметр *корреляционное_целое* служит для связывания данного условия с функциями DEPTH и PATH.

UNDER_PATH

Определяет, можно ли найти XML-документ ниже указанного пути в БД. Такое условие появилось в версии Oracle9*i* Release 2 и применяется для представлений RESOURCE_VIEW и PATH_VIEW. Задается в следующем формате:

```
UNDER_PATH (столбец, [уровни,] строка_пути [,корреляционное_целое])
```

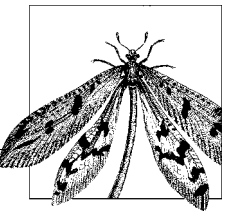
где параметр *уровни* служит для ограничения количества уровней, просматриваемых сервером Oracle вглубь; *корреляционное_целое* предназначен для связывания данного условия с функциями DEPTH и PATH.

IS OF

Проверяет объектные типы на предмет того, совпадают ли они с определенным типом. Это условие появилось в версии Oracle9*i* Release 1 и имеет такой формат:

```
выражение IS [NOT] OF [TYPE] ([ONLY] [схема.]тип[,
   [ONLY] [схема.]тип  ])
```

C



Числовые форматы

В этом приложении приведены элементы формата чисел, которые могут использоваться в разнообразных инструментах Oracle (например, SQL*Plus), во встроенных пакетах и спецификациях форматов внутри команд SQL.

Полный перечень элементов числового формата приведен в табл. C.1. Примеры применения таких элементов предложены в табл. C.2.

Таблица C.1. Элементы числового формата

Элемент формата	Функция
9	Представляет цифру в выводе.
0	Отмечает ту позицию, начиная с которой надо отображать начальные нули.
$	Включает в вывод начальный знак доллара.
,	Помещает в вывод запятую.
.	Отмечает местоположение десятичной точки.
B	Вынуждает отображать нулевые значения как пробелы.
C	Помечает то место, в котором должен появиться ISO-код валюты. Для долларов США это USD.
D	Отмечает местоположение десятичной точки.
MI	Добавляет завершающий знак «минус» к числу и может находиться только в конце строки формата.
S	Добавляет к числу знак + или – и может располагаться как в начале, так и в конце строки формата.[a]
PR	Приводит к отображению отрицательных значений в угловых скобках, например –123,99 будет выведено как <123,99>.
G	Помещает в вывод разделитель групп (обычно это запятая).

[a] SQL*Plus всегда принимает во внимание знак при отображении числа, где бы он ни был указан. По умолчанию знак расположен слева от числа и отображается только для отрицательных чисел. Для положительных чисел в крайней левой позиции выводится пробел.

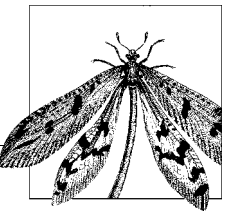
Элемент формата	Функция
L	Помечает то место, в котором должен появиться национальный код валюты. Для долларов США это знак доллара.
V	Выводит масштабированные значения. Количество разрядов справа от V указывает, на сколько позиций справа от десятичной точки происходит смещение, прежде чем число выводится.
EEEE	Вывод значения в экспоненциальном представлении. Необходимо ввести именно четыре E, причем они должны находиться в конце строки формата.
RN или rn	Выводит число римскими цифрами. RN в верхнем регистре приводит к выводу римских цифр в верхнем регистре, rn – к выводу римских цифр в нижнем регистре. Римскими цифрами могут быть представлены целые числа в диапазоне от 1 до 3999 включительно.
DATE	Считает, что число представляет дату юлианского календаря и выводит ее в формате MM/DD/YY.

Таблица С.2. Элементы числового формата

Значение	Формат	Результат
123	9999	123
1234,01	9 999,99	1 234,01
23456	$999 999,99	$23 456,00
1	0999	0001
1	99099	001
-1000,01	9 999,99mi	1 000,01-
1001	S9 999	+1 001
-1001	9 999PR	<1 001>
1001	9 999PR	1 001

D

Форматы дат



В этом приложении приведены элементы формата дат, применяемые в разнообразных инструментах Oracle (например, SQL*Plus), во встроенных пакетах и спецификациях форматов внутри команд SQL.

Полный перечень элементов формата дат приведен в табл. D.1. Примеры применения некоторых из этих элементов предложены в табл. D.2.

Таблица D.1. Элементы формата дат

Элемент формата	Функция
-/,.;:	Знаки пунктуации, включаемые в вывод.
'текст'	Заключенный в кавычки текст для воспроизведения в выводе.
AD или A.D. BC или B.C.	Индикатор AD, A.D., BC, B.C., включаемый в дату.
AM или A.M. PM или P.M.	Вывод в соответствующих случаях AM, A.M., PM или P.M.
CC	Номер века. Года с 1900 по 1999 – это XX век.
SCC	Аналогичен CC, отрицателен для дат до нашей эры
D	Номер дня недели от 1 до 7.
DAY	Полное название дня недели.
DD	День месяца.
DDD	День недели.
DY	Сокращенное название дня недели.
E	Сокращенное название эры. Действует только для календарей Japanese Imperial, ROC Official и Thai Buddha.
EE	Полное название эры (см. также E).
FF	Дробные секунды. Действует только для типов TIMESTAMP.
FM	Подавляет избыточные пробелы и нули в представлении даты символьной строкой. Например, задав формат 'FMMonth DD', можно получить строку 'July 4' вместо 'July 04'.

Элемент формата	Функция
HH	Час дня в 12-часовом формате.
HH12	Час дня в 12-часовом формате.
HH24	Час дня в 24-часовом формате.
IW	Номер недели по стандарту ISO, принимающий значение от 1 до 53 (см. также IYYY).
IYYY	Четырехзначный год по стандарту ISO. Год в терминах ISO начинается 1 января только в том случае, если 1 января выпадает на понедельник. Год начинается в предыдущий понедельник, если 1 января выпадает на вторник, среду или четверг. Если же 1 января – это пятница, суббота или воскресенье, то год начинается в понедельник, следующий за 1 января.
IYY	Последние три цифры номера года ISO.
IY	Последние две цифры номера года ISO.
I	Последняя цифра номера года.
J	День по юлианскому календарю. День с номером 1 – это 1 января 4712 года до нашей эры.
MI	Минута.
MM	Номер месяца.
MON	Трехбуквенное сокращение названия месяца.
MONTH	Полное название месяца.
Q	Квартал года. Первый квартал включает себя месяцы с января по март и т. д.
RM	Номер месяца римскими цифрами.
RR	Двузначный год.
RRRR	Четырехзначный год.
SP	Суффикс, задающий преобразование числа в пропись (например, ONE, FOUR).
SS	Секунда.
SSSSS	Количество секунд после полуночи.
TH	Суффикс, который может располагаться в конце любого элемента, форматирующего вывод числа, и задает вывод порядкового числительного (например, 1st, 4th).
SPTH	Суффикс, преобразующий число в порядковое числительное, записываемое словом (FIRST, FOURTH). См. также TH.
TZD	Сокращенное имя часового пояса (например, EST, PST и т. д.).
TZH	Составляющая часов в смещении часового пояса относительно UTC (например, –05 для EST – восточного времени США).
TZM	Составляющая минут в смещении часового пояса относительно UTC (обычно 0).

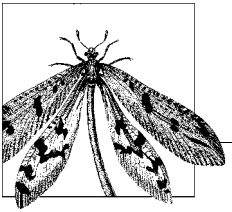
Таблица D.1 (продолжение)

Элемент формата	Функция
TZR	Регион часового пояса (например, US/Eastern).
WW	Неделя года.
W	Неделя месяца. Первая неделя начинается первого числа месяца, вторая неделя начинается 8-го числа месяца и т. д.
X	Национальный символ десятичной дроби, например точка (в Америке и Англии).
Y,YYY	Четырехзначный год с запятой после первого разряда.
YEAR	Год прописью.
SYEAR	Год прописью с начальным знаком «минус» для годов до нашей эры (см. также YEAR).
YYYY	Четырехзначный год.
SYYY	Четырехзначный год с начальным знаком «минус» для годов до нашей эры.
YYY	Последние три цифры номера года.
YY	Последние две цифры номера года.
Y	Последняя цифра номера года.

Если определяется элемент формата даты, отвечающий за вывод текстового значения (например, названия месяца), то регистр, указанный для элемента формата, определяет регистр вывода.

Таблица D.2. Примеры форматов дат

Формат	Результат
dd-mon-yyyy	13-dec-2002
dd-Mon-yyyy	13-dec-2002
DD-MON-YYYY	13-DEC-2002
Month dd,yyyy	December 13, 2002
Mm/dd/yyyy	12/13/2002
Day	Sunday



E

Дополнительные ресурсы

Обо всем, что было так или иначе затронуто в книге, можно узнать еще массу интересного. В этом приложении мы порекомендуем некоторые ресурсы для дальнейшего самостоятельного обучения.

Веб-сайты

Корпорация Oracle

Это интернет-дом компании Oracle. Тут можно найти самые последние новости и предложения, а также некоторую полезную техническую информацию.

http://www.oracle.com/

Oracle Technology Network

Технологическая сеть Oracle (Oracle Technology Network – OTN, или TechNet) полна ресурсов, относящихся ко всем аспектам технологии Oracle. Это отличное место для тех, кому нужно скачать программное обеспечение или документацию Oracle; доступно и множество примеров кода. Заходите и ищите то, что вам нужно.

http://technet.us.oracle.com/

Oracle FAQ

Этот никак не связанный с корпорацией Oracle сайт содержит множество полезной информации по Oracle, включая ответы на часто задаваемые вопросы, сценарии, советы, новости, вакансии, конференции, чаты и многое другое.

http://www.orafaq.com

Oraclezone

Сайт предлагает обширные сведения об Oracle, в том числе новости, сценарии, советы и разнообразные хитрые приемы специально для администраторов БД Oracle.

http://www.oraclezone.com

Ixora

Сайт поддерживается Стивом Адамсом (Steve Adams) и предлагает массу сведений о внутреннем устройстве Oracle, а также советы по настройке производительности.

http://www.ixora.com.au

OraPub

Сайт поддерживается Крэйгом Шеллахамером (Craig Shallahamer), долгое время работавшим в группе анализа производительности Oracle, и предоставляет полезную информацию о настройке производительности Oracle.

http://www.orapub.com

Quest Pipelines

Это сайт сообщества разработчиков и администраторов Oracle, поддерживаемый компанией Quest Software. Разделы, посвященные PL/SQL и администрированию, регулярно просматриваются многими авторами Oracle и другими специалистами. Оба канала предлагают архивы учебных материалов и полезного кода. На конференциях, где разработчики и администраторы имеют возможность получения бесплатных консультаций, возникает множество бурных дискуссий.

http://www.quest-pipelines.com

PL/Net.org

Этот сайт, поддерживаемый Биллом Прибылом (Bill Pribyl), представляет собой хранилище открытых программных средств, написанных на PL/SQL, и предназначен в основном для PL/SQL-разработчиков. На нем представлен список часто задаваемых вопросов с ответами, а также ссылки на разнообразное полезное программное обеспечение, в том числе utPLSQL (поблочное тестирование для разработчиков на PL/SQL).

http://plnet.org

Журнал «Oracle Professional», выпускаемый издательством Pinnacle

Сайт содержит сведения о ежемесячном печатном информационном бюллетене для пользователей (существующем наряду с онлайновой версией), который предлагает исчерпывающие данные о технологии Oracle.

http://www.oracleprofessionalnewsletter.com/

Журнал «Oracle Magazine» корпорации Oracle

Этот сайт содержит информацию о ежемесячном печатном издании, охватывающем различные аспекты технологии Oracle и нередко рассказывающем о самых последних новшествах.

http://www.oramag.com/

Издательство O'Reilly

Посвященная Oracle область веб-сайта компании O'Reilly содержит подробные описания книг по Oracle, статьи, описывающие технологии Oracle, а также ссылки на другие полезные онлайн-ресурсы. На главной странице сайта вниманию посетителей предложены ссылки на множество книг разнообразной тематики. Не забудем и о сети O'Reilly Network, которая обеспечивает доступ к статьям, конференциям и другим ресурсам, относящимся к открытым и новым технологиям:

http://oracle.oreilly.com
http://www.oreilly.com
http://oreillynet.com

Книги

Дополнительную информацию по любой из тем, рассмотренных в нашей книге, можно найти в разнообразных печатных изданиях.

В первую очередь упомянем документацию Oracle. Стандартная документация Oracle была значительно усовершенствована за прошедшие годы. В настоящее время она распространяется только через Интернет и на компакт-дисках и представляет собой великолепный источник информации.

Общая информация

Книги этого раздела содержат общие сведения о продуктах и возможностях Oracle. Издания, относящиеся к конкретным предметным областям, приведены в последующих разделах.

Greenwald, Rick «Oracle Essentials: Oracle9*i*, Oracle8*i*, and Oracle8», O'Reilly & Associates, Inc.

Kreines, David C., and Brian Laskey «Oracle Database Administration: The Essential Reference», O'Reilly & Associates, Inc.

Kyte, Thomas «Expert One-on-One», Oracle Wrox Press.[1]

Lewis, Jonathan «Practical Oracle8*i*: Building Efficient Databases», Addison Wesley.

Loney, Kevin, and George Koch «Oracle9*i*: The Complete Reference», Osborne-McGraw Hill.

Thakkar, Meghraj «e-Business for the Oracle DBA», Sams Publishing.

Loney, Kevin and Marlene Theriault «Oracle9*i* DBA Handbook», Osborne-McGraw Hill.[2]

Theriault, Marlene, et al. «Oracle DBA 101», Osborne-McGraw Hill.[3]

Безопасность

Theriault, Marlene, and Aaron Newman «Oracle Security Handbook», Osborne-McGraw Hill.[4]

Работа в сети

Gennick, Jonathan, and Hugo Toledo «Oracle Net8 Configuration and Troubleshooting», O'Reilly & Associates.

Theriault, Marlene «Oracle Networking 101», Osborne-McGraw Hill.[5]

SQL

Kreines, David C. «Oracle SQL: The Essential Reference», O'Reilly & Associates, Inc.

Mishra, Sanjay, and Alan Beaulieu «Mastering Oracle SQL», O'Reilly & Associates, Inc.[6]

---

1 Кайт Т. «Oracle для профессионалов. Книга 1. Архитектура и основные особенности», Диасофт, 2002.

Кайт Т. «Oracle для профессионалов. Книга 2. Расширение возможностей и защита», Диасофт, 2003.

2 Луни К., Терьо М. «Oracle9*i*. Настольная книга администратора», Лори, 2004.

3 Терьо М. «101 Oracle. Настольная книга администратора», Лори, 2001.

4 Терьо М., Ньюмен А. «ORACLE. Руководство по безопасности», Лори, 2004.

5 Терьо М. и др. «101 Oracle 8*i*: Организация работы в сети», Лори, 2001.

6 Мишра С., Бьюли А. «Секреты Oracle SQL», Символ-Плюс, 2003.

PL/SQL

Feuerstein, Steven «Oracle PL/SQL Best Practices», O'Reilly & Associates, Inc.

Feuerstein, Steven, with Andrew Odewahn «Oracle PL/SQL Developer's Workbook», O'Reilly & Associates, Inc.

Feuerstein, Steven, with Bill Pribyl «Oracle PL/SQL Programming», O'Reilly & Associates, Inc.[1]

Feuerstein, Steven, Bill Pribyl, and Chip Dawes «Oracle PL/SQL Language Pocket Reference», O'Reilly & Associates, Inc.[2]

Pribyl, Bill, and Steven Feuerstein «Learning Oracle PL/SQL», O'Reilly & Associates, Inc.

Trezzo, Joseph C. «Oracle PL/SQL Tips and Techniques», Osborne-McGraw Hill.

Пакеты

Feuerstein, Steven, John Beresniewicz, and Chip Dawes «Oracle PL/SQL Built-ins Pocket Reference», O'Reilly & Associates, Inc.

Feuerstein, Steven, Charles Dye, and John Beresniewicz «Oracle Built-in Packages», O'Reilly & Associates, Inc.

Java

Bales, Don «Java Programming with Oracle JDBC», O'Reilly & Associates, Inc.

Flanagan, David «Java in a Nutshell», O'Reilly & Associates, Inc.[3]

Flanagan, David, Jim Farley, and William Crawford «Java Enterprise in a Nutshell», O'Reilly & Associates, Inc.

McLaughlin, Brett «Building Java Enterprise Applications», O'Reilly & Associates, Inc.

Morisseau-Leroy, Nirva et al. «Oracle8*i* SQLJ Programming», Osborne-McGraw Hill.[4]

Niemeyer, Pat, and Jonathan Knudsen «Learning Java», O'Reilly & Associates, Inc.

Price, Jason «Programming with Oracle SQLJ», O'Reilly & Associates, Inc.

Reese, George «Database Programming with JDBC and Java», O'Reilly & Associates, Inc.

SQL*Plus

Gennick, Jonathan «Oracle SQL*Plus: The Definitive Guide», O'Reilly & Associates, Inc.

Gennick, Jonathan «Oracle SQL*Plus Pocket Reference», O'Reilly & Associates, Inc.[5]

---

1 Фейерштейн С., Прибыл Б. «Oracle PL/SQL для профессионалов», 3-е издание, СПб.: Питер, 2003.

2 Фейерштейн С., Прибыл Б., Доз Ч. «Oracle PL/SQL. Карманный справочник», 3-е издание, СПб.: Питер, 2004.

3 Флэнаган Д. «Java. Справочник», Символ-Плюсс.

4 Моррисо-Леруа Н. «ORACLE 9*i*: Программирование на SQLJ», Лори, 2003.

5 Генник Дж. «Oracle SQL*Plus. Карманный справочник», 2-е издание, СПб.: Питер, 2004.

SQL*Loader

Gennick, Jonathan, and Sanjay Mishra «Oracle SQL*Loader: The Definitive Guide», O'Reilly & Associates, Inc.

Резервирование и восстановление

Freeman, Robert G., and Hart, Matthew «Oracle9*i* RMAN Backup and Recovery», Osborne-McGraw Hill.

Kuhn, Darl, and Scott Schulze «Oracle RMAN Pocket Reference», O'Reilly & Associates, Inc.

Smith, Kenny, and Stephan Haisley «Oracle Backup and Recovery 101», Osborne-McGraw Hill.

Velpuri, Rama, et al. «Oracle8*i* Backup and Recovery», Osborne-McGraw Hill.

Enterprise Manager

Vanting, Lars Bo, and Dirk Schepanik «Oracle Enterprise Manager 101», Osborne-McGraw Hill.

Производительность

Adams, Steve «Oracle8*i* Internal Services», O'Reilly & Associates, Inc.

Alomari, Ahmed «Oracle8*i* and UNIX Performance Tuning», Prentice Hall.

Burleson, Don «Oracle9*i* High-Performance Tuning with STATSPACK», Osborne-McGraw Hill.

Gurry, Mark «Oracle SQL Tuning Pocket Reference», O'Reilly & Associates, Inc.

Gurry, Mark, and Peter Corrigan «Oracle Performance Tuning», O'Reilly & Associates, Inc.

Harrison, Guy «Oracle SQL High-Performance Tuning», Prentice-Hall.

Morle, James «Scaling Oracle8*i*: Building Highly Scalable OLTP System Architectures», Addison Wesley.

Niemiec, Rich, et al. «Oracle Performance Tuning Tips and Techniques», (Osborne-McGraw Hill.

Vaidyanatha, Gaja Krishna, Kirtikumar Deshpande, and John A. Kostelac «Oracle Performance Tuning 101», Osborne-McGraw Hill.[1]

[1] Вайдьянатха Г. К. «Oracle 101. Настройка производительности», Лори, 2003.

Алфавитный указатель

Специальные символы

Числа

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

А

Б

В

Г

Д

Ж

З

И

K

Л

М

Н

О

П

P

C

Т

У

Ф

Х

Ч

Ш

Э

Я