# Лекция 1. Введение в Oracle

## Семейство продуктов Oracle Database

Oracle Database llg - последний представитель продуктов, составляющих семейство реляционных систем управления базой данных (РСУБД) Oracle, построенных на основе единых исходных текстов. В это семейство входят:

*Oracle Enterprise Edition*

Флагманский продукт и основная тема настоящей книги. Ориентирован на крупномасштабные проекты, нуждающиеся в полном наборе средств Oracle. Только Enterprise Edition поддерживает такие развитые механизмы обеспечения безопасности, как виртуальная частная база данных (Virtual Private Database, VPD), детальный аудит (Fine-Grained Auditing) и другие опции, включая Database Vault, Advanced Security и Label Security. Лишь в Enterprise Edition хранилища данных поддерживают сжатие повторяющихся значений, кросс-платформенные переносимые табличные пространства, управление жизненным циклом информации (Information Lifecycle Management, ILM), перезапись запросов с материализованными представлениями, а также секционирование (Partitioning), OLAP и добычу данных (Data Mining). К числу механизмов обеспечения высокой доступности, включенных в Enterprise Edition, относятся Data Guard, ретроспективные (flashback) базы, ретроспективные таблицы и ретроспективные транзакции. В Oracle Database llg добавлена опция сжатия Advanced Compression Option для любой рабочей нагрузки, в том числе для обработки транзакций, хранения больших объектов (Large Object, LOB) и резервных копий; подсистема тестирования базы данных, которая называется Real Application Testing Option и включает в себя программы Database Replay и SQL Performance Analyzer, а также опция Total Recall Option, обеспечивающая режим архивации ретроспективных данных Flashback Data Archive, который сохраняет данные, необходимые для выполнения хронологических запросов (запросов с конструкцией AS OF, где задается дата в прошлом).

*Oracle Standard Edition*

Эта СУБД ориентирована на реализацию баз данных малого и среднего размера. Ее можно развернуть в серверной конфигурации, имеющей до 4 ЦП, на одном компьютере или на кластере с использованием подсистемы Real Application Clusters (RAC).

*Oracle Standard Edition One*

Ориентированная на небольшие проекты, эта СУБД поддерживает до двух ЦП и не поддерживает RAC. В остальном набор возможностей схож с реализованным в редакции Oracle Standard Edition.

*Oracle Personal Edition*

СУБД, используемая разработчиками-одиночками для создания кода, который будет выполняться в многопользовательской СУБД.

В отличие от Express Edition, требует лицензии, но обладает всей функциональностью Enterprise Edition.

*Oracle Express Edition*

СУБД начального уровня, доступная для Windows и Linux бесплатно. Может использовать не более 1 Гбайт памяти и 4 Гбайт дискового пространства. Предоставляет часть функциональности, включенной в редакцию Standard Edition One. Отсутствуют такие функции, как виртуальная Java-машина, управляемое сервером резервное копирование и восстановление, а также подсистема Automatic Storage Management. Oracle Enterprise Manager не умеет управлять этой СУБД, однако ее можно развернуть так, что она будет доступна из административного интерфейса Oracle Application Express (бывший HTML-DB), позволяющего управлять несколькими пользователями.

Обычно Oracle выпускает новые версии своей флагманской СУБД каждые три-четыре года. Новые версии, как правило, посвящены какой-то одной теме и включают целый ряд новых функций. В последних версиях тема обозначалась в названии версии продукта. Так, в 1998 году Oracle анонсировала версию Oracle8i, где буква *i* обозначала поддержку развертывания для работы в Интернете. Версия Oracle9i продолжила эту тему. В 2003 году вышла версия Oracle Database 10g, где *g* означает сконцентрированность на моделях развертывания с поддержкой grid-вычислений. Oracle продолжает эту тему и в текущей версии СУБД, которая рассматривается в настоящей книге. Между основными версиями Oracle выпускает промежуточные. В них тоже добавляются новые возможности, но основное внимание все же уделено совершенствованию уже реализованных средств.

Термины «Oracle», «Orac1e8», «Orac1e8£», «Orac1e9£», «OracleDataba- se lOg» и «Oracle Database

llg» в этой книге не всегда употребляются строго, поскольку Oracle Database llg включает все функции из предшествующих версий. Описывая функцию, которая впервые была включена в некую конкретную версию, во избежание путаницы мы старались отмечать этот факт, понимая, что многие читатели работают не с самой свежей версией Oracle. При описании функций, общих для всех этих версий, мы говорим просто «Oracle».

С 1983 года подразделение Oracle Development ведет разработку на основе модели единого набора исходных текстов для всего семейства продуктов, связанных с базами данных. Хотя в реализации каждой СУБД на самых нижних уровнях встречается системно-зависимый код, необходимый для лучшего учета особенностей конкретной платформы, интерфейсы, раскрываемые пользователям, разработчикам и администраторам, одинаковы. Поскольку поведение функций не зависит от платформы, любая организация может безболезненно переносить СУБД Oracle и приложения для них с одной аппаратной платформы или операционной системы на другую. Такая стратегия позволяет Oracle реализовывать новые функции только один раз для каждого набора продуктов.

## Сводка функций СУБД Oracle

СУБД Oracle - очень крупный продукт. Чтобы создать первичное представление о нем, мы начнем с высокоуровневого обзора основной функциональности.
Чтобы как-то структурировать широкий спектр возможностей СУБД Oracle, мы выделили следующие аспекты:

- средства разработки приложений базы данных;
- средства установления соединения с базой данных;
- распределенные базы данных;
- средства перемещения данных;
- средства повышения производительности;
- средства управления базой данных;
- средства обеспечения безопасности базы данных.

## Средства разработки приложений баз данных

СУБД Oracle обычно используется для хранения данных, которые извлекаются приложениями. Описанные в этом разделе средства и соответствующие продукты применяются для создания таких приложений. Мы решили отдельно рассмотреть программирование баз данных и возможности их расширения.

## Программирование баз данных

Во все варианты СУБД Oracle включены языки и интерфейсы, позволяющие программистам извлекать данные из базы и манипулировать ими. Средства программирования баз данных обычно интересуют разработчиков, которые создают коммерческие приложения на базе Oracle, а также ИТ-отделы, создающие приложения для нужд собственных организаций. Для доступа к данным в Oracle можно использовать SQL, ODBC, JDBC, SQLJ, OLE DB, ODP.NET, SQL/XML, XQuery и WebDAV. Программы, хранящиеся в самой базе данных, могут быть написаны на языках PL/SQL и Java.

### SQL

Описываемый стандартом ANSI язык Structured Query Language (SQL) включает базовые средства манипулирования данными, управления транзакциями и извлечения записей из базы данных. Бизнес-пользо- ватели по большей части взаимодействуют с Oracle посредством приложений или инструментов бизнес-анализа, которые предоставляют интерфейсы, скрывающие SQL и присущую ему сложность.

### PL/SQL

PL/SQL- это разработанное Oracle процедурное расширение языка SQL. Обычно на нем реализуются логические программные модули для приложений. На языке PL/SQL можно писать хранимые процедуры, триггеры, циклы, условные предложения и обработку ошибок. Процедуры на PL/SQL можно откомпилировать и сохранить в базе данных. Блоки, написанные на PL/SQL, можно также исполнять непосредственно с помощью интерактивного инструмента SQL*Plus, имеющегося во всех версиях Oracle. Программные блоки на PL/SQL можно скомпилировать заранее.

### Java

В Oracle8i язык Java впервые начал использоваться для написания хранимых процедур, а виртуальная Java-машина (JVM) была встроена непосредственно в СУБД (первоначальное

название JServer). JVM обеспечивает поддержку написания на Java хранимых процедур, методов и триггеров, а также технологий Enterprise JavaBeans™ (EJB), CORBA, ПОР и HTTP.

Включение Java в СУБД Oracle позволяет программистам, владеющим Java, применить свои знания к разработке приложений для Oracle. Ja- va-приложения можно развертывать на стороне клиента, внутри сервера приложений или в базе данных - в зависимости от конкретных обстоятельств. Oracle Database llg включает JIT-компилятор Java, по умолчанию активированный.

### Oracle и веб-службы

Начиная с версии Oracle Database llg, СУБД может служить поставщиком веб-служб, реализованных в базе данных с помощью технологии XML DB. Веб-службы позволяют создавать запросы на языках SQL или XQuery и получать результаты в формате XML либо вызывать PL/SQL-функции или функции в составе пакета и получать их результаты. Реализация XQuery в Oracle Databasellg поддерживает пока еще обсуждаемый стандарт JSR-225 и включает ряд мер, повышающих производительность.

### Большие объекты

Интерес к применению больших объектов (LOB) постоянно растет, особенно в контексте хранения таких нетрадиционных типов данных, как изображения. В базе данных Oracle уже довольно давно можно было хранить большие объекты. В Oracle8 появилась возможность иметь в одной таблице несколько LOB-столбцов. В Oracle Database 10g по существу было снято ограничение на размеры больших объектов. В Oracle Database llg внедрена технология SecureFiles, что заметно повысило производительность операций выборки и вставки больших объектов. Для LOB-данных с применением SecureFiles поддерживается прозрачное шифрование.

### Объектно-ориентированное программирование

Инфраструктура объектов для поддержки объектно-ориентированного подхода в программировании существовала со времен Oracle8L Например, программист мог создать определяемый пользователем тип данных, содержащий методы и атрибуты. Поддержка объектов в Oracle включает механизм Object Views, с помощью которого объектно-ориентированные программы могут работать с уже хранящимися в базе реляционными данными. Хранить объекты в базе данных можно в виде массивов переменной длины (VARRAY), вложенных таблиц или индекс-таблиц (index organized tables, ЮТ).

### Языки третьего поколения (3GL)

Программисты могут обращаться к базе данных Oracle из программ, написанных на языках C, C++, Java или COBOL, встраивая в них команды SQL. Перед тем как подавать такое приложение на вход платформенного компилятора, его необходимо пропустить через прекомпилятор. Последний заменяет команды SQL вызовами библиотечных функций, понятных стандартному компилятору. Oracle поддерживает такую методику с помощью дополнительного прекомпилятора Pro*C для языков C и C++ и прекомпилятора Pro*COBOL для языка COBOL. В последние версии Oracle включен прекомпилятор SQLJ для языка Java, который заменяет команды SQL обращениями к библиотеке времени выполнения SQLJ, также написанной на Java.

### Драйверы базы данных

Во все версии Oracle включены драйверы, позволяющие приложению обращаться к базе данных посредством ODBC (открытый стандарт взаимодействия с базами данных) или JDBC (открытый стандарт взаи-
модействия с базами данных для Java). Имеются также поставщики данных для OLE-DB и .NET.

### Интерфейс уровня вызовов Oracle

Опытный программист, стремящийся добиться максимальной производительности, может определить команду SQL в виде символьной строки объемлющего языка, затем явно разобрать эту команду, привязать к ней переменные и выполнить ее с помощью интерфейса уровня вызовов Oracle (Oracle Call Interface, OCI). Интерфейс OCI гораздо детальнее предыдущих, для работы с ним и последующей отладки программисту придется затратить много времени и усилий. Разработка приложений с помощью OCI может занять много времени, но расширение функциональности и повышение быстродействия оправдают дополнительные затраты. Например, если механизм обеспечения высокой доступности реализован так, что несколько систем разделяют общие диски с помощью подсистемы Real Application Clusters, то OCI дает возможность написать программу, которая позволит пользователю прозрачно присоединиться

ко второму серверу, если первый выйдет из строя.

### Поддержка национальных языков

Подсистема поддержки национальных языков (National Language Support, NLS) предоставляет наборы символов и прочие данные, например форматы записи чисел и дат, для различных языков. В Oracle Database llg добавлена поддержка Unicode5.0. Кодировка Unicode позволяет хранить все данные или постепенно переводить на нее отдельные столбцы. Кодировки UTF-8 и UTF-16 обеспечивают поддержку более 57 языков и 200 наборов символов. Многие вещи локализованы изначально (например, форматы данных), но при желании с помощью утилиты Oracle Locale Builder можно создать нестандартную локаль. Включен также инструментарий Globalization Toolkit для создания приложений, поддерживающих несколько языков.

## Расширяемость базы данных

Работа в Интернете и в корпоративных сетях интранет выдвигает новые требования к хранению данных нетрадиционных типов и манипулированию ими. Если нужно расширить стандартную функциональность базы данных для хранения изображений, аудио, видео, пространственных данных и временных рядов, то эти возможности можно добавить путем расширения стандартного языка SQL.

### Подсистема Oracle Multimedia

Подсистема Oracle Multimedia (бывшая mterMedia) предоставляет средства манипулирования текстом, изображениями, аудио- и видеоинформацией, географическими координатами, а именно:

- часть Multimedia, относящаяся к тексту (Oracle Text), может распознать смысл документа, производя в нем поиск по темам и ключевым фразам;
- часть Multimedia, относящаяся к изображениям, умеет сохранять и извлекать изображения в различных форматах; начиная с версии Oracle Database llg, поддерживается формат DICOM медицинских изображений;
- части Multimedia, относящиеся к аудио- и видеоинформации, способны сохранять и извлекать аудио- и видеоклипы соответственно;
- часть Multimedia, относящаяся к геоинформации, умеет извлекать данные о пространственных координатах.

### Управление контентом в Oracle

К средствам управления контентом относится подсистема Content Database Option, позволяющая сохранять в базе данных документы, а также приложения для управления контентом компании Stellent, приобретенной Oracle в 2007 году: Universal Content Management, Universal Records Management и Information Rights Management.

### Средства поиска в Oracle

В состав продуктов Oracle Database и Application Server входит инструмент поиска Ultra Search. Обычно он применяется для сбора информации о местонахождении различных текстовых данных, хранящихся в корпоративной сети. Выборка документов базируется на правах доступа конкретного пользователя. Кроме того, предлагается альтернативная система Secure Enterprise Search, обладающая большей гибкостью в среде, не основанной целиком на продуктах Oracle.

### Подсистема Oracle Spatial Option

Подсистема Oracle Spatial Option включена только в редакцию Oracle Enterprise Edition. Она позволяет оптимизировать выборку и отображение данных, привязанных к координатам, и применяется при разработке геоинформационных систем (ГИС). Некоторые производители таких систем уже включили ее в свои продукты и применяют в качестве механизма поиска и выборки.

### XMLDB

Поддержка типа данных XML была встроена в СУБД Oracle9i. Структурированный XML-объект хранится в объектно-реляционной базе данных в соответствии со спецификацией W3C DOM. Встраивание синтаксиса XPath в поисковые запросы на языке SQL отвечает спецификациям группы SQLX. Язык XQuery также поддерживается.

## Средства установления соединения с базой данных

Установление соединения между клиентом и сервером базы данных - ключевой компонент всей архитектуры. По этому соединению передаются все данные, запрашиваемые приложением. В Oracle включены различные средства для установления и настройки соединения с базой данных

(описаны в отдельных разделах ниже). Мы разбили обсуждение на две части: сетевые компоненты СУБД и продукт Oracle Application Server.

## Сетевые компоненты СУБД

Пользователи подключаются к базе данных, устанавливая с ней соединение по сети. Можно также связать между собой по сети различные серверы базы данных. Oracle предлагает несколько способов установления соединений между пользователем и базой данных или между различными серверами баз данных, как описано в следующих разделах.

### Oracle Net

Интерфейс с сетью Oracle Net в версии Oracle8 назывался Net8, а в более ранних версиях - SQL*Net. Он поддерживает широкий спектр сетевых протоколов, хотя самый распространенный сегодня - TCP/IP. Средства, ассоциируемые с Oracle Net, например разделяемые серверы, в совокупности называются Oracle Net Services.

### Oracle Internet Directory

Служба интернет-каталогов Oracle Internet Directory (OID) впервые появилась в версии Oracle8L OID заменила прежнюю службу Oracle Names, поскольку позволяет пользователю соединиться с сервером Oracle Server, не создавая конфигурационный файл на стороне клиента. OID представляет собой LDAP-совместимый каталог (Lightweight Directory Access Protocol), а потому поддерживает Oracle Net и другие протоколы на основе LDAP.

### Oracle Connection Manager

Каждое соединение с базой данных потребляет дефицитные сетевые ресурсы, и это может отразиться на производительности приложения. Менеджер соединений (Connection Manager, CMAN), показанный на рис. 1.3, позволяет уменьшить количество сетевых соединений клиентов Oracle Net с сервером за счет применения *концентраторов*, задача которых - мультиплексировать соединения, объединив несколько логических соединений в одно физическое. Достоинства механизма мультиплексирования соединений становятся очевидными при большом количестве активных пользователей.

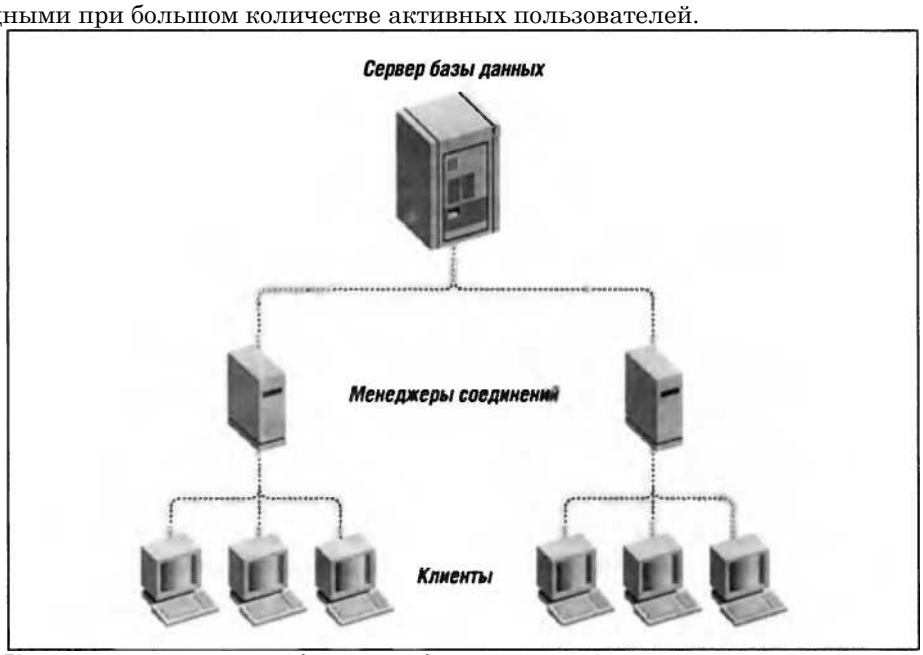



*Рис. 1.1. Концентраторы и менеджеры соединений при большом количестве пользователей*

Менеджер соединений позволяет также работать с несколькими сетевыми протоколами, если в сети имеются клиенты или серверы, не использующие TCP/IP. В версии Oracle Database 10# появилась возможность динамически конфигурировать менеджер соединений, то есть изменять его параметры, не останавливая процесс CMAN.

## Oracle Application Server

Широкое распространение приложений для Интернета и сетей интранет стало причиной перехода от архитектуры клиент/сервер (когда значительные части приложения реализованы в виде «толстых» клиентов) к трехуровневой архитектуре (когда броузер предоставляет все, что нужно «тонкому» клиенту). Сервер приложений Oracle Application Server позволяет развернуть промежуточный слой трехуровневой архитектуры для веб-приложений, компонентных

приложений и интеграции приложений масштаба предприятия. Oracle Application Server - основная часть продукта Fusion Middleware, допускающая масштабирование на несколько серверов промежуточного слоя.

Этот продукт включает веб-прослушиватель на базе популярного сервера Apache, сервлеты и сценарии JavaServer Pages (JSP), бизнес-ло- гику и/или компоненты для доступа к данным. Бизнес-логика часто развертывается в виде компонентов Enterprise JavaBeans (EJB). Компоненты для доступа к данным могут быть написаны с применением JDBC, SQLJ и EJB. TopLink - это инструмент отображения, который связывает Java-объекты с базой данных через JDBC, так что разработчик на Java может не думать о конструировании вызовов SQL и об ошибках приложения, вызванных изменениями в схеме базы данных. Oracle Application Server предлагает также механизм кэширования и готовые решения задач, возникающих при создании порталов, систем бизнес-анализа и беспроводного доступа.

*Кэширование*

Компонент Oracle Application Server Web Cache реализует промежуточный уровень для кэширования веб-страниц целиком или частично. Предшествующий механизм Oracle Application Server Database Cache, который использовался для кэширования PL/SQL-процедур и анонимных PL/SQL-блоков, начиная с версии Oracle Application Server 10g не поддерживается.

*Портал*

Компонент Oracle Application Server Portal входит также в продукт Oracle Developer Suite и применяется для создания простых в использовании корпоративных порталов. Разработанный портал развертывается внутри Application Server.

*Бизнес-анализ*

В состав продукта Application Server Business Intelligence входит компонент Portal, а также оригинальные инструменты бизнес-ана- лиза, разработанные Oracle:

- Oracle Reports - масштабируемый промежуточный слой для вывода результатов заранее заданных запросов в виде отчетов;
- Oracle Discoverer для предъявления произвольных запросов и анализа результатов;
- платформа развертывания для разработанных в JDeveloper приложений для OLAP-обработки и добычи данных.

*Oracle Wireless*

В состав компонента Oracle Wireless (бывший Oracle Portal-to-Go) входят:

- контент-адаптеры для преобразования информационного содержимого в формат XML;
- преобразователи форматов (device transformer) для преобразования из XML в язык разметки, поддерживаемый конкретным устройством;
- порталы персонализации для персонализации оповещений, адресов назначения оповещений, адресных меток (location mark) и профилей; кроме того, беспроводной портал персонализации применяется для создания, обслуживания, тестирования и публикации URL службы, а также для управления пользователями.

Продукт Oracle Application Server поставляется в нескольких редакциях: Enterprise Edition, Standard Edition, Standard Edition One и Java Edition; последний включает компоненты, необходимые разработчикам на Java. В Standard Edition и Standard Edition One включены компоненты Portal, TopLink вместе с Application Development Framework и Web Cache. В Enterprise Edition добавлены следующие компоненты: Forms Services, Reports Services, Discoverer Viewer, Oracle Internet Directory, Oracle Application Interconnect, Wireless Option и интеграция с Enterprise Service Bus (ESB). В Java Edition входят компоненты HTTP Server, OC4J и TopLink вместе с Application Development Framework.

Для редакции Oracle Application Server Enterprise Edition имеется еще ряд дополнительных опций:

*BPEL Process Manager Option*

Инструмент Business Process Execution Language (BPEL, язык исполнения бизнес-процессов) спроектирован для работы в сервисноориентированных архитектурах (SOA) и применяется для создания, администрирования и развертывания бизнес-процессов, связывающих несколько приложений. Он поддерживает стандарты BPEL, Web Services, XML, XSLT, XPATH, JMS и JCA.

*BusinessActivity Monitoring (BAM)*

Компонент BAM служит для построения инструментальных панелей реального времени, на которых отображаются основные индикаторы производительности (key performance indicator, KPI), содержащие данные от оповещений, поступающих через Сеть.

*BI Publisher*
Инструмент форматирования отчетов, применяемый для генерирования высококачественных отчетов на основе данных в формате XML.

Service Registry
Реестр служб Oracle Service Registry позволяет публиковать информацию о службах и ссылку на авторитетную систему (System of Record) для SOA-служб.

*Комплект SOA Suite для Oracle Middleware*
В этот комплект входят компоненты Oracle Fusion Middleware для SOA: BPEL, BAM, движок бизнес-правил, Enterprise Service Bus (механизм обмена сообщениями, маршрутизации и трансформации), Web Services Management (включает менеджер политик и инструментальную панель мониторинга), Web Services Registry, а также адаптеры приложений и технологий.

*Communication and Mobility Server*
В этот продукт входит компонент TimesTen, а также SIP Servlet Container, каркас активации и активаторы, средства голосового и мобильного доступа.

*WebCenter*
WebCenter - последняя разработанная Oracle инфраструктура для построения порталов. Применяется для развертывания портлетов HAjax-компонентов, особенно для приложений, следующих принципам Web 2.0. Включает форумы, сервер присутствия, клиент системы мгновенной передачи сообщений, Wiki, установление и разрыв VOIP-вызова, SIP Servlet Container, API для Java и веб-служб, интеграцию с системой Click-2-dial и программный клиент с поддержкой голосовой связи.

*Адаптеры для Fusion Middleware*
Имеются адаптеры для приложений, мониторов обработки транзакций, EDI и другие.

Комплект Fusion Middleware SOA Suite служит основой архитектуры интеграции приложений Application Integration Architecture (ALA). BALA включены также готовые бизнес-объекты и бизнес-процессы под общим названием Process Integration Packs. Эта архитектура является фундаментом для интеграции существующих и будущих приложений Oracle.

## Распределенные базы данных

СУБД Oracle славится умением обрабатывать очень большие объемы данных и поддерживать множество одновременно работающих пользователей. Oracle не только хорошо масштабируется для развертывания на все более мощных одиночных системах, но может быть развернута и в распределенной конфигурации. Экземпляры Oracle, развернутые на нескольких платформах, можно объединить, представив в виде логически единой распределенной базы данных.

В этом разделе описаны основные способы взаимодействия между базами данных в распределенной системе.

## Распределенные запросы и транзакции

Корпоративные данные часто распределены по нескольким базам из соображений емкости и распределения сфер ответственности. Но пользователям бывает нужно запрашивать или обновлять распределенные данные так, как будто они находятся в одной базе.
Корпорация Oracle первой ввела распределенные базы данных еще в начале 1980-х в ответ на требования организовать доступ к данным на разных платформах. *Распределенные запросы* позволяют извлекать данные из нескольких баз. *Распределенные транзакции* служат для вставки, удаления или обновления данных, находящихся в распределенной базе. Механизм двухфазной фиксации гарантирует, что все серверы баз данных, участвующие в транзакции, либо зафиксируют, либо откатят ее. Фоновые процессы восстановления гарантируют непротиворечивость базы данных при сбое системы во время обработки распределенной транзакции. Когда отказавшая система станет доступна, тот же самый процесс завершит распределенные транзакции.

Распределенные транзакции можно реализовать также с помощью распространенных мониторов транзакций (TP), которые взаимодействуют с Oracle по стандартному (X/Open) интерфейсу XA. В Oracle8i был добавлен механизм координации транзакций посредством сервера Microsoft Transaction Server (MTS), поэтому теперь распределенная транзакция, инициированная MTS, может распространяться и на базу данных Oracle.

## Heterogeneous Services

Компонент Heterogeneous Services позволяет обращаться из СУБД Oracle к данным, хранящимся в других базах, и сторонним службам с помощью обобщенных интерфейсов

установления связи ODBC и OLE-DB.

Дополнительный компонент Transparent Gateways пользуется агентами, специально разработанными для различных оконечных систем. Этот компонент позволяет формулировать запросы на диалекте языка SQL для Oracle и отправлять их другой СУБД. При этом запрос автоматически и прозрачно для пользователя будет транслирован на диалект SQL, понятный источнику данных. Помимо предоставления доступа к сторонним SQL-службам компонент Heterogeneous Services реализует транзакционность с помощью протокола двухфазной фиксации Oracle для других баз данных и процедурных служб, которые вызывают написанные на языке третьего поколения функции в системах, не управляемых Oracle. Пользователь взаимодействует с базой данных Oracle так, будто все объекты хранятся в ней, а компонент Heterogeneous Services прозрачно обращается к «чужой» базе данных от имени пользователя.

## Средства перемещения данных

При использовании распределенных баз данных часто требуется перемещать данные из одной базы данных Oracle в другую. А иногда необходимо организовать несколько копий одной и той же базы в разных местах, чтобы уменьшить объем сетевого трафика или повысить доступность данных. Вы можете экспортировать сами данные и словари данных (метаданные) из одной базы и импортировать их в другую. В Oracle Database 10g для экспорта/импорта была реализована высокоскоростная помпа данных (data pump).

Oracle предлагает много других дополнительных средств этой категории: переносимые табличные пространства, компоненты Advanced Queuing и Oracle Streams, а также решения для извлечения, трансформации и загрузки (ETL) данных. Рассмотрим их подробнее.

## Переносимые табличные пространства

Переносимые табличные пространства впервые появились в версии Oracle8i. Вместо того чтобы запускать процесс экспорта/импорта, который сбрасывает данные и описывающие их структуры в промежуточный файл для последующей загрузки, можно перевести табличное пространство в режим чтения, перенести или скопировать его из одной базы в другую, а затем смонтировать. При этом в исходной и конечной базах словари, описывающие табличное пространство, должны быть одинаковыми. Такой метод позволяет сэкономить немало времени в случае перемещения больших объемов данных. Начиная с версии Oracle Database 10g можно переносить табличные пространства между различными платформами или операционными системами.

## Advanced Queuing и Oracle Streams

Компонент Advanced Queuing (AQ), впервые появившийся в версии Ora- cle8i, позволяет асинхронно посылать сообщения из одной базы данных Oracle в другую. Поскольку сообщения хранятся в очереди внутри базы данных и посылаются асинхронно, когда устанавливается соединение, накладные расходы и объем сетевого трафика оказываются гораздо ниже, чем при использовании традиционных способов гарантированной доставки с помощью протокола двухфазной фиксации транзакции, включающей исходную и конечную базы данных. Сохраняя же сообщения в базе, AQ обеспечивает более надежный механизм восстановления, чем при других реализациях очередей с хранением сообщений в файловой системе.

Наличие механизма передачи сообщений в Oracle открывает возможность разработки и развертывания решений на базе публикации/под- писки с применением правил для определения подписавшихся приложений. Когда в списке рассылки публикуется новый контент, анализируются заданные для этого списка правила и принимается решение, каким подписчикам он должен быть отправлен. При таком подходе единственный список рассылки может эффективно обслужить потребности различных сообществ подписчиков. В первой версии Oracle9i в компонент AQ были добавлены поддержка XML и интеграция с каталогом Oracle Internet Directory (OID).

Во второй версии Oracle9i компонент AQ стал частью подсистемы Oracle Streams. Последняя состоит из трех основных компонентов: репликация по журналу для сбора данных, очереди для промежуточного хранения данных и определяемые пользователем правила потребления данных. Начиная с Oracle Database 10g Streams включает поддержку технологии Change Data Capture (отслеживание изменений в источниках данных) и передачи файлов. Подсистема Streams управляется из программы Oracle Enterprise Manager.

## Извлечение, трансформация и загрузка данных

Инструмент Oracle Warehouse Builder (OWB) служит для проектирования целевых баз данных, особенно используемых в качестве хранилищ (data warehouses), и предоставляет репозиторий метаданных. Однако он более широко известен как графический инструмент построения отображения исходной базы на конечную и генерации сценариев извлечения, трансформации

и загрузки данных (ETL). OWB пользуется средствами ETL, которые впервые были встроены в СУБД в версии Ora- cle9i. OWB поставляется в составе СУБД Oracle начиная с версии Oracle Database 10g Release 2.

Дополнительно Oracle предлагает инструмент интеграции данных Oracle Data Integrator (ODI), который не так тесно связан с СУБД Oracle, как OWB (хотя база данных Oracle может быть как исходной, так и конечной). Oracle Data Integrator основан на продукте компании Sunop- sis, приобретенной Oracle. Помимо средств ETL ODI может генерировать код веб-служб для развертывания в архитектуре SOA и является ключевым компонентом стратегии интеграции с SOA, реализованной в Oracle.

## Средства повышения производительности

В Oracle имеется несколько механизмов, специально предназначенных для повышения производительности в определенных ситуациях. Мы отнесли их к двум категориям: распараллеливание работы базы данных и организация хранилищ данных.

## Распараллеливание работы базы данных

Распараллеливание повышает скорость выполнения запросов, настройки и обслуживания базы данных. Разбив одну задачу на несколько меньших подзадач, каждая из которых выполняется в отдельном процессе, можно весьма заметно повысить производительность некоторых операций в базе данных. Вот некоторые типы запросов, которые могут быть распараллелены:

- сканирование таблицы;
- вложенные циклы;
- соединение таблиц методом сортировки и слияния;
- группировка GROUP BY;
- подзапросы типа NOT IN (антисоединение);
- определенные пользователем функции;
- сканирование индекса;
- SELECT DISTINCT UNION и UNION ALL;
- соединение таблиц методом хеширования;
- ORDER BY и агрегирование;
- соединение типа «звезда» по битовым индексам (bitmap star joins);
- соединение по секциям (partition-wise join);
- хранимые процедуры (на языках PL/SQL и Java, а также внешние подпрограммы).

Помимо запросов распараллеливанию поддаются многие другие средства Oracle.

## Организация хранилищ данных и бизнес-анализ

Хотя распараллеливание повышает производительность СУБД Oracle в целом, к быстродействию систем бизнес-анализа и хранилищ данных предъявляются особые требования.

### Битовые индексы

В Oracle 7.3 была добавлена поддержка битовых индексов, обеспечивающих быструю выборку некоторых типов данных. Лучше всего битовые индексы работают для столбцов, в которых число различных значений мало по сравнению с общим числом строк в таблице.

В битовом индексе не хранятся фактические значения. Вместо этого каждому возможному значению сопоставляется один бит, который равен 1, если строка содержит это значение, и 0 в противном случае.

### Оптимизация запросов типа «звезда»

Типичный запрос к хранилищу данных адресован большой *таблице фактов*, которая связана внешними ключами с гораздо меньшими по размеру *таблицами измерений*. В версии Oracle 7.3 была реализована оптимизация таких *запросов типа «звезда»*. Выигрыш в производительности достигается за счет построения декартова произведения таблиц измерений и последующего единственного соединения с таблицей фактов. В Oracle8 этот механизм получил дальнейшее развитие и называется *параллельным соединением типа «звезда» по битовым индексам* (parallel bitmap star join). В нем используются битовые индексы по внешним ключам таблиц измерений, чтобы ускорить вычисление соединения типа «звезда» для большого количества таблиц измерений.

### Материализованные представления

Начиная с Oracle8i, материализованные представления были еще одним способом существенно

повысить скорость выполнения запросов. Идея заключается в том, что информация, извлеченная из таблицы фактов, группируется по значениям полей из таблиц измерений, и полученные сводные данные сохраняются в виде материализованного представления. Если запрос может использовать это представление, то он прозрачно для пользователя переадресуется к нему. В каждой версии Oracle появляются новые способы оптимизации работы с материализованными представлениями.

### Аналитические функции

В Oracle и других СУБД все заметнее тенденция включать аналитические и статистические функции, доступные из SQL. Впервые такая возможность появилась в версии Oracle8i, когда были включены функции CUBE и ROLLUP. На сегодняшний день имеются также функции ранжирования, оконные агрегатные функции, функции запаздывания и опережения, линейная регрессия, дескриптивные статистики, корреляция, кросс-табуляция, проверка гипотез, подбор распределения и анализ Парето.

### Подсистема OLAP Option

Подсистема OLAP Option физически сохраняет многомерные кубы в реляционной базе Oracle. Чаще всего к этим кубам обращаются с помощью SQL, хотя имеется и Java API. Начиная с версии Oracle Database 11 *g* оптимизатор Oracle распознает уровни внутри кубов. В результате любой инструмент бизнес-анализа может прозрачно для пользователя получить выигрыш от повышения производительности. Обновления значений в кубах теперь выполняются аналогично обновлению материализованных представлений.

### Подсистема Data Mining Option

Начиная с версии Oracle9i в СУБД встроены популярные алгоритмы добычи данных. Они включены в подсистему Data Mining Option, а обратиться к ним можно посредством PL/SQL или специального Java APL Приложения для добычи данных, в которых применяются эти алгоритмы, обычно пишутся с помощью программы DataMiner производства Oracle или инструментов, поставляемых компаниями-партне- рами Oracle, например InforSense или SPSS. В подсистеме Data Mining Option для Oracle Databasellg реализованы следующие алгоритмы: наивная байесовская фильтрация, ассоциации, адаптивные байесовские сети, кластеризация, машины опорных векторов (SVM), факторизация неотрицательной матрицы (NMF), деревья решений и обобщенные линейные модели.

### Инструменты бизнес-анализа

К хранилищам данных в Oracle обычно обращаются из инструментов бизнес-анализа известных сторонних поставщиков. Однако по мере того как корпорация Oracle расширяет спектр предложений, приобретая другие компании, все чаще применяются предлагаемые ею программы. Изначально в состав Oracle включались только инструменты Oracle Discoverer и Reports (они до сих пор входят в состав Application Server или поставляются в составе отдельного продукта Oracle Business Intelligence Standard Edition Suite).

Флагманским продуктом Oracle в этой области является Oracle Business Intelligence Enterprise Edition Suite (OBI EE), который первоначально включал продукты бывшей компании Siebel Analytics: Oracle Answers, Dashboards, Delivers, BI Publisher и Office Plug-ins. В продукте OBI EE Plus это предложение было расширено за счет компонентов компании Hyperion: Foundation Services, Interactive Reporting, SQR production reporting, Financial Reporting, SmartView for Office и Web Analysis.

Для построения OLAP-куба и сопутствующих функций независимо от хранилища данных в базе теперь доступен компонент Essbase. Подмножество OBI EE включено в продукт Business Intelligence Standard Edition One вместе с редакцией СУБД Oracle Standard Edition One и Oracle Warehouse Builder.

Oracle предлагает также приложения бизнес-анализа, включающие средства моделирования данных, их анализа и генерации отчетов; при этом они уже заполнены готовыми метаданными о бизнесе. К флагманским приложениям относятся Oracle Business Intelligence Applications (прежнее название Siebel Business Analytics Applications) и Hyperion Financial Performance Management Applications.

## Средства управления базой данных

В Oracle включено много функций, упрощающих администрирование базы данных. Кардинально эта область улучшилась в версии Oracle Database 10g, а в Oracle Database llg ее эволюция продолжилась в сторону большей автоматизации настройки и управления. Если вы продолжаете администрировать базы данных Oracle по старинке (например, с помощью сценариев), но собираетесь перейти на новую версию, то самое время пересмотреть свой подход.

Начиная с версии Oracle Database 10g статистика собирается автоматически и сохраняется в репозитории рабочей нагрузки Automatic Workload Repository (AWR) внутри базы данных.

Автоматический диагностический монитор базы данных Automatic Database Diagnostic Monitor (ADDM) периодически обрабатывает статистику и посылает оповещения о возможных проблемах программе Oracle Enterprise Manager, в которой можно проанализировать ситуацию более подробно и, если необходимо, принять меры. Некоторые функции, теперь полностью автоматизированные, например, Automatic Memory Management (автоматическое управление памятью), также пользуются данными, сохраняемыми в AWR.

Автоматизированные рекомендации, которые дает Oracle, основаны на состоянии базы данных, близком к реальному времени. Часто рекомендации оказываются более точными, чем было возможно раньше при использовании ручных процедур. В следующих разделах мы покажем, как это влияет на программу Oracle Enterprise Manager, дополнительные пакеты, подсистему Information Lifecycle Management, резервное копирование и восстановление, а также на доступность базы данных.

## Oracle Enterprise Manager

Программа Oracle Enterprise Manager (EM) включена в большинство редакций СУБД. EM предоставляет инфраструктуру для создания инструментов администрирования базы данных и HTML-интерфейс для управления пользователями, экземплярами и различными подсистемами. С помощью EM можно также администрировать Oracle Application Server, Oracle Applications, операционную систему Linux и программные продукты других поставщиков.

На консоль базы данных в текущей версии Oracle выводится информация о состоянии базы данных, доступности, схеме, конфигурации средств перемещения данных и сопровождении ПО. В Oracle Database llg появился новый инструмент Support Workbench со своей инфраструктурой диагностики, предназначенной для передачи информации о возникших проблемах в службу технической поддержки Oracle. Одновременно к репозиторию EM могут обращаться несколько администраторов баз данных.

Развернуть EM можно разными способами: как центральную консоль для мониторинга нескольких баз данных с помощью агентов, как «консоль продукта» (по умолчанию устанавливается вместе с каждой базой данных) или для удаленного доступа (этот режим иногда называют «студийным»). Если Enterprise Manager развернут как центральная консоль, то его называют «Центром управления решеткой» (Grid Control) и используют для быстрой установки программного обеспечения Oracle, подготовки к работе и автоматизированного наложения заплат.

Подмножество функциональности Enterprise Manager доступно в броузере Microsoft Pocket PC Internet Explorer для КПК с помощью программы EM2Go, способной следить за состоянием СУБД Oracle и сервера Oracle Application Server.

## Information Lifecycle Management и ILM Assistant

Подсистема управления жизненным циклом информации Information Lifecycle Management (ILM), появившаяся в 2006 году, предоставляет средства для определения классов данных и уровней хранения. При этом она перемещает данные на те уровни хранения, которые обеспечивают оптимальное сочетание производительности и стоимости. Интерфейс ILM Assistant для настройки и администрирования ILM можно загрузить с сайта Oracle Technology Network по адресу http://otn.or- acle.com.

## Резервное копирование и восстановление

Каждому администратору базы данных известно, что резервное копирование - скучное, но необходимое занятие. Если резервная копия снята неправильно, то восстановление с нее сильно затруднено, а то и вовсе невозможно. К сожалению, важность этой повседневной задачи часто осознают лишь после утраты критически важных данных в результате сбоя системы. В следующих разделах мы расскажем о некоторых средствах, применяемых для резервного копирования.

### Recovery Manager

Основные виды резервных копий: полная копия базы данных (наиболее часто встречается), копия табличного пространства, копия файла данных, копия управляющего файла и копия архивного журнала. Вверсии Oracle8 появилась программа Recovery Manager (RMAN), с помощью которой сервер управляет резервным копированием и восстановлением базы данных, используя хранящийся в ней каталог восстановления (Recovery Catalog). RMAN умеет автоматически находить, копировать и восстанавливать (полностью или до определенного момента) файлы данных, управляющие файлы и архивные журналы. Начиная с версии Oracle9i RMAN может перезапускать процесс резервного копирования или восстановления и реализует политики окна восстановления по истечении срока хранения резервной копии. В Oracle

Enterprise Manager имеется графический интерфейс к RMAN. В версии Oracle Enterprise Manager 10g появился усовершенствованный планировщик заданий, с помощью которого можно настроить RMAN для автоматического запуска резервного копирования с записью на диск.

### Инкрементное резервное копирование и восстановление

В версии Enterprise Edition RMAN может также снимать инкрементные резервные копии баз данных. В этом случае копируются только блоки, модифицированные с момента снятия последней копии файла данных, табличного пространства или базы данных. Поэтому инкрементные копии получаются меньше, а восстановление с них выполняется быстрее, чем с полных. RMAN также умеет выполнять восстановление до заданного момента времени, что позволяет получить состояние данных непосредственно перед нежелательным событием (например, перед случайным удалением таблицы).

### Oracle Secure Backup

Программу RMAN применяют различные поставщики ПО для управления носителями, но начиная с версии Oracle Database 10g, в СУБД уже входит упрощенное решение для управления хранением резервных копий на магнитных лентах, которое называется Oracle Secure Backup XE. Дополнительно Oracle предлагает полномасштабное решение - Oracle Secure Backup.

## Доступность базы данных

Доступность базы данных зависит от надежности и правильности администрирования СУБД, операционной системы и аппаратных компонентов. Oracle повышает доступность за счет сокращения времени резервного копирования и восстановления. Достигается это следующими методами:

- возможность оперативного и параллельного резервного копирования и восстановления;
- улучшенное управление оперативными данными за счет секционирования;
- использование аппаратных средств для улучшенного мониторинга и перехвата управления при отказе.

Эти методы описаны в следующих разделах.

### Секционирование

Секционирование (partitioning) впервые введено в версии Oracle8 с целью повысить управляемость и доступность. Отдельные секции можно вывести из оперативного режима для обслуживания, сохранив доступ к остальным. При реализации хранилищ данных секционирование иногда применяется для организации скользящих окон, основанных на диапазонах дат. Имеются и другие варианты секционирования: хеш-секционирование (когда данные разносятся по секциям на основе значения хеш-функции, обеспечивающей равномерное распределение) и секционирование по списку значений ключа (данные распределяются по секциям исходя из дискретных значений, например, географического местоположения). Начиная с версии Oracle Database llg можно также организовывать интервальное секционирование, в этом случае новые диапазоны создаются автоматически по мере вставки записей.

Различные виды секционирования можно сочетать для создания «составных» секций. Примерами могут служить секции типа «диапазон- диапазон», «диапазон-хеш», «диапазон-список», «список-диапазон», «список-хеш» и «список-список».

### Data Guard

Понятие резервной базы данных (standby database) впервые появилось в версии Oracle 7.3. Это копия рабочей базы данных, которая начинает использоваться, если последняя недоступна, например из-за сбоя основного сервера или во время профилактического обслуживания. Рабочая и резервная базы данных могут быть географически разнесены. Резервная база данных создается как копия рабочей и обновляется путем накатывания архивных журналов, создаваемых в процессе эксплуатации рабочей базы. Компонент Data Guard, появившийся в версии Oracle9i, полностью автоматизирует этот процесс; раньше копировать и накатывать журналы приходилось вручную. Агенты размещаются в местах расположения рабочей и резервной баз, а Data Guard Broker координирует выполнение команд. Единственная команда Data Guard инициирует восемь шагов, необходимых для перехвата управления при отказе.

Помимо поддержки физической резервной базы данных Data Guard может создавать логическую резервную базу. В этом случае архивные журналы Oracle преобразуются в транзакции SQL и накатываются на открытую резервную базу данных.

В Oracle Database 10g появилось несколько новых функций, в том числе поддержка наката данных из журнала в реальном времени, интеграция с ретроспективной (Flashback) базой данных и сжатие архивного журнала. Начиная с Oracle Database 10g поддерживается

пошаговое обновление (rolling upgrade). В версии Oracle Database llg опция Active Data Guard Option позволяет выполнять в резервной базе данных запросы, сортировку и генерацию отчетов даже в то время, когда накатываются изменения из рабочей базы.

### Fail Safe

Подсистема Fail Safe повышает надежность базы данных Oracle. Перехват управления при отказе (failover) реализуется с помощью второй системы или узла, который обеспечивает доступ к данным, находящимся на разделяемом диске, в ситуации, когда первая система или узел выходит из строя. Подсистема Fail Safe для Windows в сочетании со службой Microsoft Cluster Services гарантирует перехват управления при отказе системы.

Fail Safe - это инструмент восстановления после катастрофического сбоя, поэтому на время выполнения операции перехвата управления данные оказываются недоступными. Начиная с версии Oracle9i для повышения доступности сервера рекомендуется применять подсистему Real Application Clusters.

### Oracle Real Application Clusters

В версии Oracle9i на смену подсистеме Oracle Parallel Server (OPS) пришла технология Real Application Clusters (RAC). RAC поддерживает перехват управления при отказе, а также повышает степень масштабируемости на кластерных конфигурациях в системах UNIX, Linux и Windows. Ключом к повышению масштабируемости стал механизм Cache Fusion, который существенно уменьшает количество операций записи на диск. Ранее именно так работало управление блокировками данных. В версии Oracle Database 10g переносимость RAC была поднята на новый уровень за счет интегрированного «кластерного ПО» (clus- terware) для всех поддерживаемых RAC платформ.

Подсистема Real Application Clusters позволяет развертывать несколько экземпляров Oracle на нескольких узлах кластера или решетки (grid). RAC координирует трафик между системами или узлами, так что все экземпляры функционируют как единая база данных. В результате база данных способна масштабироваться на десятки узлов. Поскольку кластер предоставляет нескольким экземплярам возможность доступа к одним и тем же данным, отказ одного экземпляра не вызовет заметных задержек на время восстановления системы. Достаточно просто перенаправить пользователей на другой, работающий экземпляр. Осуществить прозрачный для пользователя перехват управления приложения могут с помощью интерфейса уровня вызовов Oracle Call Interface (OCI).

### Data Guard и RAC

Начиная с версии Oracle9i сочетание Data Guard и RAC заменило технологию Parallel Fail Safe. При наличии RAC механизм Data Guard обеспечивает автоматический перехват управления с ограниченным временем восстановления. Кроме того, он перенаправляет клиентов с отказавшего экземпляра на работающий, гарантируя быстрое установление нового соединения, и автоматически диагностирует состояние экземпляров.

### Automated Storage Management

В версии Oracle Database 10g появилась подсистема Automated Storage Management (ASM), которая обеспечивает оптимальное расслоение и зеркалирование данных для достижения максимальной производительности и доступности. Поскольку ASM управляется из программы Enterprise Manager, теперь администратор базы данных может сам выполнять это критически важное задание, не согласовывая свои действия с системным администратором.

## Real Application Testing Option

В версии Oracle Database llg появилась возможность повторять все операции, выполненные в промышленной базе, и тестировать влияние изменений в системе. Это обеспечивает подсистема Real Application Testing Option. В нее входят средство Database Replay и программа SQL Performance Analyzer. Database Replay собирает информацию о рабочей нагрузке в промышленной базе, в том числе о конкуренции, зависимостях и временных затратах. Затем файлы данных о рабочей нагрузке преобразуются в файлы воспроизведения и передаются программе Replay Client для обработки. Кроме того, Database Replay предоставляет средства формирования отчетов о производительности и об имевших место ошибках. SQL Performance Analyzer получает подлежащие анализу данные о рабочей нагрузке, измеряет производительность до и после изменений в базе данных и показывает, как изменилось время выполнения каждой SQL-команды.

## Средства обеспечения безопасности базы данных

В Oracle имеются базовые средства безопасности для управления правами доступа

пользователей путем задания ролей и привилегий. Программа Enterprise Manager позволяет выполнять это локально или глобально. В последнем случае используется механизм безопасности уровня предприятия, являющийся частью подсистемы Advanced Security Option.
Имеющиеся в Oracle средства обеспечения безопасности позволяют реализовать виртуальную частную базу данных (Virtual Private Database, VPD) путем создания политик и присоединения их к таблицам, представлениям или синонимам. Выполнение политик обеспечивается путем добавления предикатов к предложению WHERE команды SELECT, INSERT, UPDATE, DELETE или INDEX.
Во многих организациях требуется более строгая защита данных, хотя в наши дни доступ к базе данных может производиться из точек за пределами организации. Корпорация Oracle включила в СУБД средства безопасного развертывания и в таких сложных условиях. Речь идет о подсистемах Advanced Security Option, Label Security Option, Database Vault и Audit Vault.

### Advanced Security Option

Подсистема Advanced Security Option раньше называлась Advanced Networking Option (ANO). Основные механизмы обеспечения повышенной безопасности Oracle Net - это шифры RC4 (разработка компании RSA Data Security), Data Encryption Standard (DES), Triple DES и Advanced Encryption Standard (AES).
Для аутентификации можно применять систему Kerberos, RADIUS или Distributed Computing Environment (DCE). Для проверки целостности данных применяются алгоритмы MD5 и SHA-1. В версии Oracle Database 11g добавилось прозрачное шифрование данных и расширенная аутентификация посредством Kerberos с применением типов шифрования Oracle.

### Label Security Option

Подсистема Label Security Option управляет доступом к данным, сравнивая метки, сопоставленные строкам данных, с правами доступа к меткам, которые хранятся в привилегиях пользователя. В одной базе данных может быть несколько уровней авторизации. Авторизация меток задается в менеджере политик Policy Manager. Политики применяются к базовым объектам базы данных, а не к представлениям, что заметно упрощает задачу управления доступом к данным и повышает степень безопасности.

### Database Vault Option

Подсистема Database Vault Option обеспечивает детальное управление доступом к данным со стороны любого пользователя, включая и администраторов базы данных. Администратор по безопасности может задать условия, определяющие возможность доступа к базе, и проводить аудит различных аспектов безопасности. На более детальном уровне можно определить области (realm), чтобы разрешить доступ только конкретным ролям или лишь к определенным схемам.

### Audit Vault Server

Сервер аудита Oracle Audit Vault Server ведет мониторинг таблиц аудита в базе данных, журналов и управляющих файлов операционной системы, отслеживая подозрительные действия. Он может генерировать отчеты и отправлять оповещения при регистрации необычной активности.

### Инструменты разработки Oracle

В распоряжении разработчиков имеется много инструментов, позволяющих представлять данные и создавать более сложные приложения для работы с базой данных Oracle. Хотя эта книга посвящена собственно СУБД Oracle, мы кратко опишем основные инструменты, которые Oracle предлагает для разработки приложений: Oracle JDeveloper, Oracle SQL Developer и Oracle Developer Suite. Комплект Developer Suite, который иногда называют Oracle Internet Developer Suite, включает программы Oracle Forms Developer, Oracle Reports Developer, Oracle Designer, Oracle Discoverer Administrative Edition и Oracle Portal.

### Oracle JDeveloper

Oracle представила программу Oracle JDeveloper в 1998 году. Она позволяет разрабатывать простые приложения на языке Java без написания кода. Сейчас JDeveloper распространяется бесплатно, ее можно загрузить с сайта Oracle Technology Network. В нее входят: мастер форм данных Data Form Wizard, мастер Beans Express Wizard для создания компонентов JavaBeans и классов BeanInfo и мастер развертывания Deployment Wizard. JDeveloper включает также средства для работы с базой данных: различные драйверы для Oracle, редактор соединений Connection Editor, позволяющий скрыть сложность JDBC API, компоненты для привязки визуальных элементов управления к данным и прекомпилятор SQLJ, позволяющий встраивать

в код на Java команды SQL для доступа к базе данных. Приложения, разработанные на JDeveloper, можно развертывать на сервере приложений Oracle Application Server. Хотя мастеры JDeveloper позволяют программисту создавать Java-объекты без какого-либо кодирования, конечным результатом все же является сгенерированный код на Java.

### Oracle SQL Developer

Программа Oracle SQL Developer была представлена в 2006 году. Она позволяет соединяться с любой базой данных Oracle версии не ниже Oracle9i Release 2. SQL Developer умеет создавать соединение с базой данных Oracle, показывать хранящиеся в базе объекты, создавать и модифицировать объекты в базе, запрашивать и обновлять данные, экспортировать данные и их описания, импортировать данные, обрабатывать команды, создавать и запускать отчеты. Входящие в состав продукта инструменты поддерживают редактирование, отладку и запуск PL/SQL-сценариев. Кроме того, SQL Developer может показывать объекты в базах данных других производителей и предоставляет средства для миграции на СУБД Oracle.
SQL Developer распространяется бесплатно, его можно загрузить с сайта Oracle Technology Network. Имеются версии для Windows, Linux и Apple Mac OS X. Кроме того, Oracle поддерживает на сайте Oracle Technology Network форум, посвященный SQL Developer.

### Oracle Forms Developer

Oracle Forms Developer - это инструмент создания диаграмм и приложений на базе форм, которые могут быть развернуты как традиционные клиент-серверные приложения или для работы в трехуровневой архитектуре. В последнем случае приложение исполняется в броузере и обращается к серверу приложений Oracle Application Server. Developer - это язык четвертого поколения (4GL). Приложение на таком языке пишется не в виде процедурного кода, а путем задания значений свойств. Developer поддерживает широкий спектр клиентов, в том числе написанных на Java. Программа Forms Builder включает встроенную виртуальную Java-машину для тестирования веб-приложений.

### Oracle Reports Developer

Программа Oracle Reports Developer предоставляет среду разработки и развертывания для быстрого построения и публикации отчетов в Сети с помощью системы Reports for Oracle Application Server. Данные могут быть представлены в виде таблиц, матриц, отчетов с группировкой, графиков или сочетания всего перечисленного. Высокое качество презентации достигается с помощью каскадных таблиц стилей (CSS).

### Oracle Designer

Программа Oracle Designer представляет собой графическую систему быстрой разработки приложений (Rapid Application Development, RAD), охватывающую весь процесс создания приложения для работы с базой данных - от построения бизнес-модели до проектирования схемы, генерации и развертывания. Проекты и изменения хранятся в многопользовательском репозитории. Инструмент позволяет выполнять реинжиниринг имеющихся таблиц и схем из баз данных как Oracle, так и других производителей, для повторного использования и перепроектирования.

Designer включает также генераторы приложений для Oracle Developer, HTML-клиентов, обращающихся к Oracle Application Server, и на языке C++. Designer может генерировать новые приложения и реконструировать имеющиеся приложения, в том числе модифицированные. Это позволяет реализовать процесс *кругового конструирования* (round-trip engineering), когда разработчик сначала генерирует приложение с помощью Designer, потом модифицирует его, реконструирует и помещает изменения обратно в репозиторий Designer.

### Oracle Discoverer Administration Edition

Программа Oracle Discoverer Administration Edition позволяет настроить и администрировать уровень Discoverer End User Layer (EUL), принадлежащий предыдущему поколению инструментов бизнес-ана- лиза для Oracle. Назначение этого уровня - оградить от сложности SQL бизнес-аналитиков, использующих Discoverer как инструмент для выполнения произвольных запросов и анализа результатов. На всем протяжении процедуры построения EUL администратору помогают мастеры. Кроме того, администратор может ограничить ресурсы, доступные аналитикам; за превышением квот будет следить входящий в Discoverer менеджер запросов.

### Oracle Portal

Oracle Portal был выпущен в 1999 под названием WebDB. Это основанный на HTML инструмент

разработки веб-приложений и сайтов, управляемых контентом. Портальные приложения развертываются в броузере. В состав Portal входят мастеры для разработки компонентов приложения, инкапсулирующих сервлеты, для доступа к другим сайтам по протоколу HTTP. Разрабатываемые порталы допускают настройку под конкретного пользователя и развертываются на промежуточном слое в составе Oracle Application Server.

Oracle Portal привнес в WebDB важное усовершенствование - возможность создания и использования *портлетов*, позволяющих разбить веб-страницу на отдельные области, способные отображать информацию и взаимодействовать с пользователем независимо друг от друга. Например, из портлетов можно независимо обращаться к компонентам Answers, Discoverer и Reports.

Следующий продукт Oracle, реализующий инфраструктуру для создания порталов, - WebCenter - был выпущен в 2006 году и первоначально поставлялся как дополнительный компонент к Application Server.

## Встраиваемые базы данных

Семейство СУБД Oracle можно использовать во встраиваемых приложениях, но потребление памяти может оказаться недопустимо большим, а функциональность частично излишней. Сегодня Oracle предлагает другие встраиваемые базы данных, в том числе TimesTen, Berkeley DB и Oracle Database Lite. Они специально написаны так, что потребляют относительно мало ресурсов, и предназначены для других целей. Поэтому мы лишь кратко опишем их ниже и больше возвращаться к ним в этой книге не будем.

## Oracle TimesTen

Oracle TimesTen - это реляционная база данных, которая находится целиком в физической памяти и обычно применяется для высокопроизводительной обработки транзакций. Доступ к данным, хранящимся в TimesTen, осуществляется посредством SQL, JDBC, JMS и ODBC. База данных под управлением TimesTen может работать в режиме монопольного или разделяемого доступа и создаваться как постоянная или временная.

Обновление базы данных производится путем сбора данных с помощью библиотек TimesTen, скомпонованных с приложением, или из базы данных Oracle посредством механизма Cache Connect. Поскольку данные извлекаются из оперативной памяти и там же обновляются, среднее время считывания или обновления обычно составляет миллионные доли секунды. Механизм Cache Connect поддерживает кэширование данных, полученных из базы Oracle, как при чтении, так и при записи. Синхронизация TimesTen и Oracle может быть двусторонней.

Как и положено встраиваемым базам данных, TimesTen почти не требует администрирования. Возможна репликация из одной базы данных TimesTen в другую с помощью дополнительных средств, причем по умолчанию это делается асинхронно.

## Oracle Berkeley DB

Oracle Berkeley DB - это встраиваемый движок базы данных, потребляющий очень мало ресурсов и обеспечивающий блокировку на уровне записей. Поставляется в виде версий для Java и XML. База спроектирована для работы в одном процессе с приложением. Если Berkeley DB развертывается в таком режиме, то никакого отдельного администрирования базы данных вообще не требуется. Для работы может хватить всего 400 Кбайт.

В редакции Berkeley DB Java Edition поддерживаются Java Transaction API (JTA), J2EE Connector Architecture (JCA) и Java Management Extensions (JMX). Продукт в этом случае представляет собой единственный JAR-файл размером 820 Кбайт и работает в контексте той же виртуальной Java-машины, что и само приложение. Для доступа к Java-объектам предназначен слой Direct Persistence Layer (DPL).

Редакция Berkeley DB XML Edition чаще всего применяется в сетевых приложениях для управления контентом. Поддерживаются языки XQuery и Xpath.

Обе редакции можно сконфигурировать для обеспечения высокой доступности за счет репликации. Также поддерживается автоматическое восстановление. Решение о таком способе развертывания принимается на этапе проектирования приложения.

## Oracle Lite

Oracle Lite - это семейство продуктов для разработки мобильных приложений, нуждающихся в базе данных. Основные компоненты - Oracle Lite Database, Mobile Development Kit и Mobile Server (расширение Oracle Application Server).

Для ядра Oracle Lite Database требуется от 50 Кбайт до 1 Мбайт памяти в зависимости от платформы. Обращаться к базе можно с помощью языков Mobile SQL, C++ и Java. Также

поддерживается интерфейс ODBC. Поддержка Java включает написанные на Java хранимые процедуры и интерфейс JDBC. Оптимизация и администрирование Oracle Lite Database производятся автоматически. Эта СУБД может работать на карманных устройствах под управлением операционных систем Windows CE, Symbian, Windows и Linux.

Обычно при работе с Oracle Lite пользователь подключает свое карманное или мобильное устройство, в котором установлена база данных Oracle Lite Database, к полноценному серверу Oracle Database Server. После этого происходит автоматическая синхронизация данных между двумя системами. Затем пользователь может отключить устройство от сети и работать в автономном режиме. Сделав все необходимое, он снова подключается к серверу и синхронизирует данные.

Oracle Lite поддерживает различные механизмы синхронизации:

- двусторонняя синхронизация между мобильным устройством и сервером Oracle;
- синхронизация на базе модели « издатель-подписчик »;
- поддержка протоколов TCP/IP, HTTP, CDPD, 802.1 и HotSync.

Можно задать репликацию подмножеств данных с разными приоритетами. Поскольку нахождение данных в разных точках может приводить к конфликтам (в каком месте находится «правильная» версия?), предоставляется механизм автоматического разрешения конфликтов, допускающий и ручную настройку.

Mobile Server предоставляет единую платформу для публикации, развертывания, синхронизации и управления мобильными приложениями. Для контроля доступа к мобильным приложениям можно использовать развернутый в Сети центр управления. Кроме того, в состав Mobile Server вошел прежний продукт Oracle «Web-to-Go», который обеспечивает централизованный, управляемый мастерами механизм разработки и развертывания приложений.

# Лекция 2. Архитектура Oracle.

## Базы данных и экземпляры

Многие пользователи Oracle употребляют термины *экземпляр* и *база данных* как синонимы. На самом деле это разные (хотя и взаимосвязанные) вещи. Различие существенно, так как проливает свет на архитектуру Oracle.

В Oracle термином *база данных* описывается физическое хранилище информации, а термином *экземпляр* - программное обеспечение, работающее на сервере и предоставляющее доступ к информации в базе данных. Экземпляр исполняется на конкретном компьютере или сервере; база данных хранится на дисках, подключенных к этому серверу.

База данных - *физическая* сущность: она состоит из файлов, хранящихся на дисках. Экземпляр-сущность *логическая*: он состоит из структур в оперативной памяти и процессов, работающих на сервере. Например, Oracle использует область разделяемой памяти System Global Area (SGA, системная глобальная область) и области памяти в каждом процессе - Program Global Area (PGA, программная глобальная область). Экземпляр может быть частью одной и только одной базы данных. Напротив, с одной базой данных может быть ассоциировано несколько экземпляров. Время жизни экземпляров ограничено, тогда как база данных при должном обслуживании может существовать вечно.

Пользователи не имеют прямого доступа к информации, хранящейся в базе данных Oracle; они должны запрашивать информацию у экземпляра Oracle.

В реальном мире есть хорошая аналогия экземплярам и базам данных. Можно считать экземпляр мостом к базе данных, а саму ее - островом. Транспорт попадает на остров и уходит с него по мосту. Если мост перекрыт, то остров на месте, но транспорту туда не попасть. В терминологии Oracle, если экземпляр запущен, то данные могут попадать в базу и уходить из нее. Физическое состояние базы данных при этом изменяется. Если же экземпляр остановлен, то пользователи не могут обращаться к базе данных, пусть даже физически она никуда не делась. База данных в этом случае статична, никаких изменений в ней не происходит. Экземпляр снова запущен - и данные тут как тут.

## Структура базы данных Oracle

База данных состоит из табличных пространств, управляющих файлов, журналов, архивных журналов, файлов трассировки изменения блоков, ретроспективных журналов и файлов резервных копий (RMAN).

### Табличные пространства

Любые данные, хранящиеся в базе Oracle, должны находиться в каком- то табличном пространстве. *Табличное пространство* (tablespace) - это логическая структура; нельзя попросить операционную систему показать вам табличное пространство. Каждое табличное

пространство состоит из физических структур, называемых *файлами данных* (data- files). В одном табличном пространстве может быть один или несколько файлов данных, тогда как каждый файл данных принадлежит ровно одному табличному пространству. При создании таблицы можно указать, в какое табличное пространство ее поместить. Тогда Oracle найдет для нее место в одном из файлов данных, составляющих указанное табличное пространство.
На рис. 2.2 показано соотношение между табличными пространствами и файлами данных.
Здесь мы видим два табличных пространства в базе данных Oracle. При создании новой таблицы ее можно поместить в табличное пространство DATA1 или DATA2. Физически таблица окажется в одном из файлов данных, составляющих указанное табличное пространство.
Начиная с версии Oracle Database 10g Release 2 для всех типов таблиц по умолчанию подразумеваются *локально управляемые табличные пространства*. В таком табличном пространстве можно создавать *большие файлы*, то есть при работе в 64-разрядных системах задействуется возможность создавать сверхбольшие файлы.
В Oracle9i появился механизм файлов, управляемых Oracle (Oracle Managed Files, OMF), позволяющий автоматически создавать, именовать и, если понадобится, удалять все файлы, составляющие базу данных. OMF упрощает обслуживание базы данных, поскольку не нужно помнить имена всех составляющих ее файлов. К тому же не возникают проблемы из-за ошибок человека, ответственного за именование файлов. Начиная с версии Oracle Database 10g сочетание OMF и табличных пространств с большими файлами делает работу с файлами данных совершенно прозрачной.

Максимальное количество файлов данных в базе Oracle- 64 000. Поскольку табличное пространство с большими файлами может содержать файл, который в 1024 раза больше файла в табличном пространстве с малыми файлами, а размер блока в табличном пространстве с большими файлами для 64-разрядных операционных систем составляет 32 Кбайт, общий размер базы данных Oracle может достигать 8 экзабайт (1 экзабайт = 1 000 000 терабайт). Табличные пространства с большими файлами предназначены для использования совместно с подсистемой автоматического управления хранением Automatic Storage Management (ASM), иными менеджерами логических томов, поддерживающими расслоение, и RAID-массивами.

### Файлы базы данных

База данных Oracle состоит из физических файлов трех основных типов:

- управляющие файлы (control files);
- файлы данных (datafiles);
- журнальные файлы, или журналы (redo log files).

В управляющем файле хранится информация о местонахождении других физических файлов, составляющих базу данных, - файлов данных и журналов. Там же хранится важнейшая информация о содержимом и состоянии базы данных:

- имя базы данных;
- время создания базы данных;
- имена и местонахождение файлов данных и журнальных файлов;
- информация о табличных пространствах;
- информация о файлах данных в автономном режиме;
- история журналов и информация о порядковом номере текущего журнала;
- информация об архивных журналах;
- информация о наборах и фрагментах резервных копий, файлах данных и журналах;
- информация о копиях файлов данных;
- информация о контрольных точках.

Управляющие файлы не только содержат важную информацию, необходимую при запуске экземпляра, они полезны и при удалении базы данных. Начиная с версии Oracle Database 10g с помощью команды DROP DATABASE можно удалить все файлы, перечисленные в управляющем файле базы данных, а также сам управляющий файл.

## Инициализация базы данных

При запуске экземпляра Oracle считываются параметры инициализации. Они определяют, как база данных должна использовать физическую инфраструктуру и иную конфигурационную информацию об экземпляре. Параметры инициализации хранятся в файле параметров инициализации экземпляра, который обычно называют просто *INIT.ORAy* или, начиная с версии Oracle9*, в репозитории, который называется файлом параметров сервера (или *SPFILE).*

Количество обязательных параметров инициализации уменьшается с выходом каждой новой версии Oracle. В дистрибутиве Oracle есть пример файла инициализации, пригодный для запуска базы данных. Либо можно воспользоваться программой Database Configuration Assistant (DCA), которая подскажет обязательные значения (например, имя базы данных).

# Развертывание физических компонентов

Этот раздел не может заменить официальную документацию по установке Oracle. Наша цель - снабдить вас практическим руководством по планированию развертывания базы данных Oracle.

## Управляющие файлы

Для любой базы данных следует хранить по меньшей мере два управляющих файла на разных физических дисках. Без актуальной копии управляющего файла вы рискуете потерять информацию о составляющих вашей базы данных. Утрата управляющих файлов не обязательно фатальна, их можно и воссоздать. Но процедура воссоздания довольно сложна и рискованна, а избежать ее несложно.

Местоположение управляющих файлов определяется, как уже было сказано, параметром инициализации CONTROL_FILES. Он позволяет задать несколько управляющих файлов, например:

```
control_files = (/u00/oradata/control.001.dbf,
                 /u01/oradata/control.002.dbf,
                 /u02/oradata/control.003.dbf)
```

Этот параметр сообщает экземпляру, где искать управляющие файлы. Oracle гарантирует, что все копии управляющего файла одинаковы, то есть любые изменения вносятся синхронно. Если параметр не задан, Oracle создаст управляющий файл с именем по умолчанию или прибегнет к услугам компонента Oracle Managed Files (если тот активирован).

Многие базы данных Oracle развертываются на том или ином варианте RAID-массива, например RAID-1 или RAID-5, чтобы избежать потери данных в случае выхода диска из строя. Напрашивается вывод, что можно обойтись без нескольких копий, сохранив управляющий файл в защищенной дисковой памяти, и что утрата диска еще не означает утраты управляющего файла. Но этот вывод неправомерен по двум причинам:

1. Если в *массиве с расслоением* (striped array) или в *зеркальной паре* (mirror-pair) отказывает больше одного диска, то все данные, хранящиеся на этих дисках, теряются. Статистически это редкое событие, но все же такое случается, и тогда есть угроза повреждения или утраты управляющего файла. Поскольку вы и так будете по горло заняты восстановлением после множественных сбоев диска, вероятно, лучше при этом избежать хотя бы воссоздания управляющих файлов. Создание дополнительных копий, пусть даже хранящихся в избыточной дисковой памяти, - это дополнительный уровень защиты.

2. Избыточная дисковая память не поможет защититься от человеческих ошибок. Кто-то может случайно удалить или переименовать управляющий файл, затереть его другим или переместить в другое место. И зеркалированный диск честно отразит эти изменения. А при резервировании управляющих файлов хотя бы одна копия да останется.

Не стоит беспокоиться о том, что запись в несколько управляющих файлов понизит производительность. Обновление управляющих файлов - ничто по сравнению с другими операциями дискового ввода/вывода, производимыми Oracle.

### Файлы данных

В файлах данных находятся собственно данные, хранящиеся в базе: таблицы и индексы, словарь данных, в котором сохраняется информация об этих структурах, и сегменты отката, необходимые для реализации конкурентного доступа.

### Структура файла данных

Первый блок файла данных называется *заголовком файла данных*. В нем хранится важная информация, необходимая для поддержания целостности базы данных, в частности *структура контрольной точки*. Она представляет собой логическую временную метку, показывающую, когда в последний раз были записаны изменения в файл данных. Эта информация абсолютно необходима для процесса восстановления, поскольку определяет, какие журналы нужно накатить, чтобы привести файл данных к состоянию на текущий момент времени.

## Журнальные файлы

Журнальный файл содержит протокол всех изменений, произведенных в базе данных в результате выполнения транзакций и внутренних операций Oracle. Обычно измененные блоки кэшируются в памяти, поэтому в случае сбоя экземпляра может оказаться, что какие-то блоки не записались в файлы

данных. Тогда можно воспользоваться протоколом, хранящимся в журнальных файлах, чтобы воспроизвести изменения, оставшиеся не записанными в момент сбоя, и тем самым обеспечить согласованность транзакции.
Кроме того, журнальные файлы применяются для реализации операций отката (undo), инициируемых командой ROLLBACK. Незафиксированные изменения в базе данных откатываются, так что база остается в том состоянии, в котором находилась в момент последней фиксации.
Чтобы упростить работу в случая сбоя, рекомендуем после любой не зарегистрированной в журнале операции снимать резервную копию, если вы не можете позволить себе потерять созданный объект или по какой-то причине операцию невозможно повторить. Помимо использования слова NOLOGGING в отдельных командах, можно пометить таблицу или все табличное пространство атрибутом NOLOGGING. Тогда запись в журнал будет подавляться для всех операций с данной таблицей или с любой таблицей в помеченном табличном пространстве.

## Как Oracle используетжурнальные файлы

Заполнив один журнальный файл, Oracle автоматически переходит к следующему. Заполнив все имеющиеся журнальные файлы, Oracle

возвращается к первому и перезаписывает его. Для отслеживания журналов применяется порядковая нумерация. Порядковый номер записывается в текущий журнальный файл.

Чтобы лучше разобраться в именах журнальных файлов и их порядковых номерах, рассмотрим три таких файла с именами *redologl.log*, *redolog2.log, redolog3.log*. Когда Oracle только начинает их использовать, журналам присвоены порядковые номера 1, 2 и 3 соответственно. После того как Oracle вернется к первому журналу - *redologl.log*- и начнет перезаписывать его, журналу будет присвоен порядковый номер 4. Затем настанет очередь файла *redolog2.log*, который получит порядковый номер 5.

Помните, что операционная система идентифицирует журнальные файлы по имени, а Oracle определяет, в каком порядке заполнялись журналы, ориентируясь на порядковый номер. Поскольку журналы автоматически используются повторно, имя файла мало что говорит о его месте в последовательности.

На рис. 2.1 проиллюстрировано заполнение журналов и их циклический перебор.

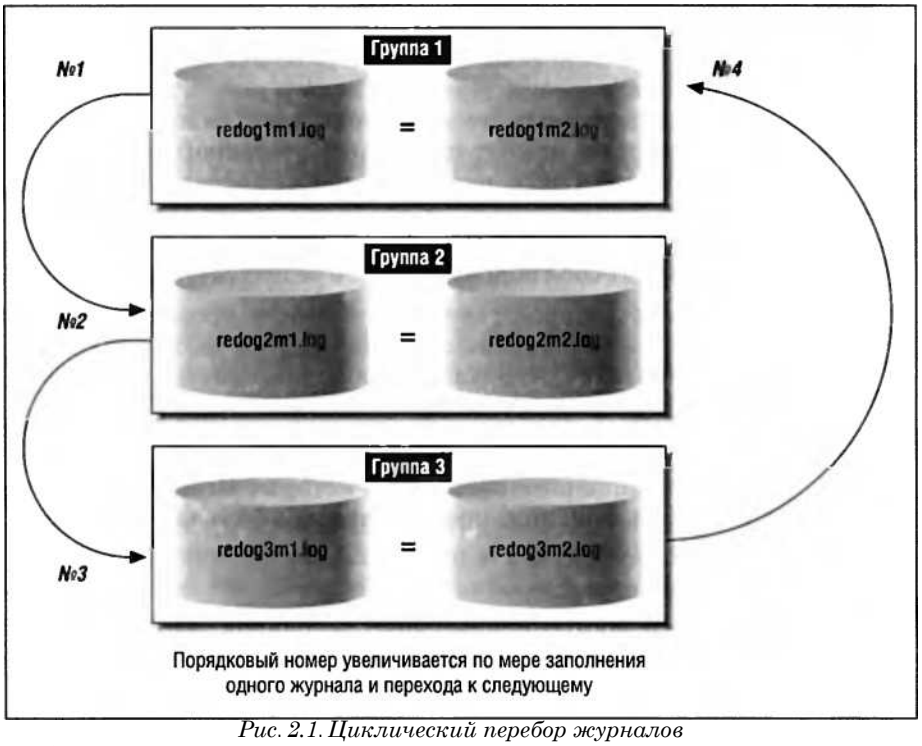



*Рис. 2.1. Циклический перебор журналов*

## Соглашения об именовании журнальных файлов

Имена различных файлов, составляющих базу данных, очень важны - по крайней мере, для тех, кто порой вынужден искать файлы по именам. Если вы не пользуетесь компонентом Oracle Managed Files, то должны выработать схему именования, отражающую назначение и существенные характеристики файла. Одно из возможных соглашений о выборе имен журнальных файлов показано на рис. 2.1:

```
redog1m1.log, redog1m2.log, ...
```

Префикс `redo` и суффикс . `log` отражают тот факт, что файл содержит журнал. В строках `g1m1` и `g1m2` закодирована информация о номерах группы и элемента. Это всего лишь пример, но мы рекомендуем принять какое-то соглашение, которое кажется вам осмысленным, и строго придерживаться его.

## Архивные журнальные файлы

Возможно, вас интересует, как избежать потери критически важной информации, коль скоро Oracle возвращается к ранее записанному журналу.

Есть два подхода к этой проблеме. Первый чрезвычайно прост: вы даже не пытаетесь избежать 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 восстановить всю базу данных в состоянии на момент последнего резервного копирования. Но воспроизвести изменения, внесенные позже последнего резервного копирования, без журнала не удастся. Таким путем идут очень немногие компании, работающие с Oracle, ведь невозможность восстановить ситуацию на момент сбоя недопустима - в результате теряются данные.

Второе (и более распространенное) решение проблемы 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, произведенные в базе данных, циклически перебирая файлы.

*Архивные журналы*

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

СУБД Oracle может работать в одном из двух режимов архивирования журналов:

*NOARCHIVELOG*

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

Тем, кто выбирает работу без архивирования журналов, будет доступно значительно меньшее количество вариантов и параметров резервного копирования базы данных (см. главу 11). Корпорация Oracle не рекомендует этот режим.

*ARCHIVELOG*

Переходя к новому журналу, Oracle архивирует предыдущий. Для того чтобы не допустить появления пропусков в истории изменений, повторное заполнение журнала начинается только после его успешного архивирования. Архивные и оперативные журналы содержат полную историю изменений, произведенных в базе данных. В совокупности они позволяют серверу восстановить все зафиксированные транзакции вплоть до момента сбоя. При работе в этом режиме возможно резервное копирование табличных пространств и отдельных файлов данных.

Использовать оперативные и архивные журналы на этапе восстановления базы данных серверу помогают вышеупомянутые внутренние порядковые номера.

## Режим ARCHIVELOG и автоматическое архивирование

Начиная с версии Oracle Database 10g* можно перевести БД в режим ARCHIVELOG командой:

```
ALTER DATABASE ARCHIVELOG
```

Если база данных работает в режиме ARCHIVELOG, то, заполнив очередной журнал, Oracle помечает его как подлежащий архивированию. Прежде чем заполненный журнал можно будет использовать повторно, его необходимо архивировать. Команда ALTER DATABASE ARCHIVELOG по умолчанию включает режим автоматического архивирования и запускает процессы-архиваторы.

До выхода версии Oracle Database 10g* пометка файла журнала как готового к архивированию еще не означала, что он будет архивирован автоматически. Необходимо было также задать параметр в файле инициализации:

```
LOG_ARCHIVE_START = TRUE
```

Только в этом случае запускался процесс копирования заполненного оперативного журнала в архивный.

Местоположение и формат имен архивных журналов определяются еще двумя параметрами: LOG_ARCHIVE_DEST и LOG_ARCHIVE_ FORMAT. Например, строка

```
LOG_ARCHIVE_DEST = C:\ORANT\DATABASE\ARCHIVE
```

определяет, в какой каталог Oracle будет записывать архивные журналы, а строка

```
LOG_ARCHIVE_FORMAT = ORCL%t_%s_%r.arc
```

описывает формат имен файлов архивных журналов. В данном случае имена начинаются с префикса ORCL и заканчиваются суффиксом .arc. В форматной строке распознаются следующие спецификаторы:

*%t*

Включать в имя файла номер цепочки журнальных файлов.

**%s**

Включать в имя файла порядковый номер журнала.

*%r*

Включать в имя файла идентификатор сброса порядкового номера журнала.

Если вы хотите, чтобы номер цепочки, порядковый номер журнала и идентификатор сброса номеров в именах архивных журнальных файлов дополнялись нулями, то следует записывать спецификаторы заглавными буквами, например:

```
LOG_ARCHIVE_FORMAT = "ORCL%T_%S_%R.arc"
```

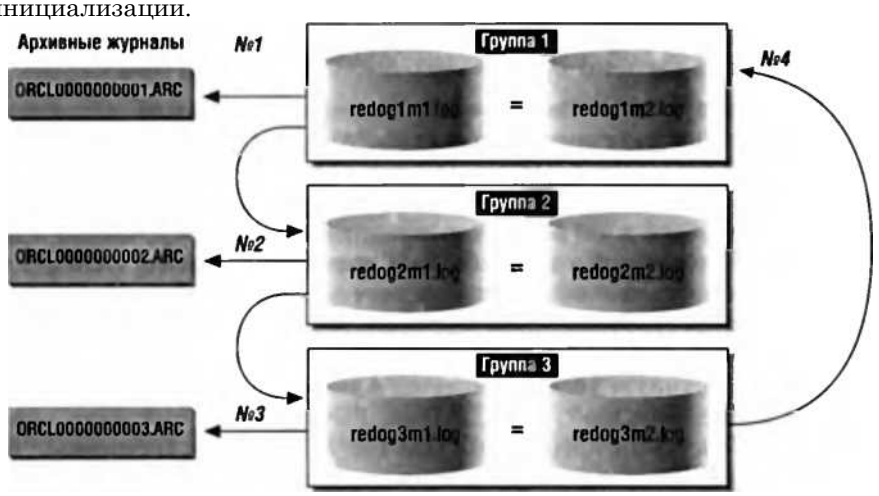
Поскольку файл инициализации считывается в момент запуска экземпляра Oracle, любые изменения вступят в силу только после останова и перезапуска экземпляра. Однако не забывайте, что включение режима автоматического архивирования еще не переводит базу данных в режим ARCHIVELOG. И

обратно, перевод базы данных в режим AR- CHIVELOG не включает режим автоматического архивирования.

Следует также убедиться, что в файловой системе достаточно места для размещения архивных журналов. Если эта файловая система переполнится, то Oracle зависнет, не имея возможности архивировать журнальные файлы.

На рис. 2.2 показан порядок использования журналов, когда архивирование включено.

Архивные журналы исключительно важны для восстановления базы данных. Как и для оперативных, для архивных журналов можно задать несколько мест расположения. В этом случае Oracle будет копировать заполненные журналы в указанные места. Можно также указать, следует ли ожидать удачного завершения копирования всех журналов. Эта функциональность управляется следующими параметрами инициализации.




При заполнении одного журнала и переходе к следующему заполненный журнал архивируется

***Рис. 2.2. Циклический перебор журналов с архивированием***

*LOG_ARCHIVE_DUPLEX_DEST*

Задает дополнительные места для копий архивных файлов.

*LOG_ARCHIVE_MIN_SUCCEED_DEST*

Определяет, должно ли успешно завершиться копирование всех или хотя бы нескольких копий заполненного журнала. Допустимы значения от 1 до 10, если используется мультиплексирование, и 1 или 2 в случае дуплексного режима.

Дополнительные параметры и представления, управляющие описанной функциональностью, описаны в документации Oracle.

## Память и процессы 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 является не одной большой однородной областью памяти, а состоит из различных компонентов, описанных в следующем разделе. Все процессы экземпляра, как системные, так и пользовательские, совместно обращаются к SGA.

До версии Oracle9i 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 Oracle9i можно изменять размер SGA и ее компонентов, не останавливая экземпляр. В Oracle9i также введено понятие *гранулы*, то есть наименьшего объема памяти, который можно добавить или удалить из SGA.

В версии Oracle Database 10g появился механизм автоматического управления разделяемой памятью (Automatic Shared Memory Management, ASMM), а в Oracle Databasellg- механизм автоматического управления памятью (Automatic Memory Management, AMM) для компонентов SGA и PGA. Если задан параметр инициализации MEMORY_ TARGET (появился в Oracle Database llg) или SGA_TARGET, то база данных автоматически распределяет память между различными компонентами SGA, обеспечивая оптимальное управление памятью. К автоматически распределяемым компонентам относятся разделяемый пул (его размер вручную устанавливается с помощью параметра SHARED_PO- OL_SIZE), большой пул (LARGE_POOL_SIZE), пул Java (JAVA_PO- OL_SIZE), кэш буферов (DB_CACHE_SIZE) и пул Streams (STREAMS_ POOL_SIZE). Параметры инициализации, относящиеся к автоматическому управлению памятью, можно задать в Oracle Enterprise Manager.

Фоновые процессы взаимодействуют с операционной системой и между собой, управляя структурами памяти экземпляра. Эти процессы также управляют собственно базой данных на диске и выполняют общие действия по обслуживанию экземпляра.
На рис. 2.3 показаны структуры памяти и фоновые процессы, рассматриваемые в следующем разделе.

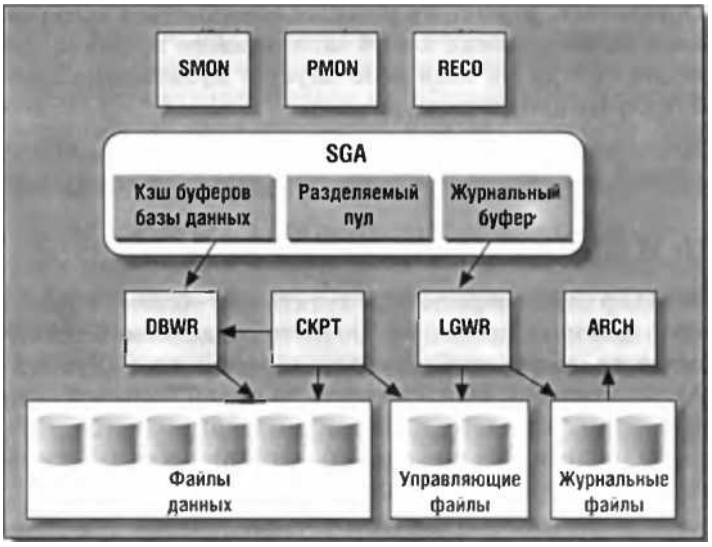



*Рис. 2.3. Экземпляр Oracle*

Если активированы еще какие-то компоненты СУБД, то могут работать и другие фоновые процессы, например: разделяемые серверы (до версии Oracle9i - Multi-Threaded Server, MTS), очереди заданий или репликация.

## Структуры оперативной памяти экземпляра

Как видно из рис. 2.3, системная глобальная область (SGA) состоит из нескольких областей. На рисунке показаны кэш буферов базы данных, разделяемый пул и журнальный буфер. Еще могут присутствовать пул Java, большой пул и пул Streams. Все эти области описаны ниже. Более подробно вопрос о связи SGA с производительностью рассмотрен в разделе «Как Oracle использует системную глобальную область» главы 7.

### Кэш буферов базы данных

В кэше буферов хранятся 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 (например, в результате недавно выполненного запроса), ее можно извлечь из оперативной памяти без затрат на обращение к диску. Oracle управляет кэшем на основе алгоритма LRU (Least Recently Used), когда выталкиваются элементы, не использовавшиеся дольше всех. Другими словами, если пользователь запрашивает данные, обращение к которым осуществлялось недавно, то с большой вероятностью они содержатся в кэше буферов и могут быть возвращены немедленно, без повторного считывания с диска.

Если же запрашивается блок, отсутствующий в кэше, то его необходимо считать и загрузить в кэш. Когда пользователь модифицирует данные в блоке, изменения сначала записываются в блок, находящийся в кэше, а спустя некоторое время сбрасываются в файл данных, которому принадлежит блок. Поэтому пользователю не нужно дожидаться завершения операции записи измененных блоков на диск.

Идея откладывания операций ввода/вывода до тех пор, пока это не станет абсолютно необходимо, красной нитью пронизывает всю СУБД Oracle. Диски - самый медленный компонент вычислительной системы, поэтому чем меньше объем ввода/вывода, тем быстрее работает система. Откладывая выполнение некритичных операций ввода/вывода, Oracle достигает более высокой производительности.

## Фоновые процессы экземпляра

Чаще всего используемые фоновые процессы показаны на рис. 2.3, хотя их состав меняется от версии к версии. Ниже описаны некоторые процессы.
*Процесс записи в базу данных (Database Writer, DBWn)*

Записывает блоки данных из кэша буферов SGA в файлы данных, расположенные на диске. Для одного экземпляра Oracle может существовать до 20 процессов DBW, обрабатывающих ввод/вывод вразличные файлы данных, отсюда и обозначение DBW*n*. Большинство экземпляров обходится одним процессом DBW. DBW записывает блоки из кэша на диск по двум основным причинам:

• Для установки контрольной точки (то есть для обновления блоков файлов данных, с тем чтобы они «догнали» журналы). Oracle записывает в журнал информацию, необходимую для повторного выполнения транзакции после ее фиксации, а позже сохраняет сами блоки. Периодически Oracle устанавливает контрольную точку для того, чтобы привести содержимое файла данных в соответствие с информацией, записанной в журнал для зафиксированных транзакций.

• Когда необходимо прочитать запрошенные пользователем блоки, а в кэше буферов не осталось свободного места. Тогда на диск сбрасываются блоки, которые дольше всего не использовались. Запись блоков именно в таком порядке позволяет минимизировать влияние отсутствия блоков в кэше на производительность.

*Процесс записи в журнал (Log Writer, LGWR)*
Записывает информацию из журнального буфера в SGA во все копии текущего журнального файла на диске. Пока транзакция обрабатывается, необходимая для ее повторного выполнения информация хранится в журнальном буфере в SGA. После фиксации транзакции Oracle отправляет эту информацию на постоянное хранение, вызывая процесс LGWR для ее записи на диск.

*Системный монитор (System Monitor, SMON)*
Отвечает за общую исправность и безопасность экземпляра. SMON восстанавливает экземпляр при запуске после сбоя. Он же координирует доступ и реализует восстановление экземпляра, когда к базе данных обращаются сразу несколько экземпляров, как бывает при работе с Real Application Clusters. SMON также приводит в порядок смежные области свободного пространства в файлах данных, объединяя их, и избавляется от пространства, используемого для сортировки строк, когда в нем уже нет необходимости.

*Монитор процессов (Process Monitor, PMON)*
Следит за пользовательскими процессами, обращающимися к базе данных. В случае аварийного прекращения пользовательского процесса PMON отвечает за очистку оставшихся занятыми ресурсов (таких как оперативная память) и за снятие всех блокировок, установленных сбойным процессом.

*Архиватор (Archiver,ARCH)*
Читает заполненные файлы журнала и копирует их в один или несколько архивных журналов. Поддерживается до 10 процессов архивации, которым присваиваются HMeHaARC0-ARC9. По мере необходимости LGWR запускает дополнительные архиваторы в зависимости от нагрузки. Максимально разрешенное количество процессов архивации задается параметром инициализации LOG_ARCHRTE_ MAX_PROCESSES. По умолчанию этот параметр равен 2, необходимость изменять это значение возникает редко.

*Контрольная точка (Checkpoint, CKPT)*
Обновляет заголовки файлов данных после установки контрольной точки.

*Процесс восстановления (Recover, RECO)*
Автоматически очищает неудавшиеся и отложенные распределенные транзакции.

*Диспетчер (Dispatcher)*
Эти необязательные фоновые процессы запускаются, если развернут разделяемый сервер.

*Служба глобального кэша (Global Cache Service, LMS)*
Управляет ресурсами, необходимыми Real Application Clusters, а также координирует использование ресурсов несколькими экземплярами.

*Очередь заданий (Job Queue)*
Служба выполнения команд и процедур PL/SQL по заданному расписанию.

*Монитор очередей (Queue Monitor, QMNn)*
Осуществляет мониторинг очередей сообщений подсистемы Oracle Streams. Поддерживается до 10 таких мониторов.

*Процессы подсистемы автоматического управления хранением (Automatic Storage Management,ASM)*

Процесс RBAL координирует перебалансировку работ для групп дисков. *ORBn* выполняет собственно перебалансировку. Процесс ASMB обеспечивает коммуникацию между базой данных и экземпляром ASM.

## Словарь данных

Во всех версиях Oracle присутствует набор *метаданных*, описывающих различные структуры данных, например определения таблиц иограничений целостности. Совокупность таблиц и представлений, в которых хранятся метаданные, называется *словарем данных* Oracle. Всем компонентам, которые рассматриваются в настоящей главе, соответствуют какие-то системные таблицы и представления в словаре данных, полностью описывающие характеристики компонента. К этим таблицам и представлениям можно обращаться с помощью стандартных команд SQL. Втабл. 2.1 показано, где в словаре данных можно найти информацию о каждом компоненте.

Таблицы словаря данных всегда находятся в табличном пространстве SYSTEM. Имена динамических таблиц начинаются с префиксов V$ или GV$ (данные в динамической таблице постоянно обновляются, отражая текущее состояние базы данных). Имена статических таблиц могут начинаться с одного из префиксов DBA_, ALL_ или USER_, обозначающего область видимости представленных в таблице объектов.

# Лекция 3. Установка и запуск Oracle

## Установка Oracle

Вплоть до версии Oracle8Z программа установки Oracle для UNIX поставлялась в текстовом и графическом вариантах. Графический инсталлятор использовал библиотеку Motif и работал под управлением системы X Windows. Для платформы Windows NT поставлялась только графическая версия. Начиная с версии Oracle8i инсталлятор переписан на языке Java.

Инсталлятор Oracle - одна из первых программ, демонстрирующих преимущества переносимости Java: он выглядит и работает одинаково во всех операционных системах. Установка Oracle стала довольно простым делом - пользователю надо лишь несколько раз щелкнуть мышью и ответить на вопросы о требуемых функциях и опциях.

Корпорация Oracle приложила немало усилий к тому, чтобы еще упростить установку в версии Oracle Database 10g. И эту версию, и Oracle Database *llg* можно установить меньше чем за 20 минут.

В процессе установки инсталлятор также запускает помощник конфигурирования сети Net Configuration Assistant и помощник конфигурирования базы данных Database Configuration Assistant, поэтому по завершении процедуры вы получаете работающий экземпляр Oracle.

Если по какой-то причине установка не завершена, то в файле протокола вы найдете команды, завершившиеся с ошибкой. Это поможет вам понять, в чем проблема, и после ее устранения выполнить оставшиеся команды самостоятельно.

Хотя процедура установки теперь одинакова для всех платформ, все же для каждой платформы есть особенности. Каждая версия Oracle Database Server поставляется со своим комплектом документации. Внего входит руководство по установке, замечания к версии (сюда включена информация, добавленная уже после публикации руководства по установке) и книга «Приступая к работе». Перед началом установки следует прочитать эти документы, поскольку в каждом имеются весьма ценные сведения об особенностях процедуры установки.

Например, вам предстоит заранее определиться с тем, где разместить корневой каталог Oracle и файлы данных. Эти вопросы подробно рассматриваются в документации. Помимо печатной документации на приобретенном вами носителе с программным обеспечением имеется документация в электронном виде. В ней вы найдете дополнительную информацию о СУБД и смежных продуктах.

Обычно руководство по установке вкладывается в ту же коробку, где находится компакт-диск. В руководстве приведены требования к системе (объем оперативной и дисковой памяти), действия, которые необходимо завершить до установки, указания по выполнению установки и замечания о переходе со старых версий Oracle на новую. Помните, что полная установка ПО не исчерпывается копированием программ на диск; необходимо еще сконфигурировать и запустить основные службы.

В старых версиях еще до начала установки Oracle вы должны были определиться со структурой каталогов и соглашением об именовании файлов, составляющих базу данных. Ясные, последовательные и хорошо спланированные соглашения - необходимое условие минимизации человеческих ошибок при администрировании операционной системы и базы данных. Ныне выбор имен в значительной степени автоматизирован. К наиболее важным решениям в этой области можно отнести:

- имена дисков или точек монтирования;
- структуры каталогов для ПО Oracle и файлов базы данных;
- имена файлов, составляющих базу данных: управляющих файлов, файлов данных и

журнальных файлов.

Основой соглашения об именовании всех этих файлов стал стандарт оптимальной гибкой архитектуры Optimal Flexible Architecture (OFA), описанный в следующем разделе.

## Стандарт Optimal Flexible Architecture (OFA)

Консультанты Oracle из крупных организаций разработали (по необходимости) полный набор стандартов, описывающих структуру каталогов и схему именования файлов, еще до того, как в Oracle появились автоматизированные процедуры установки. Этот набор стандартов называется *An Optimal Flexible Architecture for a Growing Oracle Database* (Оптимальная гибкая архитектура для растущей базы данных Oracle), или, как принято в сообществе Oracle, OFA. Например, в OFA четко прописано, как организовать развертывание на одной машине нескольких баз данных и нескольких версий Oracle. Даются рекомендации относительно точек монтирования, структуры каталогов, имен файлов итехники написания сценариев. Любой человек, знакомый с OFA, сможет быстро определить местоположение программных файлов Oracle и файлов, используемых базой данных и экземпляром. Стандартизация повышает продуктивность и помогает избежать ошибок.

Стандарты OFA были встроены в инсталлятор Oracle еще в версии Oracle7. Администраторам операционной системы и базы данных, работающим с Oracle, стоит ознакомиться с OFA, даже если система Oracle уже установлена. Документация по OFA включена в руководство по установке Oracle.

## Поддержка нескольких версий Oracle на одной машине

Можно установить и запустить несколько версий Oracle на одном сервере. Все продукты Oracle используют переменную окружения ORAC- LE_HOME, которая должна указывать на базовый каталог, в котором находится необходимое программное обеспечение. Поэтому, чтобы установить несколько версий Oracle на одной машине, достаточно сопоставить каждой свою переменную ORACLE_HOME. Для программы, которой необходима конкретная версия Oracle, следует просто задать нужное значение ORACLE_HOME.

Oracle поддерживает разные переменные ORACLE_HOME в системах UNIX и Windows путем использования разных каталогов. В OFA имеются четкие стандарты для реализации такой схемы.

## Переход на новую версию базы данных Oracle

В версии Oracle Database 10g появилось два дополнительных средства для перехода на новую версию уже установленной базы данных: помощник Database Upgrade Assistant и поддержка пошаговых обновлений.

Если вы хотите модернизировать СУБД, то можете воспользоваться помощником Database UpgradeAssistant, который запускается из Oracle Universal Installer. Начиная с версии Oracle Database llg Database Upgrade Assistant позволяет модернизировать бесплатную версию Oracle XE до полнофункциональной редакции СУБД.

Есть старая проблема перехода на новую версию: необходимо остановить базу данных, обновить программное обеспечение и заново запустить базу. При этом время простоя может оказаться неприемлемым в конкретных условиях эксплуатации. Если вы пользуетесь подсистемой Real Application Clusters, доступной с версии Oracle Database 10g, то можете выполнить *пошаговое обновление*. Эта технология позволяет остановить отдельные узлы кластера, обновить на них ПО, а затем снова запустить, включив в кластер. Ту же процедуру можно выполнить для всех остальных узлов. В результате все ПО Oracle будет модернизировано без прерывания работы базы данных.

# Создание базы данных

Мы уже отмечали, что Oracle можно установить для работы в условиях различных рабочих нагрузок. Для создания любой новой базы данных следует применять двухшаговую процедуру. Сначала выясните, для какой цели эта база будет использоваться, и только потом создайте ее с подходящими параметрами.

## Планирование базы данных

Как и при установке ПО Oracle, следует потратить некоторое время на то, чтобы понять назначение базы данных, прежде чем создавать ее. Изучите задачи, для решения которых предназначена база данных, и сколько данных в ней предположительно будет храниться. Следует знать свое оборудование - количество и типы процессоров, объем оперативной памяти,

количество дисков, контроллеров дисков и так далее. Поскольку база данных хранится на дисках, многих проблем можно избежать, если правильно спланировать емкость и другие характеристики подсистемы ввода/вывода.

Для планирования базы данных и необходимого аппаратного обеспечения нужно хорошо понимать объем рабочей нагрузки и тип выполняемых операций. Вот несколько вопросов, которые стоит себе задать.

*Сколько пользователей будут работать с базой данных?*

Сколько пользователей могут одновременно соединяться с базой? Сколько из них будут одновременно выполнять транзакции или предъявлять запросы?

*Должна ли база данных поддерживать OLTP-приложения (оперативную обработку транзакций) или хранилище данных?*

От ответа на этот вопрос зависят характер и интенсивность работы сервера базы данных. Например, в системах оперативной обработки транзакций обычно бывает довольно много пользователей, выполняющих небольшие транзакции, а для хранилищ данных характерно небольшое количество пользователей, предъявляющих сложные запросы.

*Каковы ожидаемые количество и размеры объектов в базе данных?*

Сколько места выделять для этих объектов изначально и каков ожидаемый темп прироста?

*Каковы характеристики доступа к различным объектам базы данных?*

Некоторые объекты используются чаще других. Понимание количества и типов различных операций критически важно для планирования и оптимизации базы данных. Иногда составляют так называемую CRUD-матрицу, содержащую индикаторы создания, чтения, обновления и удаления (Create, Read, Update, Delete), или даже оценивают, сколько операций будет выполняться для каждого из основных объектов, участвующих в бизнес-транзакциях. Это может быть оценка количества операций в минуту, в час, в день или в иной период времени, характерный для конкретной системы.

*Сколько оборудования имеется сейчас и сколько можно будет добавить по мере роста базы данных?*

Дисковые накопители постоянно дешевеют. Предположим, вы планируете базу данных с начальным объемом 100 Гбайт, которая в ближайшие два года может вырасти до 300 Гбайт. Возможно, уже сейчас имеется дисковое пространство для всех ожидаемых 300 Гбайт, однако более вероятно, что сначала закупается меньший объем, а новые диски добавляются позже. Важно спланировать начальное размещение базы, держа в уме ожидаемый рост.

До версии Oracle9i, если в процессе выполнения пакетной операции в табличном пространстве заканчивалось место, приходилось откатывать всю операцию. В Oracle9i появилось возможность возобновления функционирования после появления необходимого пространства (resumable space allocation). Если для выполнения операции не хватило места на диске и в данном сеансе включен режим возобновления, то выполнение приостанавливается на заданное время, что позволяет оператору устранить проблему. Можно даже создать триггер AFTER SUSPEND, который сработает, если операция была приостановлена.

Подсистема Automatic Storage Management (ASM), появившаяся в версии Oracle Database 10g, позволяет добавлять новые диски или извлекать имеющиеся, не пресекая доступ к базе данных. Это не устраняет необходимость тщательно оценивать требования к объему дисковой памяти, но плата за неправильное решение, выраженная в простоях, с ASM заметно уменьшилась.

*Каковы требования к доступности?*

Какие элементы необходимо предусмотреть для гарантии требуемой доступности (например, дополнительные дисковые накопители)? ASM поддерживает также автоматическое зеркалирование, помогающее повысить живучесть данных.

*Каковы требования к производительности?*

Каково время реакции, ожидаемое пользователями, и какое вы сможете обеспечить? Что является показателем производительности - время реакции (среднее, максимальное или в период пиковой нагрузки), общая пропускная способность или средняя нагрузка?

*Каковы требования к безопасности?*

Кто должен обеспечивать безопасность: приложение, операционная система или СУБД Oracle (или какая-то их комбинация)?

## Важность оценки

Даже если вы не уверены в каких-то параметрах, например в объеме или характеристиках использования, все же попытайтесь оценить начальные значения и темп роста и документируйте свои оценки. По мере развития базы данных вы сможете сравнить первоначальные оценки с новой информацией и откорректировать планы. Предположим, вы оценили начальный размер некоторой таблицы в 5 Гбайт с ежегодным приростом 3 Гбайт, но после запуска в эксплуатацию обнаружилось, что на самом деле размер таблицы - 3 Гбайт, а полгода спустя она выросла до 8 Гбайт. Теперь вы можете пересмотреть оценки с учетом более быстрого роста, предотвратив нехватку места. Сравнение реального размера, темпа роста и характеристик использования с первоначальными оценками поможет понять, как избежать проблем в будущем. Следовательно, документирование исходных предположений принесет пользу на более поздних этапах.

То же касается основных требований, например к доступности и производительности. Если точные требования неясны, выдвиньте какие- то гипотезы и задокументируйте их. Эти базовые требования сильно влияют на решения относительно избыточности и емкости. По мере эволюции системы требования становятся яснее, и прежняя история окажется весьма ценной для понимания того, почему был сделан тот или иной выбор и как действовать в будущем.

Автоматический репозиторий нагрузки (Automatic Workload Repository, AWR), впервые появившийся в версии Oracle Database 10g, хранит историю рабочей нагрузки и измерений производительности. Эти данные используются автоматическим диагностическим монитором базы данных (Automatic Database Diagnostic Monitor, ADDM) для выявления аномалий производительности. Вы тоже можете использовать AWR для наблюдения за происходящими изменениями.

## Инструменты создания баз данных

Есть два основных способа создать базу данных Oracle:

- воспользоваться графическим помощником Database Configuration Assistant;
- выполнить сценарии в командном режиме.

В комплект поставки Oracle входит графическая утилита Database Configuration Assistant, которую можно запустить автономно или из инсталлятора Oracle. Она написана на Java и потому выглядит одинаково на всех платформах. Помощник позволяет легко создать, модифицировать и удалить базу данных. Он умеет создавать как базу данных с типичными предопределенными параметрами (в этом случае вводить почти ничего не требуется), так и нестандартную базу (придется выбирать из нескольких вариантов и отвечать на дополнительные вопросы). Как правило, в первый раз утилита Database Configuration Assistant вызывается в ходе стандартной процедуры установки.

От типа базы данных зависят задаваемые по умолчанию значения параметров конфигурации.

Другой метод создания базы данных заключается в том, чтобы написать новый или отредактировать имеющийся SQL-сценарий для выполнения необходимых команд. У большинства администраторов баз данных Oracle есть любимый сценарий, который они изменяют по мере необходимости. В версиях Oracle7 и Oracle8 сценарий запускался спомощью командной утилиты Server Manager; начиная с версии Oracle8i можно использовать SQL*Plus. На дистрибутивном компакт- диске Oracle имеется пример сценария *BUILD_jDB.SQL*, описанный в документации.

# Конфигурирование Oracle Net

Подсистема Oracle Net (также известна под названиями Net8 для Oracle8 и Oracle8i и SQL*Net для более ранних версий) - это слой программного обеспечения, позволяющий разным физическим машинам обращаться к базе данных Oracle по сети.

Та или иная версия Oracle Net работает как на машине клиента, так и на сервере базы данных, позволяя клиентам и серверам обмениваться данными по сети. При этом поддерживаются практически все распространенные сетевые протоколы. Oracle Net также умеет транслировать протоколы. Например, клиент, работающий по протоколу LU 6.2, может взаимодействовать с сервером, понимающим протокол TCP/IP.

Кроме того, Oracle Net обеспечивает *прозрачность местоположения*, то есть клиентское приложение не обязано знать, где физически расположен сервер. Все необходимые согласования выполняет слой Oracle Net, то есть вы можете перенести базу данных на другую машину и просто изменить конфигурацию Oracle Net. Клиенты по-прежнему смогут найти базу данных без каких-либо изменений в приложениях.

Oracle Net поддерживает понятие *имени службы*, или *псевдонима*. Клиент сообщает псевдоним, чтобы идентифицировать базу данных, к которой хочет обратиться, не указывая реальный адрес машины или экземпляра. Зная имя службы, Oracle Net находит нужную машину и экземпляр, прозрачно направляя клиента к запрошенной им базе данных.

## Разрешение имен служб Oracle Net

Здесь перечислены варианты конфигурации Oracle Net для преобразования указанного клиентом имени службы в имена хоста и экземпляра, необходимые для доступа к базе данных.

*Локальное разрешение имен*

Для локального разрешения имен на каждой клиентской машине устанавливается файл *TNSNAMES.ORA*, в котором прописано соответствие между псевдонимами Oracle Net и парами (хост и экземпляр Oracle). Если местоположение базы данных изменяется, следует модифицировать эти файлы на всех клиентских машинах. Поскольку топология сети со временем почти всегда меняется, такой вариант затрудняет работу администратора. При использовании описанного ниже каталога Oracle Internet Directory файл *TNSNAMES.ORA* не нужен.

*Служба Oracle Names*

Служба Oracle Names, которая поддерживалась и в ранних версиях Oracle, устраняет необходимость размещать файл *TNSNAMES.ORA* на каждом клиенте. Это хорошо. А плохо то, что Oracle Names - нестандартное решение. Поскольку каталог Oracle Internet Directory основан на стандартах и предлагает ту же функциональность, служба Oracle Names объявлена устаревшей в версиях после Oracle9i.

*Oracle Internet Directory*

Потребность в централизованной службе имен выходит далеко за пределы Oracle. Есть фактический стандарт для размещения такого рода информации - протокол Lightweight Directory Access Protocol (LDAP). Начиная с версии Oracle Database llg в состав подсистемы Fusion Middleware включен каталог Oracle Internet Directory (OID). OID - это базирующийся на протоколе LDAP каталог, играющий ту же роль, что и прежняя служба Oracle Names. Он используется и для многих других целей, например для реализации единой точки входа в продукте Oracle Application Server Portal. Начиная с версии Oracle Database 10g из каталога разрешается экспортировать данные для создания локального файла *TNSNAMES.ORA*, который можно установить на клиентах, если те не используют каталог или если каталог недоступен.

*Именование хостов*

Клиент может просто указать имя хоста, на котором работает экземпляр. Это возможно в сетях TCP/IP, где имеется механизм преобразования имени хоста в IP-адрес. Например, служба доменных имен Domain Name Service (DNS) транслирует имя хоста в 1P-адрес аналогично тому, как Oracle Names транслирует имена служб. Начиная с версии Oracle Database 10g этот метод можно применять, указывая имя хоста (при необходимости вместе с доменом) или IP- адрес, но при этом не будут поддерживаться некоторые дополнительные возможности, например пул соединений.

*Сторонние службы разрешения имен*

Можно организовать интерфейс Oracle Net с внешними или службами разрешения имен и аутентификации сторонних фирм, например Kerberos или Radius. В этом случае может потребоваться компонент Oracle Advanced Security (до версии Oracle8i он назывался Advanced Networking Option).

Перечисленные варианты разрешения имен не являются взаимоисключающими. Например, можно совместно использовать каталог Oracle Internet Directory и механизм локального разрешения имен (файлы *TNSNAMES.ORA).* Нужно лишь определить в файле *SQLNET.ORA* порядок перебора вариантов (например, поискать имя в OID, а если оно там отсутствует, то в файле *TNSNAMES.ORA*). Это бывает полезно в тех случаях, когда для некоторых клиентов есть специальные службы баз данных. Тогда стандартные корпоративные службы, например электронная почта, могут пользоваться разрешением имен OID, а специфичные для конкретного клиента (например, база данных для разработки) - обращаться к файлу *TNSNAMES.ORA.*

Можно соединяться с базой данных Oracle и напрямую. Это так называемый *простой метод именования соединения*, когда задаются имя или IP-адрес сервера Oracle и имя экземпляра базы данных. Но этот способ применим только в сетях TCP/IP и рекомендуется лишь для сравнительно небольших организаций, где имена хостов меняются редко.

## Oracle Net Manager

В версии Oracle8 появилась графическая утилита Net8 Assistant для создания различных конфигурационных файлов, необходимых подсистеме Net8. В версии Oracle9i она получила название Oracle Net Manager.

Как и Database Configuration Assistant, программа Oracle Net Manager написана на Java и потому выглядит одинаково на любой платформе. Как правило, впервые она запускается из инсталлятора. Синтаксис конфигурационных файлов Oracle Net весьма специфичен, в них присутствуют многочисленные вложенные уровни, задаваемые скобками. Утилита Oracle Net Manager позволяет избежать ошибок, вероятных при ручном вводе таких файлов. Ее внешний вид для версии Oracle Database llg показан на рис. 3.1.

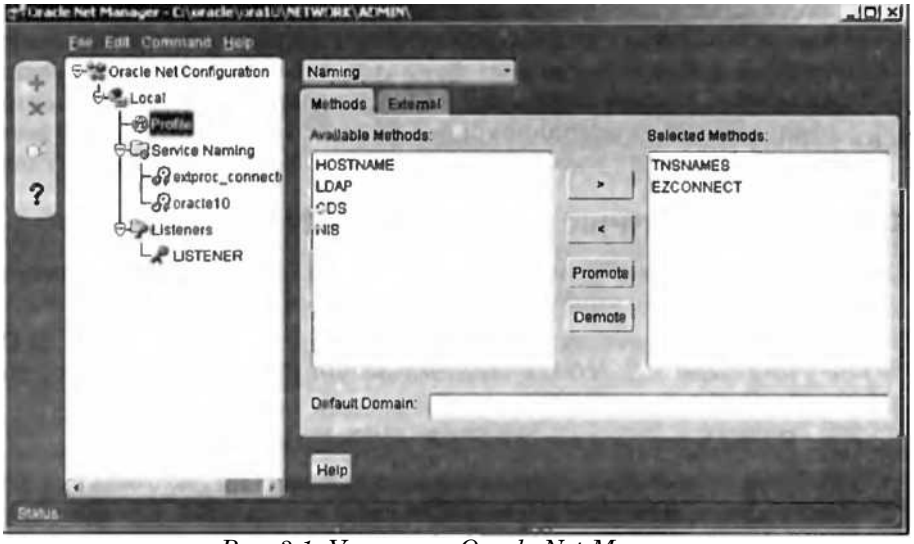



*Рис. 3.1. Утилита Oracle Net Manager*

### Отладка сетевых ошибок

Первое, что нужно сделать при возникновении сетевой ошибки, - убедиться, что конфигурационные файлы Oracle Net сгенерированы, а не введены вручную. Если сомневаетесь, скопируйте текущие конфигурационные файлы и заново сгенерируйте их с помощью Oracle Net Manager. При обращении в службу технической поддержки Oracle первое, о чем вас спросят, - созданы ли эти файлы вручную.

## Автоматическое обнаружение и агенты

Начиная с версии Oracle 7.3 в СУБД появились агенты автоматического обнаружения, позволяющие автоматически находить новые базы данных. Впоследствии поддержка этого механизма развивалась и совершенствовалась. В версии Oracle8i универсальный инсталлятор и Oracle Net Manager, работая совместно, автоматически конфигурируют подсистему Oracle Net.

Основной компонент сети Oracle, делающий автоматическое обнаружение возможным, называется Oracle Intelligent Agent. Он работает на той же машине, что и база данных, действуя как агент в интересах других функций, которым необходимо отыскать на этой машине базу данных и начать с ней взаимодействовать. Например, агент, зная о различных экземплярах Oracle, работающих на данной машине, выполняет важные административные функции, скажем, следит за возникновением определенных событий в базе данных и выполняет задания. Для обнаружения экземпляров и баз данных Oracle Net опрашивает агентов.

## Конфигурационные файлы Oracle Net

Для подсистемы Oracle Net необходимы несколько конфигурационных файлов. По умолчанию они находятся в следующих местах:

- В Windows это каталог *ORACLE_HOME\net80\admin* для Oracle8 или *ORACLE_HOME\network\admin* для Oracle8i и более поздних версий.
- В UNIX это каталог *ORACLE_HOME/network/admin*.

Можно поместить эти файлы и в другой каталог, но тогда следует указать его в значении переменной окружения TNS_ADMIN. Впрочем, при конфигурировании подавляющего большинства систем стандартное местоположение не изменяется.

Перечислим файлы, необходимые для простой конфигурации Oracle Net.

*LISTENER.ORA*

Содержит информацию о конфигурации прослушивателя Oracle Net Listener: какие экземпляры или службы обслуживает данный прослушиватель. Как и следует из названия, эта программа «прослушивает» сеть в ожидании входящих запросов о соединении от клиентов, желающих обратиться к базе данных.

*TNSNAMES.ORA*

Позволяет преобразовать имя службы в адрес машины и экземпляра Oracle для отправки запроса о соединении. (Если вы используете

описанную выше подсистему Oracle Names или ОШ, то файл *TNSNAMES.ORA* не нужен.) Этот файл - основа механизма прозрачности местоположения, применяемого Oracle Net. При переносе базы данных с одной машины на другую достаточно просто внести изменения в файлы *TNSNAMES.ORA* на разных клиентах, указав адрес новой машины для имеющегося имени службы. Предположим, клиенты обращаются к некоей базе данных по имени службы SALES. В файле *TNSNAMES.ORA* имеется запись о службе SALES, сопоставляющая ей машину с именем HOSTl и экземпляр PROD. Если перенести базу, с которой работает приложение SALES, на машину HOST2, то понадобится лишь обновить запись в *TNSNAMES.ORA*, указав в ней имя машины HOST2. После этого все запросы от клиентов будут прозрачно направляться на новую машину и модифицировать само приложение не придется.

*SQLNET.ORA*

Содержит важные умолчания и различные конфигурационные параметры, например, принимаемое по умолчанию доменное имя вашей сети.

*LDAP.ORA*

В версии Oracle8i и более поздних в этом файле хранятся параметры конфигурации, необходимые для работы с LDAP-каталогами, например Oracle Internet Directory. Здесь, в частности, задаются местонахождение LDAP-сервера и административный контекст для него, принимаемый по умолчанию. Начиная с версии Oracle Database 10g этот файл необязателен, если LDAP-сервер зарегистрирован на сервере доменных имен (DNS).

В версии Oracle9i появился файл параметров сервера *SPFILE,* где сохраняются изменения системных параметров, выполненные в процессе работы экземпляра Oracle с помощью команды ALTER SYSTEM. При следующем запуске экземпляра сохраненные значения этих параметров снова вступят в силу. Можно указать, должно ли изменение системного параметра быть постоянным (то есть сохраняться в *SPFILE)* или временным.

*SPFILE* - это двоичный файл, находящийся на серверной машине. Начиная с версии Oracle9i по умолчанию на этапе запуска экземпляр сначала ищет *SPFILE*, а потом файл *INIT.ORA.*

*SPFILE* может находиться и на разделяемом диске. Тогда при работе в кластерном режиме (Oracle Real Application Clusters) с его помощью можно инициализировать несколько экземпляров.

## Запуск СУБД

Запустить СУБД совсем не сложно. В Windows достаточно запустить службы Oracle (или сконфигурировать их так, чтобы они запускались на этапе загрузки операционной системы), а в UNIX и Linux - выполнить команду STARTUP в SQL*Plus или Enterprise Manager. Хотя со стороны кажется, что запуск СУБД - всего одно действие, на самом деле оно состоит из нескольких фаз. При запуске СУБД автоматически выполняются следующие действия:

1. *Запуск экземпляра*. Oracle считывает параметры инициализации экземпляра из файла *SPFILE* или *INIT.ORA* на сервере. Затем выделяется память для системной глобальной области и запускаются фоновые процессы экземпляра. В этот момент ни один из физических файлов базы данных еще не открыт, экземпляр находится в состоянии NOMOUNT. (Отметим, что в версиях Oracle Database 10g и Oracle Database llg заметно меньше параметров, которые должны быть обязательно определены в файле *SPFILE* на этапе начальной установки ПО.

При некоторых обстоятельствах запустить экземпляр не удается. Например, из-за ошибок в файле инициализации или потому, что операционная система не может выделить необходимый объем разделяемой памяти для SGA. Кроме того, для запуска экземпляра требуется особая привилегия SYSOPER или SYSDBA, предоставленная на уровне операционной системы или в файле паролей.

2. *Монтирование базы данных*. Экземпляр открывает управляющие файлы базы данных. Где их искать- сообщает параметр CONT- ROL_FILES. В этот момент открыты только управляющие файлы. Это состояние называется MOUNT, и в нем база данных доступна только администратору, который может выполнять с ней некоторые служебные операции. Например, администратор базы данных может переместить или переименовать один из файлов базы данных. Файлы данных,

перечисленные в управляющем файле, в состоянии MOUNT еще не открыты. Для переименования файла данных администратор может выполнить команду ALTER DATABASE, которая запишет в управляющий файл новое имя файла данных.

3. *Открытие базы данных*. Экземпляр открывает файлы журналов и файлы данных, пользуясь информацией, записанной в управляющем файле. В этот момент база данных наконец полностью открыта и доступна пользователям.

## Останов СУБД

Логично, что последовательность останова СУБД или пометки ее как недоступной противоположна рассмотренной выше:

1. *Закрытие базы данных*. Oracle сбрасывает на диск еще не записанные блоки данных из SGA, а также переписывает из журнального буфера в журналы информацию, необходимую для повторного выполнения. Затем Oracle записывает в файлы данных контрольную точку, отмечая, что их заголовки являются «текущими» на момент закрытия базы данных, после чего закрывает файлы данных и журналы. Начиная с этого момента, пользователи уже не могут обращаться к базе данных.
2. *Размонтирование базы данных*. Экземпляр Oracle размонтирует базу данных. В управляющих файлах помечается, что был выполнен корректный останов, после чего эти файлы закрываются. В этот момент сама база данных закрыта, остался лишь работающий экземпляр.
3. *Останов экземпляра*. Oracle останавливает фоновые процессы экземпляра и освобождает разделяемую память, занятую SGA.

В некоторых случаях (например, при сбое компьютера или после аварийного останова экземпляра администратором) корректного закрытия базы данных не происходит. Это означает, что у Oracle не было возможности сбросить модифицированные блоки из SGA в файлы данных. При следующем запуске экземпляр обнаружит, что имело место аварийное завершение и воспользуется хранящейся в журналах информацией для автоматического выполнения процедуры *восстановления после сбоя*. Гарантируется, что будут произведены все изменения, относящиеся к зафиксированным транзакциям, а изменения, относящиеся к незафиксированным или находившимся в процессе выполнения транзакциям, - отменены. Незафиксированные транзакции, выявленные после применения журналов, автоматически откатываются.

## Доступ к базе данных

В предыдущих разделах мы описали процедуры запуска и останова базы данных. Но база данных - это лишь одна из частей всей системы. Необходим еще клиент, обращающийся к базе данных, пусть даже его процесс работает на той же физической машине.

### Серверные процессы и клиенты

Для доступа к некоторой базе данных пользователь соединяется с экземпляром, предоставляющим доступ к ней. Программа, обращающаяся к базе данных, на самом деле состоит из двух частей - клиентской программы и серверного процесса, устанавливающего соединение с экземпляром Oracle. Например, при запуске текстовой утилиты SQL*Plus создаются два процесса:

- сам процесс SQL*Plus, выступающий в роли клиента;
- серверный процесс Oracle, иногда называемый *теневым процессом*, который обеспечивает соединение с экземпляром Oracle.

#### Серверный процесс

Серверный процесс Oracle всегда работает на том же компьютере, что и экземпляр. Серверный процесс присоединяется к разделяемой памяти, в которой находится SGA, так что может читать и записывать в нее данные.

Как и следует из его названия, серверный процесс обслуживает клиентский процесс - читает и возвращает запрашиваемые данные, выполняет изменения от имени клиента и так далее. Например, когда клиент хочет прочитать строку из конкретного блока базы данных, серверный процесс находит нужный блок и либо получает информацию из кэша буферов, либо считывает ее из соответствующего файла данных и помещает в кэш буферов. Далее, если пользователь требует внести изменения, серверный процесс модифицирует блок в кэше, а также порождает и сохраняет в журнальном буфере внутри SGA информацию, необходимую для повторного выполнения. Однако серверный процесс не записывает эту информацию в сами журнальные файлы и не сбрасывает модифицированные блоки из кэша буферов в файлы данных. Эти действия выполняют соответственно процессы Log Writer (LGWR) и Database Writer (DBWR).

### Клиентский процесс

Клиентский процесс может работать на одной машине с экземпляром или на отдельном компьютере. Оба компьютера соединены сетью, которая обеспечивает обмен данными между двумя процессами. В любом случае суть не меняется - во взаимодействии клиента с базой данных участвуют два процесса. Если оба процесса работают на одной машине, то Oracle применяет локальные механизмы межпроцессной коммуникации (Inter Process Communication, IPC); если на разных, то для коммуникации по сети задействуется подсистема Oracle Net.

## Серверы приложений и веб-серверы в роли клиентов

Выше мы постоянно пользовались терминами *клиент* и *сервер*. Но не следует думать, что СУБД Oracle построена исключительно на принципах архитектуры клиент/сервер. Корпорация Oracle одной из первых реализовала модель клиент-серверных вычислений, основанную на идее двух задач - клиента и сервера. Но при рассмотрении многоуровневых моделей, в которых участвуют веб-серверы и серверы приложений, понятие клиента смещается. «Клиентский» процесс теперь оказывается на промежуточном уровне, то есть на сервере приложений.

Логично считать любой процесс, соединяющийся с экземпляром Oracle, клиентом в том смысле, что он обслуживается СУБД. Но не путайте этого *так называемого* клиента с настоящим клиентом в многоуровневой конфигурации. Таковым является программа, реализующая интерфейс с пользователем, например Java-апплет, работающий в броузере.

Роль промежуточного слоя на платформе Oracle играет сервер приложений Oracle Application Server. Он интегрирован с СУБД Oracle и написан с применением тех же технологий.

На рис. показаны пользователи, соединяющиеся с экземпляром Oracle для доступа к базе данных, в двух- и трехуровневой конфигурации, с применением локальных или сетевых коммуникаций. Акцент здесь ставится на модель соединения с серверным процессом, а не на взаимодействие фоновых процессов. Слева изображена традиционная двухуровневая модель соединения клиента с сервером, справа - трехуровневая модель с участием сервера приложений, а в центре- локальное соединение. В двух- и трехуровневой модели для взаимодействия с базой данных необходима сеть, а для локального клиента достаточно механизма IPC.

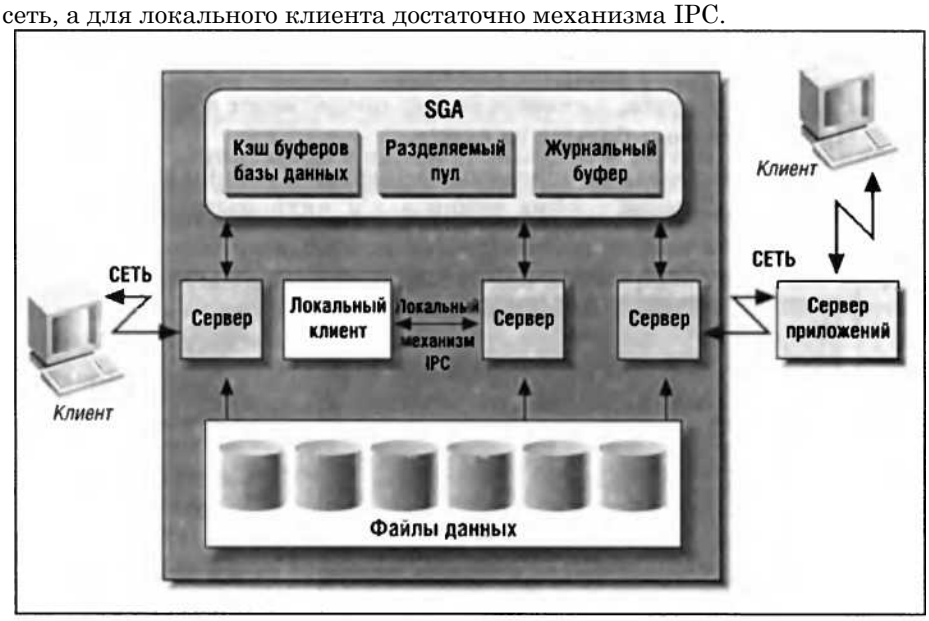



***Рис. 3.5. Доступ к базе данных***

## Oracle Net и установление сетевых соединений

Серверные процессы, показанные на рис. 3.5, соединяются с клиентскими процессами с помощью той или иной сети. Как же клиентский процесс устанавливает соединение с серверным процессом Oracle в самом начале работы?

«Сваха», которая «сводит» клиентские и серверные процессы в Oracle, называется Oracle Net Listener. Этот компонент «прослушивает» сеть в ожидании входящих запросов о соединении с экземплярами. Про- слушиватель не является частью экземпляра, он лишь направляет экземпляру запросы о соединении. Прослушиватель запускается и останавливается независимо от экземпляра. Если прослушиватель остановлен, а экземпляр запущен, то клиенты не смогут найти экземпляр в сети, поскольку их некому направить в нужное место. Если, напротив, прослушиватель запущен, а экземпляр остановлен, то клиентов некуда направлять.

Принцип работы прослушивателя довольно прост:

1. Клиент обращается к прослушивателю по сети.
2. Прослушиватель обнаруживает входящий запрос и «знакомит» клиента с серверным процессом.
3. Процедура «знакомства» сводится к информированию клиента и сервера о сетевых адресах друг друга.
4. Прослушиватель самоустраняется, позволяя клиенту и серверу общаться между собой напрямую.

Найдя друг друга, клиент и сервер продолжают общаться напрямую. Прослушиватель больше не нужен.

## Разделяемый/многопоточный сервер

Показанные на предыдущем рисунке серверные процессы являются *выделенными*, то есть каждый обслуживает только одного клиента. Следовательно, если с приложением одновременно работают 1000 клиентов, то с экземпляром Oracle будет связано 1000 серверных процессов. Каждый серверный процесс потребляет системные ресурсы, такие как память и время процессора. Для обслуживания множества пользователей может потребоваться очень много ресурсов. Ответом Oracle на все более насущные потребности в масштабировании стало появление в версии Oracle7 многопоточного сервера Multi-Threaded Server (MTS). Начиная с версии Oracle9i он называется *разделяемым сервером* (shared server).

Разделяемый сервер позволяет экземпляру Oracle обслуживать многих пользователей с помощью небольшого набора серверных процессов. Теперь каждому клиенту не сопоставляется отдельный выделенный серверный процесс - вместо этого клиенты пользуются разделяемыми серверами. Это позволяет существенно уменьшить общее потребление ресурсов при обслуживании большого числа пользователей.

Обычна ситуация, когда в определенные промежутки времени клиент не обращается активно к своему серверному процессу, например человек изучает данные, полученные из базы. В *модели с выделенным серверным процессом* последний все равно удерживает системные ресурсы, хотя ничего полезного не делает. В *модели с разделяемым сервером* этот сервер может использовать ресурсы для обслуживания других клиентских процессов.

Выбор той или иной модели не является взаимно исключающим. Oracle может одновременно поддерживать разделяемые и выделенные серверные процессы, а клиенты могут соединяться с любым из них. Все это прописывается в конфигурационных файлах Oracle Net, где одно имя службы может направлять клиента на выделенный сервер, адругое- на разделяемый. Синтаксис задания такой конфигурации описан в документации по Oracle Net.

Тип обслуживающего серверного процесса клиенту неизвестен. С точки зрения клиента, факт разделения серверного процесса скрыт «под капотом». Запросы о соединении с выделенными и многопоточными серверами обслуживаются одним и тем же прослушивателем.

Но шаги, предпринимаемые самим прослушивателем для установления соединения с разделяемым сервером, несколько отличаются. В этом случае появляются дополнительные фоновые процессы: диспетчеры экземпляра и собственно разделяемые серверы.

*Диспетчеры*

Вы уже знаете, что прослушиватель организует соединение между клиентом и сервером, а потом самоустраняется. С этого момента клиент полагается на то, что серверный процесс всегда готов ответить ему. Но разделяемый серверный процесс может в это время обслуживать другого клиента, поэтому клиент соединяется с диспетчером, готовым принять запрос от клиента в любой момент. Для каждого поддерживаемого протокола есть свой диспетчер (например, диспетчер для TCP/IP и так далее). Диспетчеры играют роль выделенных серверов для клиентов. Напрямую клиент соединяется не с сервером, а с диспетчером. Диспетчер принимает запросы от клиента и помещает их в очередь запросов - структуру, находящуюся в SGA. Для каждого экземпляра имеется только одна очередь запросов.

*Разделяемые серверы*

Разделяемый серверный процесс читает запрос из очереди, обрабатывает его и помещает результат в очередь ответов соответствующего диспетчера. Для каждого диспетчера имеется ровно одна очередь ответов. Диспетчер читает результаты из очереди и посылает их клиентскому процессу.

Есть пул диспетчеров и пул разделяемых серверов. В самом начале Oracle запускает столько тех и других, сколько указано в параметре инициализации SHARED_SERVERS. Это минимальное количество - Oracle может запустить и дополнительные разделяемые серверы, их общее число ограничено значением необязательного параметра инициализации

MAX_SHARED_SERVERS. Если для обработки возросшего потока запросов были запущены дополнительные процессы, а потом нагрузка снизилась, то Oracle будет постепенно уменьшать количество процессов, пока оно не достигнет нижнего порога - SHARED_SERVERS.

Следующее пошаговое описание показывает, чем отличаются процедуры установления соединения с разделяемым и выделенным серверными процессами.

1. Клиент обращается по сети к прослушивателю.
2. Прослушиватель обнаруживает входящий запрос и, анализируя конфигурацию Oracle Net, понимает, что запрос адресован многопоточному серверу. Поэтому он передает клиента не выделенному серверу, а диспетчеру того протокола, которым воспользовался клиент.
3. Прослушиватель «знакомит» клиента с диспетчером, сообщая каждому сетевой адрес другой стороны.
4. Теперь клиент и диспетчер знают, как найти друг друга, и далее общаются напрямую. Прослушиватель больше не нужен. Клиент посылает запросы напрямую диспетчеру.
5. Диспетчер помещает запрос клиента в очередь запросов в SGA.
6. Оказавшийся свободным разделяемый серверный процесс извлекает запрос из очереди и обрабатывает его.
7. Разделяемый сервер помещает результат обработки клиентского запроса в очередь ответов того диспетчера, которому изначально поступил запрос.
8. Диспетчер извлекает результат из очереди.
9. Диспетчер посылает результат клиенту.

### Память сеанса для разделяемых и выделенных серверных процессов

В Oracle есть понятие *памяти сеанса*, или *состояния сеанса*. Это основные данные, характеризующие текущее состояние сеанса Oracle. Например, в памяти сеанса хранится информация о командах SQL, выполненных в данном сеансе. В случае выделенного сервера состояние сохраняется в частной области памяти самого сервера. Это нормально, так как выделенный сервер обслуживает только одного клиента. Эта область называется программной глобальной памятью (Program Global Area, PGA).

Однако разделяемый сервер может работать от имени любого клиента, поэтому состояние сеанса нельзя хранить в PGA процесса разделяемого сервера. Доступ к состоянию сеанса должны иметь все серверы, поскольку сеанс может переходить от одного сервера к другому. Поэтому в данном случае Oracle хранит состояние сеанса в системной глобальной области (System Global Area, SGA).

Любой сервер может считывать данные из SGA. Размещение информации о состоянии в SGA позволяет обрабатывать разные запросы на разных серверах. Сервер, забравший из очереди запрос, просто считывает данные о состоянии сеанса из SGA, обновляет состояние в соответствии с результатами обработки и снова записывает его в SGA по завершении.

Для очередей запросов и ответов, а также для информации о состоянии необходима дополнительная память в SGA; в прежних версиях Oracle ее следовало выделять вручную. По умолчанию память для сеанса с разделяемыми серверами берется из разделяемого пула. Но можно сконфигурировать специальную область памяти для разделяемых серверов - *большой пул*. Применение большого пула позволяет избежать накладных расходов на координирование доступа к памяти для хранения разделяемых SQL-команд, кэша словаря данных и реализации других функций разделяемого пула. При выделении памяти из большого пула не возникает конкуренция с другими подсистемами за место и доступ к разделяемому пулу. Начиная с версии Oracle Database 10g* разделяемая память по умолчанию управляется автоматически. В версии Oracle Database 11 *g* автоматизировано также управление размером как SGA, так и PGA, если задан параметр инициализации MEMORY_TARGET.

## Oracle за работой

Чтобы вы лучше поняли совместную работу отдельных компонентов СУБД Oracle, в этом разделе мы рассмотрим пример последовательности шагов, выполняемых в ответ на запрос пользователя. Предположим, пользователь хочет добавить новую информацию в базу данных, то есть выполнить транзакцию.

## Oracle и транзакции

*Транзакцией* называется поступающий от клиента запрос на вставку, обновление или удаление данных. Команды, которые изменяют данные, составляют подмножество языка SQL, называемое *языком манипулирования данными* (Data Manipulation Language, DML). Транзакции должны обрабатываться так, чтобы гарантировать целостность данных.

*Транзакции - логически неделимая единица работы*

В терминах баз данных транзакцией называется логическая единица работы, состоящая из одного или нескольких изменений данных. В транзакцию могут входить несколько команд INSERT, UPDATE, DELETE, затрагивающих данные разных таблиц. Набор изменений либо успешно выполняется целиком, либо не выполняется вовсе. Транзакция начинается первой командой DML и заканчивается фиксацией или откатом.

*Фиксация или откат*

После того как пользователь передал данные для транзакции, он может либо *зафиксировать* (commit) ее, чтобы подтвердить изменения и сделать их постоянными, либо *откатить* (roll back), чтобы отменить изменения.

*Системный номер изменения (SCN)*

Ключом к сохранению целостности базы данных является информация о том, какая транзакция началась раньше. Например, если СУБД должна не дать более поздней транзакции случайно перепи

сать изменения, сделанные более ранней, то должна знать, какая из двух транзакций началась раньше. Для этого предназначен системный номер изменения (System Change Number, SCN), логическая временная метка, позволяющая отследить порядок возникновения событий. SCN применяется в Oracle также для реализации многоверсионной согласованности по чтению.

*Сегменты отката*

Сегменты отката - это структуры в базе данных Oracle, в которых хранится информация, необходимая для отмены изменений в случае отката транзакции. Они позволяют восстановить блоки данных в состоянии на момент начала транзакции. Если в ходе транзакции изменяются данные в каком-то блоке, то сначала старый образ данных записывается в сегмент отката. Информация сегментов отката служит для двух целей: обеспечить возможность отката транзакции и поддержать модель многоверсионной согласованности по чтению.

Сегмент отката - не то же самое, что журнал. В журнале протоколируются все транзакции, имевшие место в базе данных, и используется он для восстановления базы после сбоя. А сегменты отката обеспечивают откат транзакций и согласованность по чтению.

Блоки сегментов отката кэшируются в SGA наряду с блоками таблиц и индексов. Если в течение определенного промежутка времени блок сегмента отката не используется, то он может быть вытолкнут из кэша и записан на диск.

*Быстрая фиксация*

Поскольку при каждой фиксации транзакции информация записывается в журнал, журналами можно воспользоваться для ускорения операций в базе данных. Когда пользователь фиксирует транзакцию, Oracle может записать изменения на диск одним из двух способов:

- записать все измененные в ходе транзакции блоки в соответствующие файлы данных;
- записать только журнальную информацию; как правило, объем ввода/вывода в этом случае гораздо меньше. Если впоследствии произойдет сбой, то изменения можно будет воспроизвести.

Чтобы обеспечить максимальную производительность без риска для транзакционной целостности, Oracle записывает только журнальную информацию. В момент фиксации транзакции гарантируется, что вся информация, необходимая для повтора произведенных изменений, находится в журналах на диске. Сами же измененные блоки данных могут быть помещены в файлы данных позже. Если сбой произойдет до сброса измененных блоков из кэша на диск, то с помощью журналов можно будет воспроизвести изменения во всей полноте. Поскольку физический диск - самая медленная часть вычислительной системы, такой механизм быстрой фиксации минимизирует расходы на фиксацию транзакции, обеспечивая максимальную производительность без угрозы целостности.

## Ретроспекция

В Oracle9i сегменты отката также применялись для реализации так называемых *ретроспективных запросов* (Flashback Query). Напомним, что сегмент отката служит для получения согласованного образа данных в предшествующий момент времени. Механизм ретроспективных запросов позволяет запросить у Oracle результаты SQL-запроса в конкретный момент времени. Например, можно запросить данные в том виде, в каком они были два часа назад. Эта возможность основана на механизмах отката, которые и так уже являются частью базовой архитектуры Oracle.

Для ретроспекции применяются сегменты отката, поэтому углубляться в прошлое можно не дальше, чем позволяет информация, хранящаяся в существующих сегментах отката. Это не слишком длительный период - обычно не удается заглянуть на несколько дней назад, поскольку данные для отката так долго не хранятся. Но, несмотря на это ограничение, бывают ситуации, когда ретроспективный запрос оказывается очень полезен, например, если нужно вернуться к моменту, который предшествует ошибке пользователя, приведшей к потере данных.

Со временем в СУБД Oracle появились дополнительные возможности ретроспекции. В Oracle Database 10# этот механизм был значительно расширен за счет добавления:

- ретроспективной базы данных (Flashback Database), то есть возможности откатить всю базу данных в согласованное состояние;
- ретроспективной таблицы (Flashback Database) для отката одной таблицы;
- ретроспективного удаления (Flashback Drop) для отката операции DROP;
- запроса к ретроспективным версиям (Flashback Versions Query) для возврата изменений в одной или нескольких строках.

В Oracle Databasellg развитие продолжилось - добавилась возможность ретроспективных транзакций (Flashback Transaction), которой можно воспользоваться для обращения эффекта некоторой транзакции и всех транзакций, зависящих от нее.

## Транзакция – шаг за шагом

В этом простом примере иллюстрируется вся процедура выполнения транзакции. Мы взяли таблицу EMP с данными о работниках; она является частью демонстрационной схемы, традиционно входящей в комплект поставки СУБД Oracle. Будем считать, что сотрудник отдела кадров хочет изменить фамилию работника. Для этого он считывает данные о работнике из базы, изменяет фамилию и фиксирует транзакцию.

Предполагается, что обновить информацию в данной строке пытается только один пользователь. Поэтому мы не включили в рассмотрение шаги, которые Oracle предпринимает для защиты от изменений со стороны других пользователей.

Сотрудник отдела кадров видит запись о работнике у себя на экране, то есть блок базы данных, содержащий данную строку, уже находится в кэше буферов. Начиная с этого момента последовательность шагов выглядит так:

1. Пользователь исправляет фамилию работника на экране, и приложение посылает SQL-команду UPDATE серверному процессу по сети.
2. Серверный процесс ищет такую же команду в разделяемой области SQL в разделяемом пуле. Если найдет, то использует ее повторно. В противном случае производится синтаксический анализ команды и определяется оптимальный способ ее выполнения. Этот шаг называется *синтаксическим анализом и оптимизацией*. После завершения этого шага команда кэшируется в разделяемой области SQL.
3. Серверный процесс копирует старый образ данных о работнике, которые предстоит изменить, в сегмент отката и в сегмент журнала. Изменения в сегменте отката также протоколируются в журнале.

   На первый взгляд, это кажется странным, но вспомните, что в журнал попадают *все* изменения, произведенные в результате транзакции. Содержимое сегмента отката изменилось, потому что в него были записаны старые данные о работнике, необходимые для отмены. Это изменение содержимого сегмента отката является частью транзакции и потому записывается в журнал для данной транзакции.
4. Завершив эту работу, серверный процесс модифицирует блок базы данных, записывая в него измененную фамилию работника. Пока этот блок еще находится в кэше буферов.
5. Сотрудник отдела кадров фиксирует транзакцию.
6. Процесс Log Writer (LGWR) переписывает всю информацию о транзакции из журнального буфера в текущий журнальный файл на диске. Когда операционная система подтвердит, что запись в журнальный файл успешно завершилась, транзакция считается зафиксированной.
7. Серверный процесс посылает сообщение клиенту, подтверждая фиксацию.

Пользователь мог бы не зафиксировать, а откатить транзакцию, тогда серверный процесс воспользовался бы старым образом данных из сегмента отката, чтобы отменить внесенные в блок базы данных изменения.

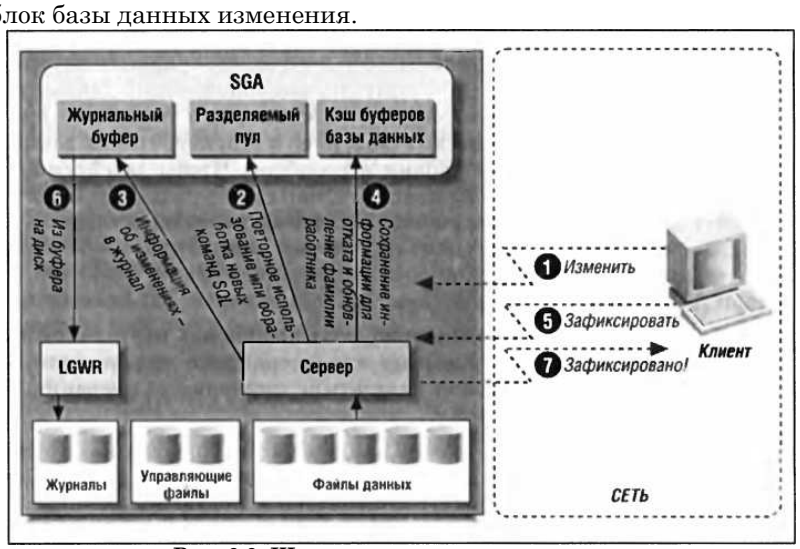



***Рис. 3.3. Шаги выполнения транзакции***

# Лекция 4. Структуры данных Oracle

## Типы данных

Тип данных - это один из атрибутов *столбца* или переменной в хранимой процедуре. Он описывает характеристики информации, хранящейся в столбце, и может налагать ограничения на операции, применимые к данному столбцу.

Типы данных в Oracle можно разбить на три больших категории: символьные, числовые и прочие. При создании столбца в таблице можно задать любой тип данных, как показано в следующей команде:

```
CREATE SAMPLE_TABLE( char_field CHAR(10),
    varchar_field VARCHAR2(10), todays_date DATE)
```

Типы данных также применяются при определении переменных в PL/ SQL-процедуре.

## Символьные типы

*Символьные типы данных* служат для хранения строковых значений, в том числе представлений числовых значений в виде строк. При попытке записать в столбец строку, длина которой больше указанной или допустимой для символьного типа, возникает ошибка времени выполнения. К стандартным (не длинным) символьным типам применимы строковые функции, такие как UPPER, LOWER, SUBSTR и SOUNDEX.

Есть несколько символьных типов данных.

*CHAR*

Позволяет хранить символьные значения фиксированной длины от 1 до 2000 символов. Если длина явно не указана, она предполагается равной 1. Если длина присваиваемого значения меньше указанной в определении типа CHAR, то Oracle автоматически дополнит его пробелами. Вот несколько примеров значений типа CHAR:

```
CHAR(10) = "Rick ", "Jon                    ", "Stackowiak"
```

*VARCHAR2*

Предназначен для хранения символьных строк переменной длины. В определении этого типа можно указать длину, но она интерпретируется как максимальная. Значения, присваиваемые столбцу или переменной типа VARCHAR2, не дополняются пробелами. Максимальная возможная длина типа VARCHAR2 - 4000 символов. Для хранения данных типа VARCHAR2 обычно требуется меньше места, чем для данных типа CHAR, поскольку сохраняются только значимые символы, записанные в столбец.

В версиях начиная с Oracle8 типы данных VARCHAR и VAR- CHAR2 - синонимы, однако рекомендуется пользоваться типом VARCHAR2 ввиду возможных будущих изменений. Если приведенные выше значения присвоить столбцу типа VARCHAR2, получится следующий результат:

```
VARCHAR2(10) = "Rick", "Jon", "Stackowiak"
```

*NCHAR и NVARCHAR2*

Содержат символьные данные соответственно фиксированной или переменной длины, представленные в кодировке, отличной от используемой в остальных частях базы данных. Кодировка задается при создании базы. Можно также указать дополнительную кодировку (эту возможность обеспечивает подсистема National Language Set (NLS). Эта дополнительная кодировка и используется для столбцов типа NCHAR и NVARCHAR2. Например, у вас могут быть поля, содержащие японские символы, в то время как вся остальная информация БД хранится в английской кодировке. Тогда при создании БД необходимо в качестве дополнительной указать кодировку, поддерживающую японские символы, а указанным столбцам назначить тип NCHAR или NVARCHAR2.

Начиная с версии Oracle9i можно указать, что размер столбцов типа NCHAR и NVARCHAR2 должен измеряться в символах, а не в байтах. Таким образом, если вы зададите 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 из ранних версий СУБД Oracle. Сейчас для хранения символьных данных большого объема Oracle рекомендует применять типы CLOB и NCLOB. На использование типа LONG в таблицах и запросах накладывается много ограничений, например, нельзя использовать его в конструкциях WHERE, GROUP BY, ORDER BY и CONNECT BY, а также в командах SQL с квалификатором DISTINCT. По столбцу типа LONG также нельзя создать индекс.

*CLOB и NCLOB*

До версии Oracle Database 10g* эти типы предназначались для хранения символьных данных размером до 4 Гбайт. А в этой версии объем может достигать 128 терабайт в зависимости от размера блока базы данных. Тип NCLOB позволяет хранить данные в кодировках, поддерживаемых NLS. Начиная с версии Oracle Database 10g между типами CLOB и NCLOB производится неявное преобразование.

## Числовые типы

В СУБД Oracle числа хранятся в стандартном внутреннем формате переменной длины, обеспечивающем точность до 38 разрядов.

Единственный числовой тип данных, поддерживаемый Oracle, - это NUMBER. Тип NUMBER, заданный для столбца или переменной, ав

томатически обеспечивает 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. Если разрядность явно не указана, по умолчанию подразумевается значение 38. *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. Например, в результате выполнения такого кода:

column_round NUMBER(10,-2) column_round = 1,234,567

в столбец `column_round` будет записано значение 1 234 600.

Для хранения числовых данных в Oracle используется только тип NUMBER. Все определенные в стандарте ANSI типы данных: DECI- MAL/DEC, NUMBER, INTEGER/INT, SMALLINT, DOUBLE PRECISION и REAL представлены в базе данных типом NUMBER. В языке или продукте, которым вы пользуетесь для доступа к данным в БД Oracle, эти типы могут поддерживаться, но хранятся они все равно в столбцах типа NUMBER.

В версии Oracle Database 10g* добавлены типы BINARY_FLOAT и BI- NARY_DOUBLE, поддерживающие разрядность, определенную в стандарте IEEE 754-1985. В версии Oracle Database *llg* введена поддержка типа SIMPLE_INTEGER.

## Тип Date

Как и в случае типа NUMBER, дата и время хранятся во внутреннем формате. При вводе дат подразумевается формат DD-MON-YY HH:MI: SS, где DD - двузначный день месяца, MON - трехзначное сокращенное название месяца, YY - двузначный номер года, а HH, MI и SS - двузначное представление часа, минуты и секунды соответственно. Если время не указано, по умолчанию подразумеваются нули.

Изменить формат ввода даты для конкретного экземпляра позволяет параметр NLS_DATE_FORMAT. Можно сделать это и для одного сеанса с помощью SQL-команды ALTER SESSION или в конкретном запросе, воспользовавшись встроенной функцией TO_DATE.

В Oracle SQL поддерживаются арифметические операции над датами, 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 опера- цийсдатами:

12_DEC_07 + 10 = 22_DEC_07
31_DEC_07:23:59:59 + .25 = 1_JAN_2008:5:59:59

Начиная с версии Oracle9i, поддерживаются два интервальных типа данных: INTERVAL YEAR TO MONTH и INTERVAL DAY TO SECOND для хранения промежутков времени. Значения этого типа можно применять в арифметических операциях с датами.

## Прочие типы данных

Помимо основных типов, предназначенных для хранения символов, чисел и дат, Oracle поддерживает ряд специализированных типов данных.

*RAWuLONG RAW*

Обычно сервер БД Oracle не только хранит, но и интерпретирует данные. Когда данные запрашиваются или экспортируются из базы, может потребоваться то или иное преобразование. Например, при выводе данных из столбца типа NUMBER в файл записываются внешние представления чисел, а не значения во внутреннем формате.

Типы данных RAW и LONG RAW позволяют предотвратить интерпретацию со стороны Oracle. Если указывается один из таких типов, то Oracle сохраняет данные в виде именно той последовательности битов, которая была ему предъявлена. Типы RAW обычно применяются для хранения объектов в характерном для них внутреннем формате, например растровых изображений. Тип RAW позволяет сохранить до 2 Кбайт, а тип LONG RAW - до 2 Гбайт.

*ROWID*

ROWID - это специальный тип столбца, называемый *псевдостолбцом* (pseudocolumn). К псевдостолбцу ROWID можно обращаться так же, как к обычному столбцу, в SQL-команде SELECT. Псевдостолбец ROWID есть в каждой строке БД Oracle и представляет собой адрес этой конкретной строки. Псевдостолбец ROWID имеет тип ROWID.

ROWID устанавливает связь с конкретным местоположением на диске, следовательно, это самый быстрый способ выборки отдельной строки. Однако ROWID для

строки может измениться в результате записи дампа базы данных или ее перезагрузки. Поэтому рекомендуем использовать значение ROWID только в пределах одной транзакции. Например, не стоит сохранять ROWID строки, закончив работу с ней в приложении.

Задать значение ROWID при помощи команды SQL невозможно.

В версии Oracle8 формат псевдостолбца ROWID изменился. Теперь он включает не только имя файла данных, номер блока и строки, но также идентификатор объекта в базе данных. Чтобы понять, как физически хранятся строки в базе данных, вы можете разобрать значение, полученное из псевдостолбца ROWID.

Разрешается определить столбец или переменную типа ROWID, но Oracle не гарантирует, что значение, помещенное в такую переменную или столбец, будет корректным идентификатором ROWID.

*ORA_ROWSCN*

Начиная с версии Oracle Database 10g поддерживается псевдостолбец ORA_ROWSCN, где хранится системный номер изменения (SCN), соответствующий последней транзакции, в которой строка была изменена. Этот псевдостолбец позволяет легко проверить, была ли строка модифицирована с момента начала транзакции.

*LOB*

Тип данных больших объектов (LOB) позволяет хранить данные объемом до 4 Гбайт. Есть три разновидности типа LOB:

- CLOB для хранения символьных данных
- NCLOB для хранения символьных данных в национальной кодировке
- BLOB для хранения двоичных данных

Можно указать, должны ли данные типа LOB храниться в самой базе или во внешнем файле, на который указывает значение в столбце.

Данные типа LOB могут участвовать в транзакциях. При выборке таких данных Oracle возвращает указатель. Для манипуляции самими данными, хранящимися в столбце типа LOB, можно воспользоваться встроенным PL/SQL-пакетом DBMS_LOB или интерфейсом уровня вызовов OCI.

Для преобразования данных типа LONG в LOB в версию Oracle9i поддержка LOB включена в большинство функций, предназначенных для работы с типом LONG. Кроме того, в команду ALTER TABLE добавлена возможность автоматического преобразования типа LONG в LOB.

*BFILE*

Тип данных BFILE - это указатель на файл, хранящийся вне базы данных Oracle. Поэтому столбцы и переменные типа BFILE не участвуют в транзакциях, а хранящиеся в них данные доступны только для чтения. Объем данных, которые можно сохранить в столбце типа BFILE, определяется ограничением операционной системы на размер файла.

*XMLType*

В рамках поддержки XML в версии Oracle9i появился тип данных XMLType. В столбце такого типа можно хранить XML-документ как большой символьный объект. Соответствующие встроенные функции позволяют извлекать из документа отдельные узлы. Кроме того, по любому узлу документа типа XMLType можно строить индексы.

*Типы данных, определяемые пользователем*

Начиная с версии 0racle8 пользователи могут создавать собственные составные типы данных, комбинируя описанные выше стандартные типы Oracle. Кроме того, разрешается создавать объекты, состоящие из стандартных и пользовательских типов данных.

*AnyType, AnyData, AnyDataSet*

В версию Oracle9i включены три новых типа данных, позволяющие явно определять структуры, которые невозможно представить ни одним из существующих типов. Эти типы должны быть определены в программных модулях, обучающих Oracle обрабатывать данные соответствующего типа.

## Преобразование типов

Oracle автоматически преобразует некоторые типы в другие в зависимости от контекста SQL, в котором встречается значение.

Если присвоить символьное значение числовому типу данных, Oracle выполнит неявное преобразование символьной строки (в кодировке ASCII) в число. Например, автоматическое преобразование будет иметь место при записи строки 10 в столбец типа NUMBER.

Попытка присвоить объекту числового типа символьное значение приведет к неожиданному (и неверному) результату, поэтому обращайте внимание на корректность присваивания значений.

В Oracle также имеется немало функций для явного преобразования данных. Явные преобразования лучше применять, если заранее известно, что преобразование неизбежно. Такое применение документирует факт преобразования и не пройдет незамеченным, как бывает в случае неявных преобразований.

## Конкатенация и сравнение

На большинстве платформ оператор конкатенации в диалекте Oracle SQL обозначается двумя символами вертикальной черты (||). Конкатенация применяется к двум символьным значениям. Механизм автоматического преобразования типов позволяет конкатенировать и два числовых значения. Если NUM1 - числовой столбец, содержащий 1, NUM2 - числовой столбец, содержащий 2, а NUM3 - числовой столбец, содержащий 3, то истинны следующие выражения:

```
NUM1 || NUM2 || NUM3 = "123" NUM1 || NUM2 + NUM3 = "15" (12
+ 3)
NUM1 + NUM2 || NUM3 = "33" (1 + 2 || 3)
```

Результатом вычисления каждого выражения является символьная строка, но она может быть автоматически преобразована обратно в число, если это необходимо для последующих вычислений.

Сравнение значений одного типа не приносит неожиданностей. Например, более поздняя дата считается больше, чем более ранняя, а ноль и любое положительное число - больше любого отрицательного. Операторы сравнения можно применять для сравнения дат и чисел. При сравнении символьных значений отдельные символы сравниваются в соответствии с кодовой страницей, в которой представлены символы. Строки, состоящие из нескольких символов, сравниваются до тех пор, пока не обнаружатся отличающиеся символы.

При сравнении строк различной длины Oracle применяет две разных семантики сравнения: *сравнение с дополнением пробелами* и *сравнение без дополнения*. В первом случае более короткая строка дополняется пробелами, и дальше сравнение производится, как описано выше. Во втором случае, если первые *N* символов (где *N* - длина более короткой строки) обеих строк совпадают, то более короткая строка считается меньшей. Например, при сравнении с дополнением пробелами строки `"A "` (заглавная буква А, после которой следует пробел) и `"A"` (только заглавная А) считаются совпадающими, поскольку вторая строка будет дополнена пробелом. При сравнении же без дополнения вторая строка считается меньшей, так как она короче первой. Сравнение без дополнения применяется в случаях, когда хотя бы одно значение имеет тип VARCHAR2 или NVARCHAR2, а сравнение с дополнением пробелами - во всех остальных случаях.

В версии Oracle Database 10g* появился механизм фильтрации выражений (Expression Filter), позволяющий сохранять сложные выражения, включающие сравнение, в строке таблицы. В запросе можно воспользоваться функцией EVALUATE, чтобы учесть при отборе результат вычисления таких выражений.

## Неопределенное значение (NULL)

«Значение» NULL - одна из ключевых характеристик любой реляционной СУБД. По существу, NULL означает отсутствие какого-либо значения. Если в некотором столбце таблицы значение должно присутствовать обязательно, то для нее указывается атрибут NOT NULL, означающий, что значение NULL недопустимо. При попытке записать в базу данных строку, в которой столбцу с атрибутом NOT NULL не присвоено значение, Oracle вернет сообщение об ошибке.

Разрешается присвоить NULL в качестве значения любого типа данных. Наличие NULL привносит в команды SQL так называемую *трехзначную логику*. При обычном сравнении возможно только два результата: TRUE или FALSE. Если же сравниваемые величины могут принимать значение NULL, то логически исходов три: TRUE, FALSE или ни то ни другое.

Ни одно из следующих условий не является истинным, если столбец A содержит NULL

## Основные структуры данных

В этом разделе описаны три основные структуры данных Oracle - таблицы, представления и индексы. Здесь же обсуждается механизм секционирования, влияющий на способ хранения таблиц и индексов.

## Таблицы

*Таблица* - это базовая структура данных в любой реляционной СУБД. Таблица представляет собой набор строк. Каждая *строка* таблицы состоит из одного или нескольких *столбцов*. Если вы незнакомы с реляционными базами данных, то можете считать таблицу аналогом файла, а строку - аналогом записи в нереляционной базе.

В версии Oracle9i появились *внешние таблицы*. Как легко понять, данные внешней таблицы хранятся вне базы данных, обычно в плоском (неструктурированном) файле. Внешняя таблица допускает только чтение, обновить находящиеся в ней данные нельзя. Кроме прочего, внешняя таблица удобна для загрузки данных из файла или для выгрузки в файл.

В Oracle Database 11 *g* добавлена возможность создавать в таблице виртуальные столбцы. Они определяются выражением и, хотя результаты вычисления выражения не хранятся, приложение может обращаться к таким столбцам.

## Представления

*Представлением* (view) в Oracle называется структура данных, определяемая SQL-командой. Эта SQL-команда хранится в базе данных. Когда в запросе встречается представление, выполняется определяющий его запрос и пользователю возвращаются данные из базовой таблицы. Сами представления не содержат данных, а лишь позволяют взглянуть на них так, как это определено в запросе.

Представления можно использовать в разных целях:

- Чтобы упростить доступ к данным, хранящимся в разных таблицах.
- Чтобы реализовать специальные требования к безопасности данных в таблице (например, создав представление с предложением WHERE, ограничивающим набор данных, доступных через это представление). Начиная с версии Oracle9i эту задачу можно решить с помощью механизма *детального контроля доступа* (finegrained access control), позволяющего автоматически ограничивать доступ к данным в зависимости от значения, хранящегося в строке.
- Чтобы скрыть от приложения точную структуру базовых таблиц.

Представление строится над набором *базовых таблиц*, каждая из которых может быть либо реальной таблицей базы данных, либо другим представлением. Если изменить любую базовую таблицу, так что она перестанет соответствовать определению представления, то само представление окажется непригодным для использования.

В общем случае в одной команде SQL допускается запись в столбец только одной какой-то базовой таблицы, указанной в определении представления. Есть дополнительные ограничения на операции INSERT, UPDATE и DELETE. Кроме того, при наличии в SQL-команде некоторых предложений обновить данные представления вообще невозможно.

Осуществить запись в необновляемое представление позволяет триггер INSTEAD OF.

В версии Oracle8i появились *материализованные представления*. По существу, это не представления в описанном выше смысле, а физические таблицы, в которых хранятся заранее агрегированныеданные, что позволяет заметно повысить производительность хранилищ данных.

## Индексы

*Индекс* - это структура данных, ускоряющая доступ к строкам базы данных. Индекс ассоциируется с одной таблицей и содержит данные из одного или нескольких столбцов.

Базовый синтаксис SQL для создания индекса:

```
CREATE INDEX emp_idx1 ON emp (ename, job);
```

Здесь `emp_idx1` - имя индекса, `emp` - имя таблицы, над которой строится индекс, а `ename njob` - имена индексируемых столбцов.

Сервер Oracle автоматически модифицирует хранящиеся в индексе значения при изменении значений в соответствующих столбцах. Поскольку в индексе хранится меньше данных, чем в полной строке таблицы, и поскольку индексы организованы в виде специальной структуры, ускоряющей чтение, для доступа к данным в индексе требуется меньше операций ввода/вывода. Выборка строк по значениям индексированных столбцов может оказаться быстрее выборки по значениям, хранящимся только в базовой таблице. Кроме того, большинство индексов хранятся в отсортированном виде (по возрастанию или убыванию значений в зависимости от способа создания индекса). Из-за этого выборка строк по диапазону значений или возврат строк в отсортированном порядке гораздо быстрее выполняется по проиндексированным столбцам.

Помимо собственно данных в каждой записи индекса хранится ROW- ID ассоциированной строки. Знание ROWID дает самый быстрый способ выбрать строку из базы данных, поэтому доступ посредством индексов наиболее оптимален.

Индекс может быть уникальным (то есть в таблице не может быть двух строк с одинаковым значением индексного ключа) или неуникальным. Строки, для которых значение индексного ключа равно NULL, не включаются в индекс.

Индекс в Oracle - это физическая структура, используемая внутри базы данных. *Ключом* называется логическая сущность, обычно значение, хранящееся в индексе. Как правило, в документации Oracle эти термины взаимозаменяемы. Исключением является внешний ключ.

В следующих разделах описаны четыре способа организации индексов, применяемые в Oracle: стандартные B*-деревья, индексы по реверсированному ключу, битовые индексы и индексы по ключам-функциям, появившиеся в Oracle8L B Oracle Database П^добавились невидимые индексы, которые будут описаны ниже. В Oracle есть также возможность кластеризации данных в таблицах, что позволяет повысить производительность.

## Индексы на базе B*-деревьев

По умолчанию индекс в Oracle имеет структуру B*-дерева. Свое название она получила из-за сходства с перевернутым деревом (рис. 4.1).

B*-дерево состоит из одного или нескольких промежуточных уровней и одного листового уровня. Промежуточные узлы содержат информацию о диапазоне значений, находящихся на следующем промежуточ-

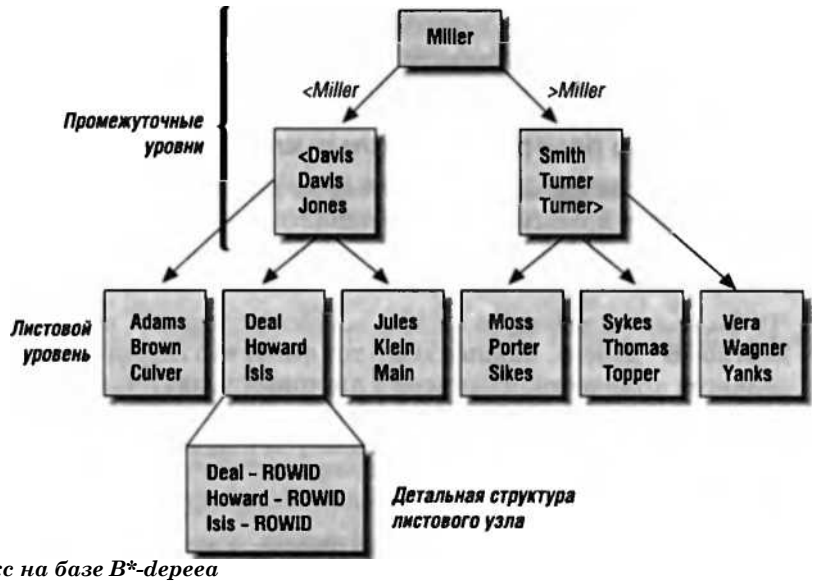



***Рис. 4.1. Индекс на базе B*-дереева***

ном уровне. Количество промежуточных уровней между корнем и листовым уровнем называется *глубиной* индекса. На листовом уровне находятся узлы, содержащие сами индексированные значения и ROWID ассоциированных с ними строк таблицы.

На верхних уровнях B*-дерева узлов мало, поэтому для спуска по дереву требуется относительно небольшое число операций ввода/выво- да. Все листовые узлы расположены в индексе на одной и той же глубине, поэтому для получения любой индексной записи требуется одно и то же количество операций. Это выравнивает производительность доступа по индексу.

Oracle позволяет создавать *индекс-таблицы* (index organized tables, ЮТ), в которых листовые узлы содержат всю строку данных, а не просто ROWID, указывающий на ассоциированную строку. Для индекс- таблицы требуется меньше места на диске, чем для раздельного хранения индекса и таблицы, поскольку в листовых страницах не нужно хранить ROWID. Однако для индекс-таблиц невозможно задать ограничение UNIQUE, и их нельзя хранить в кластере. Кроме того, для индекс-таблиц не поддерживаются распределение, репликация и секционирование, хотя начиная с версии Oracle Database 10g их можно использовать совместно с Oracle Streams для сбора и применения изменений.

В версии Oracle9i механизм индекс-таблиц подвергся ряду усовершенствований, в частности, снято ограничение на использование битовых индексов в качестве вторичных индексов для индекс-таблиц и реализована возможность создавать, перестраивать и объединять вторичные индексы над индекс-таблицами. В Oracle Database 10g эта тенденция получила

дальнейшее развитие - для индекс-таблиц разрешены репликация и секционирование всех видов. Имеются и другие улучшения.

### Индексы по реверсированному ключу

Как следует из названия, в *индексе по реверсированному ключу* порядок байтов в хранимом ключе меняется на обратный. Если в строке хранится значение `"ABCD"`, то в ключе индекса по реверсированному ключу окажется значение `"DCBA"`.

Чтобы понять, зачем это нужно, вспомним еще раз, как устроено стандартное B*-дерево. Важнее всего тот факт, что глубина B*-дерева определяется количеством записей в листовых узлах. Чем больше глубина дерева, тем больше в нем промежуточных уровней и тем больше операций ввода/вывода требуется для доступа к нужному листовому узлу.

На рис. 4.1 мы видим приличный индекс по строковому столбцу. Он сбалансирован, то есть записи равномерно распределены по всем листовым узлам. Однако индексы не всегда устроены так замечательно. Монотонно возрастающие значения, например последовательные порядковые номера или увеличивающиеся даты, всегда добавляются справой стороны индекса. Кроме того, любые удаления из индекса имеют тенденцию смещаться к левому краю по мере удаления старых строк. Со временем B*-дерево становится несбалансированным, листовые узлы с левой стороны заполнены меньше, чем с правой. Из-за несбалансированного роста мы получаем излишне глубокое B*-дерево, в котором новые - большие - значения группируются справа, а левая часть оказывается разреженной. Все сказанное относится и к автоматически уменьшающимся значениям, только в этом случае более плотно заполненной оказывается левая часть.

Эту проблему можно решить, периодически удаляя и заново создавая индекс. Но есть и другое решение - воспользоваться индексом по реверсированному ключу, изменив порядок байтов в значении на обратный. В результате индексные записи более равномерно распределяются по листовым узлам. Например, вместо того чтобы добавлять значения 234, 235 и 236 в правую часть индекса, мы преобразуем их в 432, 532 и 632 в момент записи и восстановим исходные значения в момент чтения. Такие значения распределены по листовым узлам более равномерно.

В итоге применение индекса по реверсированному ключу позволяет исправить дисбаланс, вызванный добавлением монотонно возрастающих значений в стандартное B*-дерево. Дополнительную информацию об индексах по реверсированному ключу и условиях их применения вы найдете в документации Oracle.

### Битовые индексы

В стандартном B*-дереве значения ROWID хранятся в листовых узлах индекса. В *битовом индексе* каждый бит представляет собой один RO- WID. Если строка содержит определенное значение, соответствующий этой строке бит «поднят». Для преобразования номера бита в ROWID применяется функция отображения. В отличие от других типов, в битовых индексах представлены и строки, содержащие NULL в качестве значения ключа.

Для хранения битового индекса с небольшим числом значений требуется гораздо меньше места, чем для стандартного B*-дерева.

Возможности битовых индексов особенно важны при организации хранилищ данных, когда в каждом измерении хранилища встречается много одинаковых значений.

### Индексы по ключам-функциям

*Индексы по ключам-функциям* появились в версии Oracle8i. Такой индекс похож на стандартное B*-дерево или на битовый индекс, только значением ключа является результат вычисления SQL-функции, а не значение, хранящееся в столбце (или в нескольких столбцах).

Этот механизм еще полезнее, если вы создаете собственные функции. Можно написать очень сложную функцию, а потом построить на ее основе индекс и тем самым кардинально повысить производительность запросов, в которых эта функция используется.

### Невидимые индексы

В версии Oracle Database llg для всех рассмотренных выше типов индексов появилась дополнительная возможность - *невидимый индекс*. Обычно любой индекс принимается во внимание оптимизатором. Скрыть индекс от оптимизатора можно, выведя его из оперативного режима или удалив. Но и в том, и в другом случае нужно будет в дальнейшем привести индекс в актуальное создание.

Но что, если нужно лишь временно исключить индекс из рассмотрения оптимизатором, например, для тестирования производительности? Тут-то и приходят на помощь невидимые индексы - такой индекс не считается одним из возможных путей доступа к данным, однако операции обновления и удаления данных в нем тем не менее отражаются.

## Секционирование

С редакцией Enterprise Edition для версий Oracle8 и выше можно приобрести опцию секционирования Partitioning Option. Как следует из названия, это механизм разбиения таблиц и индексов на секции, находящиеся в разных физических областях хранения. Разбиение структуры данных на секции производится на основе значений в столбцах таблицы. Можно организовать секцию, содержащую строки со значением столбца в определенных диапазонах (часто - в диапазонах дат) или сзаданным значением хеш-функции (которая возвращает результат некоего вычисления над значениями из одного или нескольких столбцов). В Oracle9i можно также определить секцию списком значений, что бывает особенно полезно при работе с хранилищами данных. В Oracle Database llg добавлен механизм *интервального секционирования*, позволяющий автоматически порождать новую секцию, если вставляемые данные не попадают ни в один из имеющихся диапазонов.

Можно также организовать двухуровневое секционирование с помощью механизма *составных секций*, используя сочетание разных методов разбиения. До версии Oracle Database llg допускалось только сочетание секционирования по диапазону и по значению хеш-функции. Но теперь секционирование по списку значений ключа можно сочетать с секционированием по списку, по диапазону и по значению хеш- функции, а секционирование по диапазону - с другим способом секционирования по диапазону.

Oracle использует секции для повышения производительности двумя способами:

- не обращается к тем секциям, в которых заведомо нет данных, удовлетворяющих запросу;
- если все данные в некоторой секции удовлетворяют части указанного в запросе условия WHERE, то Oracle просто отбирает все строки из этой секции, не вычисляя условие для каждой строки.

Секционированные таблицы особенно полезны в хранилищах данных, когда данные можно разбить по временным периодам.

Не менее важен тот факт, что секционирование существенно сокращает область действия операций обслуживания и повышает доступность данных. Все операции обслуживания - резервное копирование, восстановление, загрузку - можно выполнять над одной секцией. Такая гибкость позволяет обрабатывать очень большие объемы данных, и при этом обслуживание занимает разумное время. Кроме того, если по какой-то причине вы должны восстановить одну секцию таблицы, то остальные ее секции в это время могут оставаться оперативно доступными.

Если вам доводилось работать с другими СУБД, не располагающими механизмом секционирования, то, возможно, вы пытались реализовать подобную функциональность, разбивая одну таблицу на несколько и далее пользуясь SQL-конструкцией UNION для просмотра данных из всех таблиц разом. Секционированные таблицы дают все преимущества идентичных таблиц, объединяемых с помощью UNION, но без сопряженных с такой реализацией сложностей.

Для извлечения максимума пользы из секционирования иногда имеет смысл одинаково секционировать таблицу и индекс над ней, так чтобы в секциях таблицы и индекса оказались одни и те же строки. Это называется *эквисекционированием* и может быть реализовано автоматически, если при построении индекса над секционированной таблицей указать слово LOCAL. Локальные индексы проще в обслуживании, поскольку такие стандартные операции, как удаление секции, прозрачно применяются к секциям таблицы и индекса.

Oracle продолжает развивать функциональность механизма секционирования. В версии Oracle Database 10g* Release 2 можно оперативно реорганизовывать отдельные секции, максимальное число секций увеличено с 64 Кбайт - 1 до 128 Кбайт - 1 и улучшена оптимизация запросов с отсечением ненужных секций.

В Oracle Database 11# алгоритм отсечения секций снова был усовершенствован. Кроме того, теперь можно управлять секционированием из приложения и добавлен консультант Partition Advisor, который помогает понять, сможет ли секционирование повысить производительность базы данных.

Подробную информацию о структуре секционированных таблиц и связанных с ними ограничениях вы найдете в документации Oracle.

# Дополнительные структуры данных

В базе данных Oracle есть и другие структуры данных, которые могут оказаться полезными при некоторых обстоятельствах.

## Последовательности

Одна из серьезных проблем в многопользовательской базе данных - порождение уникальных числовых значений для ключей или идентификаторов. Для решения этой задачи Oracle предлагает объект, называемый *последовательностью*. Когда кто-нибудь запрашивает значение из последовательности, Oracle возвращает его и увеличивает внутренний счетчик,

избегая конкуренции и временных затрат на взаимодействие с запрашивающим приложением. Oracle может кэшировать диапазон последовательных значений, так что для получения следующего числа не потребуется обращаться к диску - запрос будет удовлетворен из диапазона, хранящегося в SGA.

Для задания последовательности нужно указать имя, значение инкремента и некоторую дополнительную информацию. Последовательности существуют независимо от таблиц, так что одну и ту же последовательность можно использовать для нескольких таблиц.

Посмотрим, что могло бы случиться, если бы в Oracle не было последовательностей. Например, можно сохранить последний порядковый номер в столбце некоторой таблицы. Пользователь, желающий получить очередной номер, должен прочитать значение из этого столбца, увеличить его на фиксированное приращение и записать обратно. Но если получить номер одновременно попытаются сразу несколько пользователей, то каждый может прочитать «последнее» значение до того, как кто-то успеет его обновить. А можно поставить блокировки на строку таблицы, столбец которой содержит порядковый номер, но это приведет к задержкам, поскольку остальным пользователям придется ждать снятия блокировки. Так что же делать? Создайте последовательность.

В версии Oracle Database *llg* разрешено обращаться к последовательностям в выражениях PL/SQL.

## Синонимы

Все структуры данных в базе Oracle принадлежат какой-то *схеме*. Схема ассоциируется с именем конкретного пользователя; обращаясь к объекту, следует указывать перед его именем имя схемы.

Например, пОлное имя таблицы EMP в схеме DEMO - DEMO.EMP. Если имя схемы не указано, Oracle ищет структуру в схеме для текущего имени пользователя.

Схемы удобны потому, что имена объектов должны быть уникальны только в пределах схемы, которой принадлежат. Однако задавать полностью квалифицированные имена объектов утомительно, особенно для конечных пользователей. Чтобы упростить имена, сделав их более удобными для восприятия, можно создать *синоним* для любой таблицы, представления, моментальной копии, последовательности, PL/ SQL-процедуры, функции или пакета.

Синоним может быть *публичным* (тогда к нему может обратиться любой пользователь базы данных) или *приватным* (доступным только владельцу содержащей его схемы).

Например, если пользователь DEMO создаст публичный синоним EMP для таблицы EMP в своей схеме, то все остальные пользователи смогут обращаться к таблице DEMO.EMP просто по имени EMP. Но предположим, что DEMO не создал публичный синоним, а пользователь SCOTT захотел обратиться к таблице EMP в схеме DEMO по сокращенному имени EMP. Тогда SCOTT может создать приватный синоним в собственной схеме. Разумеется, это будет работать, только если SCOTT имеет доступ к таблице EMP в схеме DEMO.

Применение синонимов упрощает доступ к структурам данных. Но синонимы позволяют и скрывать местоположение конкретной структуры данных. Это повышает степень переносимости данных и безопасность таблицы за счет скрытия имени владельца схемы.

До версии Oracle Database 10g* при изменении местоположения объекта, на который ссылается синоним, необходимо было перекомпилировать все PL/SQL-процедуры, где этот синоним встречался.

## Кластеры

*Кластер* - это структура данных, повышающая производительность выборки. Как и индекс, кластер не влияет на логическое представление таблицы.

С помощью кластеров можно хранить взаимосвязанные данные в смежных областях диска. Oracle считывает данные поблочно, поэтому хранение значений рядом сокращает количество операций ввода/вывода, необходимых для их выборки, так как в одном блоке уже могут содержаться взаимосвязанные строки.

Кластер состоит из одной или нескольких таблиц. В состав кластера входит кластерный индекс, в котором хранятся все значения соответствующего кластерного ключа. Каждое значение в кластерном индексе указывает на блок данных, содержащий только строки с тем же значением кластерного ключа.

Если кластер содержит несколько таблиц, то таблицы должны быть соединены, а кластерный индекс должен содержать значения столбцов, по которым производится соединение. Поскольку значение кластерного ключа управляет размещением связанных с этим ключом строк, то при изменении значения ключа СУБД, возможно, будет вынуждена переместить связанные с ним строки.

Кластер может оказаться непригодным для таблиц, которые регулярно приходится сканировать целиком, то есть последовательно просматривать все строки без исключения. Поскольку доступ к кластерной таблице производится через кластерный индекс, который указывает на блок данных, для полного сканирования может потребоваться даже больше операций ввода/вывода, что снижает производительность.

### Хешированные кластеры

*Хешированным* называется кластер, обладающий одной существенной особенностью, которая делает его еще быстрее. Каждый запрос данных из кластеризованной таблицы требует по меньшей мере двух операций ввода/вывода- для кластерного индекса и для данных. В хешированном кластере связанные строки данных хранятся вместе, но группируются согласно *хеш-значению* кластерного ключа. Это значение вычисляется хеш-функцией, то есть каждая операция выборки начинается с вычисления хеш-значения, а затем сразу же выбирается блок данных, содержащий нужные строки.

За счет исключения обращений к кластерному индексу выборка данных из хеш-кластеризованной таблицы может оказаться даже быстрее, чем из просто кластеризованной. Количество возможных хеш- значений управляется параметром HASHKEYS, задаваемым при создании кластера.

Поскольку хешированный кластер указывает прямо на строку в таблице, при создании кластера необходимо выделить место для всех возможных в нем хеш-значений.

Хешированные кластеры дают максимальный выигрыш, когда распределение строк с разными значениями хеш-ключей равномерно. Иногда уже имеется уникальное значение хеш-ключа, например столбец, содержащий уникальный идентификатор. Тогда в качестве хеш-ключа можно задать результат применения хеш-функции к значениям в этом столбце. При этом отпадает необходимость вычислять хеш- функцию в процессе выборки. Кроме того, в определении хешированного кластера можно задать собственную хеш-функцию.

В версии Oracle Database *10g* появились отсортированные хешированные кластеры, в которых данные хранятся не только в кластере, основанном на хеш-значении, но и в том порядке, в котором вставлялись. Такая структура данных повышает производительность приложений, которые обращаются к данным в порядке их добавления в базу.

## Дополнения к логике работы с данными

В СУБД Oracle было добавлено несколько средств, которые, не являясь структурами данных, предоставляют новые способы использования хранящихся в базе данных. Это менеджер правил Rules Manager и фильтр выражений Expression Filter.

### Rules Manager

СУБД Oracle постоянно наращивала предоставляемую функциональность: от простого хранения данных с проверкой некоторых логических атрибутов до хранимых процедур. Компонент Rules Manager, появившийся в версии Oracle Database 10g Release 2, - это еще один шаг в том же направлении.

Идея, стоящая за менеджером правил Rules Manager, проста. *Правило* хранится в базе данных, а приложения вызывают и вычисляют его. Если бизнес-требования изменяются, то описывающие соответствующий сценарий правила можно модифицировать, не затрагивая код приложения. Правила могут быть общими для различных приложений, что позволяет ввести определенные стандарты и одновременно сократить затраты на обслуживание. Можно создавать детальные правила, применяемые в различных сочетаниях для реализации разнообразных условий.

Правила вызываются в ответ на события. *Событие* приводит к вычислению правила и выполнению заданного в правиле действия немедленно или спустя некоторое время.

Менеджер правил следует структуре событие-действие, помогая пользователям определить пять необходимых элементов:

- определить структуру события, которое представляет собой объект, хранящийся в базе данных Oracle; у разных событий отличаются значения атрибутов объекта;
- создать правила, включающие условия и последующие действия;
- создать классы правил для хранения и группировки правил со сходными структурами;
- создать PL/SQL-процедуры, реализующие правила;
- определить представление результатов с целью сконфигурировать правила для внешнего использования, когда невозможно вызвать написанные на PL/SQL действия, например, в

случае, когда приложение работает на нескольких уровнях и включает действия, вызываемые на уровне сервера приложений.

Можно определить процедуры разрешения конфликтов для обработки ситуаций, когда событию соответствует сразу несколько правил. Rules Manager также может объединять различные события в составные события и хранить информацию о состоянии до тех пор, пока не будут получены все события.

Правила могут дать весьма мощный инструмент для реализации сложной логики, но влияют на проектирование приложения. Подробная информация о менеджере правил приведена в документации Oracle.

## Expression Filter

Компонент Expression Filter (фильтр выражений), включенный в версию Oracle Database 10g, применяет Rules Manager для работы с выражениями. *Выражение* - еще один тип объектов, в котором хранятся атрибуты, вычисляемые фильтром выражений. В таблицу, где хранятся атрибуты выражений, вы добавляете столбец типа VARCHAR2, записываете в этот столбец выражения посредством встроенного PL/ SQL-пакета и с помощью стандартных SQL-конструкций задаете значения для выражения. Для сравнения значений с выражением служит оператор EVALUATE в условии WHERE SQL-команды.

Выражения можно использовать для определения сложных характеристик строк, так как у выражения может быть много атрибутов. С помощью выражений можно реализовать отношение многие-ко-многим без промежуточной таблицы; в этом случае для соединения применяются выражения из двух таблиц.

В редакции Enterprise Edition с выражением можно связать индекс. Это позволяет получить выигрыш в производительности, который дают индексы, в применении к величинам, определенным как выражения.

# Оптимизация запросов

Все рассмотренные выше структуры данных - это сущности, определенные на стороне сервера. Пользователи запрашивают у сервера Oracle данные, предъявляя запросы к базе данных. Наилучший способ доступа к запрошенным данным определяет компонент Oracle, называемый оптимизатором запросов.

Одно из величайших достоинств реляционной базы данных - возможность обращаться к данным без указания путей доступа к ним. Получив от пользователя запрос, СУБД Oracle должна решить, как осуществить доступ к данным. Процесс выработки решения называется *оптимизацией запроса*, потому что Oracle ищет оптимальный способ выборки данных. Способ выборки называется *путем выполнения*. Выбрать наиболее эффективный способ доступа к данным нелегко, потому что вариантов может быть много.

Например, даже в случае, когда запрос относится только к одной таблице, у сервера Oracle есть следующие возможности:

- найти ROWID запрошенных строк с помощью индекса, а затем извлечь эти строки из таблицы;
- просмотреть всю таблицу и найти подходящие строки - это называется *полным сканированием таблицы*.

Хотя обычно выборка по индексу гораздо быстрее, процесс получения значений из индекса требует дополнительных операций ввода/выво- да. Оптимизация запроса может иногда свестись к анализу того, имеются ли в запросе условия относительно значений столбцов, хранящихся в индексе. Использование значений из индекса для выборки нужных строк требует меньше операций ввода/вывода и потому оказывается эффективнее, чем извлечение всех данных из таблицы с последующим применением к ним условий отбора.

Еще один фактор, учитываемый при определении оптимального плана выполнения запроса, - наличие в запросе предложения ORDER BY, которое можно было бы реализовать автоматически за счет использования уже отсортированного индекса. Напротив, если таблица мала, оптимизатор может решить, что выгоднее просто считать из нее все блоки, проигнорировав индекс, так как оценочная стоимость ввода/ вывода для индекса и таблицы оказывается выше, чем для одной лишь таблицы.

Оптимизатор должен принимать важные решения, даже когда в запросе участвует только одна таблица. Если же запрос более сложен, например включает много соединяемых между собой таблиц или содержит запутанный критерий отбора и несколько уровней сортировки, то трудность стоящей перед оптимизатором задачи возрастает многократно.

До выхода версии Oracle Database 10g* вам на выбор предлагалось два разных оптимизатора запросов - *по синтаксису* (rule-based optimizer) и *по стоимости* (cost-based optimizer); они описаны в следующих разделах. В Oracle Database 10g* поддержка оптимизатора

по синтаксису была упразднена. Встречающиеся ниже упоминания об оптимизаторе по синтаксису оставлены только для справки и имеют смысл, лишь если вы работаете со старыми версиями Oracle.

## Оптимизация по синтаксису

Оптимизатор запросов был в Oracle всегда, но до версии Oracle7 существовал только оптимизатор по синтаксису. В нем для принятия решения применяется набор предопределенных правил. Поскольку начиная с версии Oracle Database 10g* оптимизатор по синтаксису не поддерживается, эта тема может интересовать вас только в связи с необходимостью поддерживать старые базы данных, где еще имеется выбор.

В некоторых ситуациях оптимизация по синтаксису обеспечивает лучшую производительность, чем ранние версии оптимизатора Oracle по стоимости. Но у оптимизатора по синтаксису есть слабые стороны, одна из которых - чересчур упрощенный набор правил. В Oracle есть около 20 правил, каждому из которых приписан определенный вес. В сложных базах данных запросы часто строятся по нескольким таблицам, имеющим по нескольку 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 по синтаксису назначает каждому потенциальному пути выполнения некоторую оценку, а затем выбирает путь с наилучшей оценкой. Другая слабость оптимизатора по синтаксису проявляется в ситуации, когда имеются пути с одинаковыми оценками. В этом случае оптимизатор разрешает конфликт с помощью синтаксиса SQL-запроса. Выбор окончательного пути определяется тем, в каком порядке таблицы упоминаются в SQL-команде.

К каким последствиям приводит такой алгоритм разрешения конфликтов, можно понять, рассмотрев простой случай, когда маленькая таблица SMALLTAB из 10 строк соединяется с большой таблицей LARGETAB из 10 000 строк (рис. 4.4). Если бы оптимизатор решил сначала читать SMALLTAB, то серверу пришлось бы считать 10 строк, а потом найти в LARGETAB соответствующие им строки. Но если оптимизатор сочтет, что сначала следует читать LARGETAB, то сервер прочтет 10 000 строк, а потом будет 10 000 раз просматривать SMALLTAB в поисках соответствий. Конечно, строки из SMALLTAB, скорее всего, окажутся в кэше, что сократит затраты на каждый просмотр, но разница в производительности тем не менее будет огромной.

Такие казусы зависят от порядка упоминания таблиц в запросе. В описанной ситуации будут возвращены одни и те же результаты при обоих путях выполнения, но ресурсы, затраченные на их получение, различаются весьма существенно.

## Оптимизация по стоимости

Для улучшения качества оптимизации SQL-запросов в состав СУБД Oracle начиная с версии 7 входит оптимизатор по стоимости. Как следует из названия, его задача - не только простой выбор правил из за-

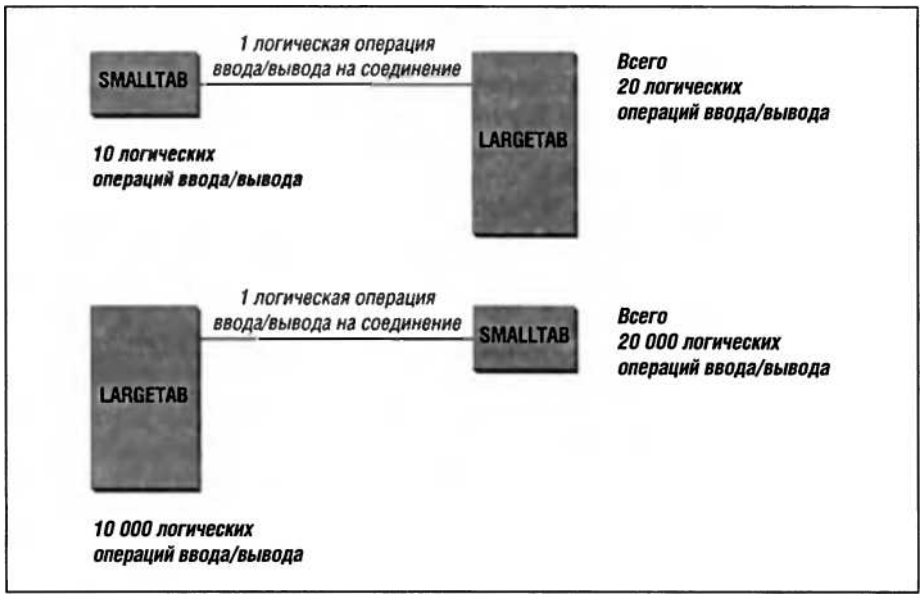



*Рис. 4.4. Выбор плана выполнения запроса оптимизатором*

данного набора. Он выбирает путь выполнения, требующий наименьшего количества логических операций ввода/вывода. Такой подход позволяет избежать потенциальных проблем, рассмотренных выше. В конечном итоге оптимизатор разберется, какая таблица больше, и начнет чтение с нее независимо от синтаксиса SQL-запроса.

В Oracle8 и следующих версиях для выбора наилучшего плана выполнения по умолчанию применяется оптимизатор по стоимости. А начиная с версии Oracle Database 10g* только он и поддерживается. Оптимизатор оценивает стоимость плана выполнения с помощью статистики, относящейся к нужным ему структурам данных. В версии Oracle Database 10# статистика собирается по умолчанию и сохраняется в автоматическом репозитории нагрузки (Automatic Workload Repository, AWR). Статистические данные содержат информацию о доступе к сегментам базы, статистику временной модели, статистику системы и сеансов, сведения об SQL-командах, приведших к наибольшей нагрузке, и статистику истории активных сеансов (Active Session History, ASH).

## Как используется статистика

Оптимизатор по стоимости находит оптимальный план выполнения, присваивая оценку каждому потенциальному плану. При этом он исходит из собственных внутренних правил и логики, а также из статистики, отражающей состояние структур данных в базе. Учитывается статистика для таблиц, столбцов и индексов, участвующих в плане выполнения запроса. В табл. 4.1 перечислены статистики для каждого вида структур данных.

*Таблица 4.1. Статистики базы данных*

| Структура данных | Тип статистики |
|---|---|
| Таблица | Количество строк |
| | Количество столбцов |
| | Количество неиспользованных блоков |
| | Среднее доступное свободное пространство в блоке |
| | Количество расщепленных строк |
| | Средняя длина строки |
| Столбец | Количество различающихся значений в столбце Второе снизу по величине значение в столбце Второе сверху по величине значение в столбце Коэффициент плотности столбца |
| Индекс | Глубина B*-дерева индекса |
| | Количество листовых блоков |
| | Количество различающихся значений |
| | Среднее количество листовых блоков на один ключ |
| | Среднее количество блоков данных на один ключ |
| | Фактор кластеризации |

В Oracle Database 10g* и более поздних версиях собирается также статистика о работе системы в целом, в том числе по производительности и использованию ЦП и подсистемы ввода/вывода. Эта статистика хранится в словаре данных.

Эти статистические данные можно использовать по отдельности или в сочетании для определения полной стоимости плана выполнения в терминах количества операций ввода/вывода. Статистики отражают как размер таблицы, так и объем неиспользованного пространства в блоках; последний показатель позволяет оценить, сколько операций ввода/вывода потребуется для выборки строк. Статистика индексов отражает не только глубину и ширину дерева индекса, но и степень уникальности хранящихся в нем значений - от этого зависит, насколько просто будет отобрать строки с помощью данного индекса.

Точность работы оптимизатора по стоимости зависит от точности используемых им статистик, поэтому своевременное обновление статистики было необходимо всегда. Раньше для оценки статистических данных приходилось пользоваться SQL-коман- дой ANALYZE. В более старых версиях многие администраторы баз данных применяли также встроенный PL/SQL-пакет DBMS_STATS, в котором имеется ряд процедур, помогающих автоматизировать сбор статистики.

Неактуальная статистика может вызвать снижение производительности, поэтому процедура сбора статистики была полностью автоматизирована. Активизировать этот механизм можно избирательно. Например, в Oracle Database 10g можно включить автоматический сбор статистики для таблицы, который

начинается, если статистика устарела (то есть с момента последнего сбора изменилось более 10 процентов строк в таблице) или отсутствует вовсе.

Статистика помогает оптимизатору принимать гораздо более осмысленные решения по выбору оптимального плана выполнения. Например, иногда оптимизатору приходится выбирать между двумя индексами, в которых можно было бы искать значение. Оптимизатор по синтаксису мог бы поставить обоим индексам одинаковые оценки и выбрать план выполнения исходя из порядка их появления в условии WHERE. Но оптимизатор по стоимости знает, что в одном индексе 1000 записей, а в другом 10 000. Он знает даже, что в первом индексе только 20 уникальных значений, а во втором - 5000. Стало быть, селективность большего индекса гораздо выше, поэтому именно он получит более высокую оценку и будет использован для выполнения запроса.

# Лекция 5. Администрирование Oracle

## Средства администрирования

В версии Oracle Database 10g с ее «интеллектуальной инфраструктурой» администрирование базы данных сильно упростилось. Ушли в прошлое многие ручные процедуры, необходимые в предыдущих версиях. В Oracle Database llg появились дополнительные средства автоматической настройки и управления. Были автоматизированы основные задачи обслуживания, в том числе сбор статистики для оптимизатора. Включены консультанты Segment Advisor и SQL Tuning Advisor. Администрирование инфраструктуры в целом достигается за счет механизмов автоматического управления в сочетании с Oracle Enterprise Manager.

Статистические данные, включая историю работы в активных сеансах, теперь накапливаются в автоматическом репозитории нагрузки (Automatic Workload Repository, AWR). С помощью этих данных автоматический диагностический монитор базы данных (Automatic Database Diagnostic Monitor, ADDM) отслеживает изменение производительности. Генерируемые сервером «своевременные» оповещения появляются в Enterprise Manager. Для решения возникающих проблем часто достаточно прочитать оповещение и применить содержащиеся в нем рекомендации. Все это в корне отличается от ситуации, имевшей место до версии Oracle Database 10g, когда приходилось активно следить за событиями, просматривать V$-таблицы, выявлять проблематичные SQL-запросы и решать, как устранить проблему.

## Консультанты по базе данных

ADDM - лишь один из консультантов, имеющихся ныне в Oracle и доступных из Enterprise Manager. К вашим услугам ряд других консультантов.

*SQLAdvisor*

В версию Oracle Database llg входят SQL Tuning Advisor, SQL Access Advisor и Partition Advisor. SQL Tuning Advisor анализирует SQL-команды и дает рекомендации по их улучшению. SQL Access Advisor и Partition Advisor советуют создавать индексы, материализованные представления и секционированные таблицы, когда это имеет смысл.

*SQL Performance ImpactAdvisor*

Этот консультант, появившийся в версии Oracle Database llg, позволяет прогнозировать, как некое изменение в системе отразится на производительности SQL.

*Консультанты по управлению памятью*

Memory Advisor - это экспертная система, которая автоматически управляет памятью и устраняет необходимость вручную подстраивать SGA и PGA. В Oracle Database llg ее рекомендуется включать. Если вместо нее включена только опция автоматической разделяемой памяти, то будут доступны консультанты Shared Pool (SGA) Advisor и PGA Advisor. Наконец, если разделяемой памятью вы управляете вручную, то к вашим услугам консультанты Shared Pool (SGA) Advisor, Buffer Cache Advisor и PGA Advisor.

*Segment Advisor*

Консультант Segment Advisor устраняет необходимость выявлять фрагментированные объекты и реорганизовывать их с помощью сценариев. Он советует, какие объекты следует уплотнить, а вам достаточно просто принять его рекомендации. Предоставленной информацией можно воспользоваться и для планирования будущей емкости.

*UndoAdvisor*

Помогает выбрать размер табличного пространства отката, позволяет задать нижний порог сохранения информации в пространстве отката для механизма ретроспекции. В Oracle

Database 11 *g* реализовано автоматическое управление пространством отката.

*MTTRAdvisor*

Консультант Mean Time to Recovery (MTTR) Advisor дает рекомендации по настройке параметров MTTR (среднее время восстановления). Среднее время восстановления после системного сбоя исходя из потребностей бизнеса задает администратор базы данных в Enterprise Manager, после чего компоненты Oracle автоматически реконфигурируются.

*Streams TuningAdvisor*

Генерирует отчеты о пропускной способности и задержках для данной топологии подсистемы Streams, соединяющей базы данных, и способен самостоятельно находить узкие места.

В Oracle Database llg имеется еще один класс консультантов, помогающих разрешать проблемы, возникающие в базе данных. При обнаружении критических ошибок встроенная инфраструктура диагно- стируемости способна выполнить углубленный анализ, называемый *проверкой работоспособности* (health check) с применением компонента Health Monitor. Консультанты обращаются к различным диагностическим данным, в том числе к трассировкам базы данных, журналу оповещений, отчетам Health Monitor и информации, хранящейся в репозитории Automatic Diagnostic Repository (ADR). В состав инфраструктуры входит также инструмент SQL Test Case Builder, позволяющий воспроизвести ошибку и передать информацию в службу Oracle Support. К этой инфраструктуре относятся следующие консультанты:

*SQL RepairAdvisor*

Если при выполнении SQL-команды возникает критическая ошибка, SQL Repair Advisor анализирует команду и предлагает заплату, которая могла бы исправить ошибку.

*Data RecoveryAdvisor*

Применяется для восстановления запорченных блоков, запорченных или отсутствующих файлов и исправления других ошибок вданных. Интегрирован с механизмом проверки работоспособности и с RMAN.

## Автоматическое управление хранением

В версию Oracle Database 10g включена подсистема автоматического управления хранением (Automatic Storage Management, ASM). Она играет роль файловой системы и менеджера томов в базе данных, обеспечивая автоматизированное расслоение файлов и зеркалирование экстентов. Администратору достаточно определить пул хранения или группу дисков и управлять ею из EM. Группы дисков по умолчанию создаются с обычной избыточностью (двустороннее зеркалирование). Но можно задать высокую избыточность (трехстороннее зеркалирование) или внешнюю избыточность (без зеркалирования). Группами отказа (failure group) называется совокупность управляемых ASM дисков с общей точкой отказа, поэтому для обеспечения высокой доступности зеркалирование автоматически производится на диски из другой группы отказа.

Файлами, находящимися на группах дисков ASM, управляет Oracle. ASM может управлять файлами данных, файлами журналов, управляющими файлами, архивными журналами и наборами резервных копий RMAN. В случае изменения конфигурации системы хранения, например после добавления новых устройств в пул или их удаления из пула, данные могут быть перераспределены в фоновом режиме с целью динамического балансирования рабочей нагрузки.

# Oracle Enterprise Manager

Впервые программа Oracle Enterprise Manager (EM) была включена в версию Oracle7 и предназначалась для упрощения администрирования базы данных. В первых версиях EM в качестве клиентских машин могли выступать только рабочие станции под управлением Windows. В Oracle8i появился Java-апплет, реализующий консоль внутри броузера.[1] В состав продуктов, поставляемых в версии Oracle9i, в том числе Oracle Application Server, вошла HTML-консоль. Теперь именно она является основой Enterprise Manager. В версии Oracle Databasellg консоль на базе Java-апплета уже не поставляется.

На сегодня EM - нечто гораздо большее, чем просто интерфейс администрирования базы данных. К EM прилагается много дополнительных пакетов, позволяющих управлять не только базами данных Oracle, но и другими установленными компонентами инфраструктуры:

*Пакеты для управления базой данных (Database Management Packs)*

Диагностика, настройка, управление изменениями, управление конфигурацией, провизионирование (Provisioning).

*Автономные пакеты для управления (Standalone Management Packs)* Провизионирование,

управление на уровне служб.

*Пакеты для управления приложениями (Application Management Packs )*

E-Business Suite, PeopleSoft Enterprise, Siebel.

*Пакеты для управления ПО промежуточного уровня (Oracle Application Server)*

Диагностика, управление конфигурацией, управление идентификацией, провизионирование (Provisioning), управление SOA.

*Административные коннекторы (Management Connectors)*

Microsoft Operations Manager, Remedy Helpdesk *Пакеты для*

*управления операционной системой* Oracle Linux

*Подключаемые модули для мониторинга системы*

EMC Celerra, EMC Symmetrix DMX, NetApp Filer, BEA WebLogic, JBoss Application Server, IBM WebSphere, IBM WebSphere MQ, IBM DB2, Microsoft IIS Server, Microsoft Active Directory, Microsoft Biz- Talk Server, Microsoft Commerce Server, Microsoft ISA Server, Microsoft .NET framework, Microsoft SQL Server, Check Point Firewall, Juniper Netscreeen Firewall, F5 BigIP Local Traffic Manager, Linux Hosts, Unix Hosts, Windows Hosts.

Рассмотрим роль EM в администрировании СУБД. К пакетам для управления базой данных относятся следующие компоненты:

*Пакет для диагностики базы данных (Database Diagnostics Pack)*

Обеспечивает автоматическую диагностику производительности, применяя ADDM, AWR, шаблоны мониторинга и развитые системы оповещения о событиях и предупреждений.

*Пакет для настройки базы данных (Database TuningPack)*

Обеспечивает сбор статистики, профилирование SQL, анализ путей доступа и структуры SQL-команд. Пользуется консультантом SQL Tuning Advisor и включает консультант SQL Access Advisor и мастер Object Reorganization Wizard.

*Пакет для управления изменениями базы данных (Database Change Management Pack)*

Формирует опорные характеристики работы версии, копирование объектов базы данных и самих данных, а также обновление определений объектов.

*Пакет для управления конфигурацией базы данных (Database Configuration Management Pack)*

Обеспечивает сбор данных и составление отчетов о системных компонентах, сравнение и хранение истории конфигураций. Включает менеджер политик и консультант по критическим заплатам.

*Пакет для провизионирования (Database ProvisioningPack)*

Обеспечивает автоматическое наложение заплат, клонирование, провизионирование и конвертирование единственного экземпляра в кластер RAC.

## Архитектура Enterprise Manager

Enterprise Manager можно использовать для локального и дистанционного администрирования, в том числе через брандмауэр. С помощью одной консоли можно администрировать одну или сразу несколько баз данных. Если EM применяется для администрирования СУБД Oracle, развернутой в кластере компьютеров, то иногда его называют *Grid Control*.

На начальной странице Grid Control отображается список управляемого ПО и высокоуровневый обзор состояния компонентов решетки. С этой страницы можно перейти на консоль отдельной базы данных, сервера приложений и других элементов.

*Административные агенты Oracle (Oracle ManagementAgents)*

Эти агенты следят за работоспособностью, состоянием и производительностью своих подопечных. Агенты автоматически обнаруживают все компоненты Oracle и передают EM разнообразную информацию о конфигурации программных и аппаратных средств по протоколу HTTP/HTTPS. Для каждого отслеживаемого экземпляра Oracle имеется свой агент.

*Консоль Enterprise Manager Console*

Позволяет просматривать состояние всех отслеживаемых компонентов и предоставляет инструменты управления и диагностирования.

*Служба Oracle Management Service (OMS)*

Получает информацию от агентов и сохраняет ее в репозитории Oracle Management Repository. Локальные версии репозиториев OMS хранятся в каждой локальной базе данных. OMS - это веб-прило- жение на базе технологии J2EE, которое заодно выводит пользовательский интерфейс консоли Grid Control в центральном пункте.

*Репозиторий Oracle Management Repository*

Enterprise Manager обращается к этому репозиторию, когда ему нужны данные о работоспособности, состоянии и производительности.

Репозиторий автоматически устанавливается в каждую локальную базу данных для

обслуживания локальных консолей управления Enterprise Manager Database Control. Компонент Oracle Grid Control обращается к репозиторию, обслуживаемому центральной OMS и центральным агентом Oracle Management Agent.

Административные агенты имеются для различных операционных систем, для которых есть версия СУБД Oracle, и отвечают за автоматическое обнаружение служб, мониторинг событий и выполнение предопределенных заданий. Агенты могут также посылать прерывания протокола Simple Network Management Protocol (SNMP) мониторам производительности базы данных или иным инструментам мониторинга.

## Консоли Oracle Enterprise Manager

Популярность EM растет по мере того, как базы данных Oracle развертываются на нескольких операционных системах внутри одной компании и добавляются все новые программные компоненты. EM предоставляет единый интерфейс управления в таком разнородном окружении, тогда как административные сценарии не всегда проектируются с учетом подобных условий. Кроме того, интерфейс и каркас Enterprise Manager обеспечивают простой доступ к новым возможностям автоматического мониторинга, реагирования на предупреждения и управления заданиями, отчетами, ролями и привилегиями. Консоль EM и скрывающаяся за ней «интеллектуальная инфраструктура» устанавливаются в ходе обычной процедуры инсталляции СУБД Oracle. Будучи установлен, EM автоматически находит базы, которыми будет управлять.

Простые интерфейсы к EM можно развернуть и с помощью продукта Oracle Application Server Portal. В комплект поставки портала уже входят административные портлеты для вывода итоговых сведений об управляемых системах, предупреждений об оповещениях, для которых достигнут или превышен порог, детальных метрик, временных шкал доступности и сводной информации для руководства.

Основные метрики (состояние процессора, активных сеансов и время реакции на SQL-запросы) обычно отображаются в графическом виде. Также выводятся сводные диагностические данные, сведения об использовании дискового пространства, состояние подсистемы обеспечения высокой доступности, предупреждения и информация о нарушении политик. На этой вкладке имеются ссылки на консультативный центр Advisor Central (страница для быстрого доступа к консультантам) и на страницы других метрик, например, на содержимое сигнального файла (alert log).

Performance
: Содержит важнейшие сводные статистические данные о производительности, например об использовании процессора, среднем количестве активных сеансов, дисковом вводе/выводе и пропускной способности экземпляра.

Availability
: Отсюда можно управлять резервным копированием и восстановлением с помощью таких инструментов, как RMAN и LogMiner.

Server
: Содержит ссылки на такие средства автоматизированного обслуживания, как Automatic Memory Management, AWR и планировщик заданий.

Schema
: Здесь можно управлять пользователями и их привилегиями, таблицами, индексами, представлениями, синонимами и связями баз данных, а также инициировать смежные функции, например ретроспекцию (Flashback).

Data Movement
: Администрирование средств перемещения данных, например Streams, и переносимых табличных пространств.

Software and Support
: Предоставляет доступ к рабочему месту Support Workbench для отправки в службу поддержки Oracle отчетов об аномальных ситуациях, которые можно наблюдать при помощи AWR.

## EM2Go

EM2Go - это мобильная версия Enterprise Manager, появившаяся в Oracle Database 10g. Она предназначена для удаленного администрирования экземпляров баз данных и серверов приложений Oracle по беспроводным каналам связи. В EM2Go реализовано подмножество функциональности Enterprise Manager, но он также пользуется описанной выше службой OMS, ассоциированным с ней репозиторием Management Repository и агентами Oracle. Доступ к консоли Enterprise Manager осуществляется из броузера Microsoft Pocket PC Internet Explorer,

работающего на КПК. Обмен данными между консолью и OMS и агентами производится по протоколу HTTP.

Администратор входит в Enterprise Manager с начальной страницы EM2Go, вводя имя пользователя и пароль. После успешного входа он видит перечень предупреждений и список управляемых систем. Каждый элемент списка - ссылка, позволяющая перейти на страницу с детальной информацией.

Можно настроить EM2Go, так чтобы он посылал предупреждения непосредственно на ваш КПК по электронной почте. Поддерживается выполнение произвольных SQL-запросов и команд операционной системы. В состав данных о производительности входит графическое представление истории изменения метрик, а также предупреждения и оповещения от СУБД Oracle и от сервера приложений Oracle Application Server. Кроме того, имеется доступ к главной странице базы данных.

## Фрагментация и реорганизация

Фрагментация может отрицательно сказаться на производительности, поэтому в прошлом многие администраторы всеми силами боролись с ней. Нежелательный эффект фрагментации - появление небольших кусочков несмежного «свободного пространства», которые невозможно использовать повторно.

В Oracle набор смежных блоков называется *экстентом*, а набор экстентов - *сегментом*. В сегментах можно хранить все, что угодно: таблицу, индекс или сегмент отката. Как правило, сегмент состоит из нескольких экстентов. Когда один сегмент оказывается заполнен, система начинает использовать следующий экстент в сегменте. По мере того как в результате работы базы данных в непрерывной области, занятой экстентом, появляются «дырки» (это и называется фрагментацией), количество экстентов в сегменте растет. Чем сильнее фрагментирован сегмент, тем больше приходится выполнять операций ввода/вывода, что снижает производительность.

## Устранение фрагментации

В версии Oracle Database 10g устранить фрагментацию стало довольно просто. Консультант Segment Advisor, доступный из EM, позволяет уплотнять сегменты в оперативном режиме. Диагностический монитор ADDM сообщит, когда настанет время для уплотнения, а вам останется лишь принять его рекомендации.

В СУБД Oracle9i для устранения фрагментации обычно применялась оперативная реорганизация с помощью команды CREATE TAB- LE...AS SELECT. Иными словами, содержимое исходной таблицы копировалось в новую, без запрещения обновлений исходной таблицы. Все изменения, внесенные во время копирования, запоминались, а потом применялись к новой таблице. В ходе этой операции можно было изменять физические и логические атрибуты таблицы, что и позволяло выполнять оперативную реорганизацию.

До версии Oracle9i устранить фрагментацию было сложнее. Основная рекомендация состояла в том, чтобы избегать фрагментации путем тщательного планирования. Но если уж приходилось устранять фрагментацию, то обычно таблицу экспортировали, удаляли, а затем импортировали. На протяжении всей процедуры такой реорганизации данные были недоступны. Многие администраторы уверяли, что после реорганизации сегментов в единый экстент наблюдалось повышение производительности. Но со временем количество занятых таблицей экстентов возрастало, и производительность снова падала.

Однако повышение производительности Oracle *не* было связано с уменьшением количества экстентов. При удалении и повторном создании таблицы производительность увеличивалась по нескольким причинам:

- В каждый блок загружалось максимальное возможное количество строк.
- Поэтому верхняя отметка заполненности таблицы (максимальный номер блока с данными, который в ней когда-либо имелся) оказывалась минимальной.
- Все индексы над таблицей перестраивались, следовательно, блоки индексов тоже заполнялись по максимуму. Иногда из-за этого уменьшалась глубина индекса, а, значит, и количество операций ввода/вывода, необходимых для доступа к листовым блокам.

Ввод автоматизированной дефрагментации и уплотнения сегментов без прерывания доступа к данным в версии Oracle Database 10g и последующих существенно упростил решение этой проблемы. В результате постоянно поддерживаются условия для достижения оптимальной производительности.

## Резервное копирование и восстановление

Даже если вы приняли все меры предосторожности, критически важные записи иногда могут пропасть из базы данных в результате ошибки человека или программного либо аппаратного

сбоя. Единственный способ подготовиться к такому развитию событий - регулярно выполнять резервное копирование.

К потере данных в базе Oracle могут привести две причины: *сбой экземпляра,* когда экземпляр останавливается без выполнения должной процедуры, и *отказ носителя*, когда повреждаются диски, на которых хранится информация.

В первом случае Oracle автоматически выполняет процедуру восстановления после сбоя. Например, Real Application Clusters самостоятельно выполнит восстановление после сбоя любого экземпляра. Однако восстановление после отказа носителя должен инициировать администратор. Способность к успешному восстановлению после таких отказов - результат тщательного планирования. Вам придется восстановить старые копии поврежденных файлов данных, а затем накатить архивные и оперативные журналы.

Чтобы не оказаться застигнутым врасплох, администратор должен выполнять следующие действия:

- мультиплексировать оперативные журналы, располагая копии на разных дисках, подключенных к разным контроллерам;
- эксплуатировать базу данных в режиме ARCHIVELOG, чтобы журналы архивировались перед повторным использованием;
- хранить архивные журналы в разных местах;
- поддерживать несколько копий управляющих файлов;
- часто выполнять резервное копирование физических файлов данных и, в идеале, хранить несколько копий в разных местах.

Эксплуатация базы в режиме ARCHIVELOG гарантирует возможность восстановления в состоянии, непосредственно предшествующем моменту отказа. При этом администратор может производить оперативное резервное копирование, не прекращая доступ к данным. Кроме того, архивные журналы можно применять к резервной базе данных.

Компонент Recovery Manager (RMAN), впервые появившийся в версии Oracle8 и с тех пор значительно усовершенствованный, предоставляет удобный интерфейс для выполнения этой процедуры. Ныне к RMAN можно обращаться из Enterprise Manager.

## Типы резервного копирования и восстановления

Есть два основных типа резервного копирования.

*Полное резервное копирование*

Могут копироваться отдельные файлы данных, табличные пространства, управляющие файлы (текущие или архивные) или вся база целиком (включая все файлы данных и текущий управляющий файл). В набор резервных копий включаются все блоки данных за исключением тех, что никогда не использовались (к управляющим файлам и журналам это не относится, для них никакие блоки не пропускаются).

*Инкрементное резервное копирование*

Могут копироваться отдельные файлы данных, табличные пространства или вся база целиком. Включаются только блоки данных, изменившиеся с момента снятия предыдущей копии.

Запустить процедуру резервного копирования можно с помощью Recovery Manager или из интерфейса Enterprise Manager к RMAN - в обоих случаях применяется механизм резервного копирования базы данных. А можно воспользоваться стандартными средствами резервного копирования из операционной системы.

RMAN поддерживает большинство возможностей резервного копирования базы, включая копирование открытой базы (оперативное), закрытой базы, инкрементные копии на уровне блоков Oracle, обнаружение запорченных блоков, автоматическое резервное копирование, каталоги резервных копий и копирование на носители с последовательным доступом. В версии Oracle9i к RMAN добавлена возможность задать одноразовую конфигурацию резервного копирования, окна восстановления для задания и управления датой истечения срока хранения резервных копий и возможность перезапуска процессов резервного копирования и восстановления. Кроме того, включена поддержка восстановления в тестовом режиме.

В версии Oracle Database 10g RMAN умеет копировать образы базы данных, табличных пространств или файлов данных и применять инкрементные резервные копии к образам файлов данных. Скорость инкрементного копирования повышена за счет механизма отслеживания изменений, когда считываются и помещаются в копию только изменившиеся блоки.

Для восстановления есть следующие возможности:

- полное восстановление базы данных до момента сбоя;
- восстановление табличного пространства на заданный момент времени (когда некоторое табличное пространство восстанавливается не на тот же момент времени, что остальная часть базы данных);
- восстановление всей базы данных на заданный момент времени (предшествующий последнему актуальному состоянию);
- восстановление до получения команды CANCEL;
- восстановление до указанного системного номера изменения (System Change Number, SCN).

Выполнять восстановление можно с помощью RMAN, пользуясь каталогом восстановления или управляющим файлом, либо с помощью SQL или SQL*Plus.

В версии Oracle Database 10g RMAN повысил надежность резервного копирования и восстановления за счет нескольких новых возможностей. Включено копирование и восстановление резервных управляющих файлов. RMAN теперь может автоматически повторять операцию копирования или восстановления, завершившуюся неудачей. В процессе восстановления RMAN может автоматически создавать и восстанавливать файлы данных, отсутствующие в самой последней копии. Если обнаруживается, что последняя резервная копия отсутствует или запорчена, RMAN автоматически обращается к более старой.

Для ускорения резервного копирования и восстановления в Oracle Database 10g введена область быстрого восстановления (Flash Recovery Area), позволяющая хранить необходимые для восстановления файлы в отдельном месте на диске. Речь идет о копии управляющего файла, архивных журналах, ретроспективных журналах базы данных, копиях файлов данных и резервных копиях RMAN. Параметр RETENTION AREA позволяет сохранять необходимые для восстановления файлы в течение указанного промежутка времени. Когда срок хранения резервных копий и архивных журналов истекает, они автоматически удаляются. Сконфигурировать область Flash Recovery Area позволяет подсистема ASM. Если места на диске недостаточно, то можно предписать RMAN сжимать резервные копии.

## Компонент Oracle Secure Backup

Компонент Secure Backup начал поставляться в составе версии Database 10g под названием Oracle Secure Backup Express (XE), заменив систему управления хранилищем на магнитных лентах Single Server Version (LSSV) производства компании Legato. Начиная с версии Enterprise Manager 10g Release 2 компонент Secure Backup интегрирован в интерфейс Enterprise Manager. Secure Backup XE использует способность RMAN считывать блоки базы данных напрямую и обеспечивает защиту данных на лентах для одного сервера, к которому подключен один накопитель на магнитных лентах. Если требуется решение масштаба предприятия, то Oracle предлагает за дополнительную плату версию Secure Backup, поддерживающую несколько накопителей и произвольное количество серверов.

Oracle Secure Backup поддерживает свыше 200 типов накопителей на магнитных лентах, а также протокол управления сетевыми данными Network Data Management Protocol (NDMP), хранилища, подключаемые по сети (network attached storage, NAS), виртуальную лентотеку Virtual Tape Library (VTL), администрирование на основе политик, классификацию хранилищ, динамическое разделение дисков, аутентификацию по сертификатам и шифрование резервных копий.

Разумеется, есть и множество разнообразных альтернативных решений для реализации резервного копирования в Oracle, предлагаемых сторонними производителями. Корпорация Oracle продолжает поддерживать программу Oracle Backup Solutions Program (BSP), в рамках которой партнеры могут сертифицировать свои продукты для резервного копирования и восстановления с ленточных хранилищ с применением RMAN. Актуальный список таких решений опубликован на сайте Oracle Technology Network.

## Управление жизненным циклом информации

Подсистема Information Lifecycle Management (ILM) предоставляет средства для определения классов данных, создания иерархии хранилищ для различных классов данных, создания политик доступа к данным и перемещения данных, а также реализации политик соответствия данных. Чаще всего ILM применяется для перемещения данных между различными устройствами, наиболее точно отвечающими целям хранения, например, между дисками разных типов. Обусловлено это тем, что многие администраторы предпочитают хранить наиболее часто используемые данные на дорогих скоростных дисках, а данные, к которым обращаются редко, - на дисках подешевле, пусть даже более медленных.

Поддержка ILM появилась в 2006году в версии Oracle9i, когда был представлен инструмент ILM Assistant, который можно загрузить с сайта Oracle Technology Network. Помимо ILM Assistant вам понадобится набор объектов Oracle Application Express (прежнее название HTML DB),

устанавливаемый в ту базу, где находятся управляемые данные.

ILM Assistant предоставляет графический интерфейс для создания определений и политик жизненного цикла, которые хранятся в таблицах базы данных. Он подсказывает, когда пришло время перемещать, архивировать или удалять данные, а также сообщает о потенциальной экономии и требуемом объеме места на диске. ILM Assistant также помогает определить стратегию секционирования, соответствующую вашим целям управления жизненным циклом информации. После того как стратегия определена, ILM Assistant генерирует сценарии перемещения данных.

При первом запуске ILM Assistant следует перейти на вкладку Lifecyde Setup (Настройка жизненного цикла). Здесь определяются логические уровни хранения, создаются определения жизненного цикла и выбираются таблицы, которыми предстоит управлять (рис. 5.4). Затем ILM Assistant может порекомендовать, как разместить данные. Дополнительно можно посмотреть результаты имитации секционирования, сводную информацию о жизненном цикле, стоимость хранения, а также ввести примечания к политикам.

При последующих запусках ILM Assistant вы увидите календарь событий жизненного цикла (Lifecycle Events Calendar), в котором присутствует список запланированных событий. Календарь, сами события, историю распознавания событий можно посмотреть на вкладке Lifecyde Management (Управление жизненным циклом).

На вкладке Reports (Отчеты) представлен ряд отчетов, в том числе отчет о стоимости многоуровневого хранения в разрезе жизненного цикла или таблицы, сводка логических уровней хранения, разбиение по таблицам или по уровням хранения, справка о сроках хранения и справка о защите данных. На вкладке Compliance and Security (Соответствие и безопасность) можно: посмотреть состояние виртуальной частной базы данных (VPD), политики и даты генерации цифровых подписей; создать подписанные результирующие наборы для отслеживания неизменности; посмотреть перечень определений безопасности и конфиденциальности, политик, представлений и привилегий доступа; просмотреть и настроить политики детального аудита (FGA), посмотреть и написать примечания к политикам.

## Автоматизированное наложение заплат

Служба Oracle Support размещает на сайте MetaLink извещения о найденных ошибках или уязвимостях и о выпущенных заплатах (patches). Начиная с версии Oracle Database 10g внедрена автоматизированная процедура уведомления и наложения заплат, позволяющая вам быстрее реагировать на извещения об обнаруженных ошибках. Оповещения о найденных ошибках и уязвимостях могут выводиться прямо на консоль Enterprise Manager. На вкладке Enterprise Configuration Management появится ссылка на заплату и информация о том, к какой из управляемых систем ее следует применить.

В среде кластера RAC или решетки наложить заплату можно на все узлы, не останавливая кластер или решетку целиком. Кроме того, если наблюдается аномальное поведение, для конкретного экземпляра заплату можно откатить.

# Лекция 6. Безопасность, аудит и соответствие требованиям в Oracle

Основное назначение Oracle - управление ценными данными, необходимыми буквально для всех операций в организации. Ценность данных отчасти определяется тем, что они *ваши*, то есть могут обеспечить вашей компании уникальные преимущества. Поэтому необходимо защитить данные от доступа посторонних лиц. Мы остановимся на трех различных аспектах общей задачи защиты данных.

- *Безопасность* обеспечивается инструментами, которые позволяют обращаться к данным только тем, у кого есть на то разрешение.
- *Аудит* позволяет узнать, кто и что делал с вашими данными. Под аудированием понимается процедура ведения истории доступа, которой можно воспользоваться для лучшего понимания операций, выполняемых в базе, а также для выявления попыток и фактов нарушения защиты. На этапе конфигурирования Oracle Database 11 *g* вам будет задан вопрос, хотите ли вы оставить принимаемые по умолчанию параметры безопасности без изменения. Если вы согласитесь, то будет включен аудит и установлены опции профиля, определяющие политику управления паролями. Изменится и ряд других параметров инициализации.
- *Соответствие требованиям* - это способность доказать, что данные действительно хранятся надежно и безопасно. Ныне такое доказательство часто должно предоставляться по закону. Хотя многие технические специалисты считают, что доказательство соответствия - это уже перебор, но факт остается фактом - несоответствие требованиям может стать

основанием для наложения крупного штрафа на компанию. Поэтому для руководства этот аспект представляет особый интерес.

## Безопасность

Один из самых важных аспектов эффективного администрирования базы данных Oracle в многопользовательской среде - это создание безопасной схемы управления доступом и модификацией данных. Oracle позволяет предоставить права доступа отдельным пользователям или ролям.

Управление безопасностью обычно осуществляется на трех уровнях:

- уровень базы данных;
- уровень операционной системы;
- сетевой уровень.

На уровне операционной системы у администратора базы данных (АБД) должны быть права для создания и удаления относящихся к базе данных файлов. Напротив, у обычных пользователей таких прав быть не должно. Информация о безопасности на уровне операционной системы приведена в стандартной документации Oracle. Во многих крупных организациях АБД или администратор по безопасности базы данных работают в тесном контакте с администраторами вычислительной системы, чтобы координировать усилия по разработке требований и практических мероприятий, направленных на обеспечение безопасности.

В требованиях к безопасности базы данных описываются процедуры предоставления доступа к базе путем назначения каждому пользователю пары имя/пароль (username/password). В требованиях может также оговариваться ограничение объема ресурсов (дискового пространства и процессорного времени), выделяемых одному пользователю, и постулироваться необходимость аудита действий пользователей. Механизм обеспечения безопасности на уровне базы данных также обеспечивает управление доступом к конкретным объектам схемы базы данных.

## Имена пользователей, привилегии, группы и роли

АБД (DBA) или администратор по безопасности базы данных создает *имена пользователей*, указываемые при установлении соединения с базой. В процессе установки автоматически создаются две учетные записи, приписанные к роли DBA: SYS и SYSTEM. (Роль DBA описана ниже.)

С каждым именем пользователя ассоциирован пароль для предотвращения несанкционированного доступа. Создаваемый или вводимый взамен старого пароль должен:

- содержать не меньше восьми знаков;
- содержать хотя бы одну цифру и хотя бы одну букву;
- не совпадать с именем пользователя, написанным «задом наперед»;
- не совпадать с именем пользователя и отличаться от него не только добавленным в конце числом от 1 до 100;
- не совпадать ни с одним простым словом из внутреннего списка;
- отличаться от предыдущего пароля (при замене) по крайней мере тремя знаками.

Oracle может проверять эти условия при каждой операции создания или изменения пароля, если такой режим установлен в политиках безопасности.

После успешного подключения к базе данных возможности пользователя ограничены *привилегиями*, то есть правами на выполнение определенных SQL-команд. Некоторые привилегии могут быть предоставлены на уровне системы в целом (например, право удалять строки из любой таблицы базы данных), другие применяются только к конкретному объекту в конкретной схеме (например, возможность удалять строки из конкретной таблицы).

*Ролью* называется именованная группа привилегий. Роли можно создавать, изменять и удалять. В большинстве организаций имена пользователей создает администратор базы данных или администратор по безопасности, он же назначает пользователям роли, тем самым наделяя их набором привилегий. Сегодня это чаще всего делают из консоли Oracle Enterprise Manager (EM). Например, можно создать роль, предоставляющую доступ к конкретному набору приложений, скажем, «Отдел кадров», или определить несколько ролей, так чтобы изменять почасовую оплату в кадровых приложениях могли только пользователи, приписанные некоторой роли, и больше никто.

В каждой базе данных имеется псевдороль PUBLIC, в которую входят абсолютно все пользователи. Привилегии, включенные в роль PUBLIC, доступны каждому пользователю. Например, если связь баз данных или синоним созданы с ключевым словом PUBLIC, то они будут видны всем пользователям, у которых есть привилегии для доступа к объектам, доступным

через эту связь или синоним.[1] В разделе «Аудит» мы увидим, что привилегия CREATE PUBLIC DB LINK теперь аудируется. Поскольку озабоченность уязвимостью баз данных растет, имеет смысл включить в роль PUBLIC только очень ограниченный набор привилегий.

## Управление идентичностью

Какой бы строгой ни была система безопасности, она не достигнет цели, если плохо администрируется. Чем сложнее административные задачи, тем больше вероятность ошибок, приводящих к появлению брешей в системе. Если вы хотите централизованно управлять доступом к нескольким базам данных, то обратите внимание на подсистему Oracle Identity Management, которая позволяет хранить всю информацию о пользователях и их правах в LDAP-каталоге, таком как Oracle Internet Directory (OID). Например, с помощью OID можно авторизовать соединения пользователей SYSDBA и SYSOPER.

## Привилегии безопасности

С помощью привилегий можно наложить ограничения на четыре основных вида операций с базой данных:

- SELECT для выборки строк
- INSERT для вставки строк в таблицы или представления
- UPDATE для обновления строк в таблицах или представлениях
- DELETE для удаления строк из таблиц или представлений

Помимо этих привилегий, относящихся к работе с данными, есть и несколько других, касающихся объектов внутри схемы базы данных:

- CREATE для создания таблицы в схеме
- DROP для удаления таблицы из схемы
- ALTER для изменения таблиц и представлений

Для работы с привилегиями есть две простые SQL-команды. Команда GRANT служит для предоставления привилегии пользователю или роли, а команда REVOKE - для отзыва привилегии. Эти команды позволяют модифицировать набор привилегий как для отдельного пользователя, так и для роли. Можно также предоставить привилегию на выдачу привилегий. В любой из этих команд можно указать ключевое слово PUBLIC, чтобы предоставить или отозвать привилегию для всех пользователей базы данных.

Еще одна привилегия, EXECUTE, позволяет выполнять процедуру или функцию, написанную на языке PL/SQL. По умолчанию PL/SQL- процедура работает с привилегиями того пользователя, который ее скомпилировал. Но можно указать, что процедура должна работать с *правами вызывающего*, то есть с привилегиями того пользователя, который ее вызвал.

## Специальные роли - DBA, SYSDBA и SYSOPER

В базе данных уже предопределены три специальные роли. Одна из самых важных - роль DBA. В нее включено большинство системных привилегий. По умолчанию она назначается пользователям SYS и SYSTEM, которые создаются на этапе инициализации базы данных. В схеме SYS хранятся таблицы и представления словаря данных. Таблицы из схемы SYSTEM используются для хранения административной информации, а также различными инструментами и опциями Oracle. В зависимости от того, какие компоненты Oracle развернуты, могут быть и другие административные пользователи.

Роль DBA не включает привилегий для выполнения основных административных задач, подразумеваемых системными привилегиями SYSDBA или SYSOPER. Поэтому привилегии SYSDBA или SYSOPER должны быть явно предоставлены администраторам. Подключаясь к базе данных, они выполняют команду CONNECT AS, указывая имя SYSDBA или SYSOPER, и могут таким образом получить доступ даже к закрытой базе данных. Привилегия SYSDBA может быть выдана пользователем SYS или другим пользователем, уже обладающим привилегией SYSDBA. Привилегия SYSDBA позволяет выполнять перечисленные ниже действия из командной строки в SQL*Plus или с помощью графического интерфейса Oracle Enterprise Manager:

*STARTUP*

Запуск экземпляра базы данных.

*SHUTDOWN*

---

[1] Псевдороль PUBLIC и ключевое слово PUBLIC, используемое при создании некоторых типов объектов, не связаны между собой. Роль PUBLIC определяет права доступа, а ключевое слово PUBLIC - область видимости объекта. - ***Прим. науч.ред.***

Останов экземпляра базы данных.

*ALTER DATABASE OPEN*

Открытие смонтированной, но еще не открытой базы данных. *ALTER DATABASE MOUNT*

Монтирование базы данных для уже запущенного экземпляра. *ALTER DATABASE BACKUP CONTROLFILE*

Запуск резервного копирования управляющего файла. Однако чаще резервное копирование выполняется с помощью менеджера RMAN.

*ALTER DATABASE ARCHIVELOG*

Включение режима архивирования содержимого группы журналов перед повторным использованием этой группы.

*ALTER DATABASE RECOVER*

Применение журналов по отдельности или запуск автоматической процедуры применения журналов.

*CREATE DATABASE*

Создание базы данных с указанным именем, определение местоположения и размеров файлов данных и журналов, а также предельных значений параметров.

*DROPDATABASE*
Удаление базы данных и всех файлов, перечисленных в управляющем файле.

*CREATE SPFILE*
Создание файла параметров сервера из текстового описания параметров *(INIT.ORA).*

*RESTRICTED SESSION привилегия*
Разрешение соединения с базой данных, запущенной в ограниченном режиме. Ограниченный режим предназначен для поиска и устранения неполадок и некоторых видов обслуживания, аналогичных тем, что разрешены пользователю SYS.

Администраторам, подключившимся от имени SYSOPER, доступно меньше команд: STARTUP и SHUTDOWN, CREATE SPFILE, ALTER DATABASE OPEN, или MOUNT, или BACKUP, ALTER DATABASE ARCHIVELOG, ALTER DATABASE RECOVER. Кроме того, им предоставлена привилегия RESTRICTED SESSION.

Администраторы базы данных аутентифицируются с помощью механизмов операционной системы или файла паролей. Конструкция CONNECT INTERNAL, имевшаяся в прежних версиях Oracle, больше не поддерживается. Если применяется аутентификация с помощью операционной системы, то имена пользователей с правами администратора должны входить в группы OSDBA или OSOPER. Парольный файл для аутентификации создается утилитой ORAPWD. Новых пользователей может добавлять пользователь SYS или любой, у кого есть привилегия SYSDBA.

## Политики

*Политика* - это способ расширения инфраструктуры безопасности. В политике можно задать дополнительные условия, которые должны проверяться, когда пользователь пытается активизировать роль. Политики пишутся на языке PL/SQL и могут, например, ограничить доступ, разрешив его только для конкретного IP-адреса или в определенное время суток.

Начиная с версии Oracle Database 10*g* в Enterprise Manager появился визуальный интерфейс к инфраструктуре политик, хранящихся в репозитории EM, упрощающий администрирование безопасности базы данных. Политики безопасности, или правила, сохраняются в библиотеке политик. В интерфейсе EM нарушение правил может отображаться как

критическая ошибка, предупреждение или информационное сообщение. По умолчанию о нарушениях безопасности сообщается один раз в день. В соответствии с требованиями бизнеса можно настраивать политики и изменять частоту проверки нарушений.

## Ограничение доступа к данным

Иногда нужно предоставить пользователю доступ к таблице, но разрешить ему просматривать не все хранящиеся в ней данные. Например, одни и те же таблицы могут изучать конкурирующие поставщики. Хотелось бы, чтобы каждый видел лишь товары, поставляемые им самим, и итоговые данные по всем поставщикам, но не детальную информацию о конкурентах. Это можно сделать разными способами, которые мы опишем на примерах из приложения для отдела кадров.

### Безопасность, обеспечиваемая представлениями

*Представление* можно считать своего рода виртуальной таблицей, которая определяется запросом, отбирающим данные из физических *базовых таблиц*. С помощью представлений можно показать только те строки или столбцы, которые разрешено видеть определенной группе пользователей.

Например, у пользователей из отдела кадров может быть полный доступ к базовой таблице работников, в которой хранится как открытая информация (фамилии, рабочие адреса и телефоны), так и закрытая (номера социального страхования, домашние адреса и телефоны). У прочих сотрудников компании не должно быть доступа к закрытой информации, поэтому представление будет включать только общедоступные данные.

Более безопасный способ ограничения доступа к данным дает виртуальная частная база данных или опция Label Security Option. То и другое описано ниже.

### Детальный контрольдоступа

Обеспечение безопасности - очень важная, но отнимающая много времени работа, особенно если безопасность должна быть основана на атрибуте с широким диапазоном значений. В примере с отделом кадров хорошим примером может служить следующая задача: сделать так, чтобы кадровик мог видеть лишь строки, относящиеся к подведомственным ему работникам. Здесь пришлось бы определять по одному представлению на каждого сотрудника отдела кадров, а это слишком много представлений, которые к тому же изменяются всякий раз, как приходит новый сотрудник или увольняется старый. А если требуется разрешить чтение для всех сотрудников, а запись - только для подведомственных, то задача еще усложняется. Чем мельче уровень детализации контроля доступа, тем труднее создавать и поддерживать привилегии.

Для решения подобных задач Oracle предлагает механизм *детального контроля доступа* (fine-grained access control, FGAC). С таблицами или представлениями можно ассоциировать *политики безопасности*, реализованные в виде PL/SQL-функций, создав тем самым виртуальную частную базу данных (virtual private database, VPD). Политика безопасности возвращает условие, которое динамически ассоциируется с SQL-командой и прозрачно ограничивает возвращаемые им данные. В примере с отделом кадров предположим, что каждый сотрудник отвечает за работников, фамилии которых начинаются с букв из определенного диапазона, например от A до G.

Тогда политика безопасности может содержать условие WHERE, ограничивающее множество возвращаемых строк исходя из сферы ответственности конкретного сотрудника. Диапазон букв для каждого кадровика можно хранить в отдельной таблице, которая опрашивается при выполнении функции, реализующей политику безопасности. Это упрощает контроль доступа в случае, когда роли и обязанности часто изменяются.

Политику безопасности можно ассоциировать с представлением или таблицей, воспользовавшись встроенным PL/SQL-пакетом DBMS_RLS, который заодно позволяет обновлять, активизировать и деактивизи- ровать политику безопасности.

В версии Oracle Database 10g VPD можно определить еще более детально, принудительно изменяя запрос к определенному столбцу. Производительность обработки запросов с участием VPD также повысилась за счет распараллеливания. Детализация может основываться и на типе SQL-команды. Описанные выше политики можно применить для накладывания одних ограничений на операции UPDATE, INSERT и DELETE и совсем других - на операции SELECT.

Реализация механизма FGAC с помощью PL/SQL хорошо описана в книгах «Oracle PL/ SQL Programming» Стивена Фейерштейна (Steven Feuerstein) и Билла Прибыла (Bill Pribyl) и «Oracle PL/SQL for DBAs» Арупа Нанды и Стивена Фейерштейна (издательство O'Reilly, подробности приведены в приложении 2).

## Опция Label Security Option

Опция Label Security Option позволяет обойтись без программирования VPD на PL/SQL, сопоставив каждой строке метку безопасности в случаях, когда требуется реализовать разные

уровни секретности. Набор, состоящий из меток, авторизаций меток и правил обеспечения безопасности, можно применить ко всей схеме или к отдельным таблицам.
Метки секретности определяются исходя из того, разрешено ли конкретному пользователю видеть и/или обновлять данные. Метка состоит из уровня, обозначающего степень секретности данных, категории, или отсека (compartment), позволящего выполнить более точное разделение данных, и группы, где хранится информация о владении (которое может быть организовано иерархически) и правах доступа.

Определения стандартных групп позволяют пользователям обращаться кданным, помеченным этими группами. В данных могут применяться и инверсные группы, определяющие, какие метки должны присутствовать в профиле пользователя, чтобы ему был разрешен доступ.

Из EM доступен менеджер политик, позволяющий создать и применить политики, определить метки секретности, установить и авторизовать метки пользователей. С его помощью можно также добавить SQL-предикаты и помечающие функции и управлять доверенными программными единицами, политиками детального управления доступом и контекстами приложений в VPD. Управлять метками безопасности можно начиная с версии Oracle Database 10^, при условии, что установлен каталог Oracle Internet Directory.

## Безопасность, роли и привилегии приложений

Приложение может обращаться к данным и процедурам из разных схем с разными привилегиями. Для решения проблем, обусловленных такой сложностью, в приложениях часто применяются роли. В роль приложения включаются все привилегии, необходимые для его работы, а пользователям этого приложения назначаются его роли.

Роль приложения может содержать привилегии, предоставляемые пользователям только на время работы этого приложения. Разработчик помещает в начале приложения команду SET ROLE, которая устанавливает его роль, временно отменяя все другие роли пользователя. То же самое происходит при вызове процедуры DBMS_SESSION.SET_ ROLE из PL/SQL.

Другой способ обеспечить безопасность приложения - инкапсулировать привилегии в хранимых процедурах. Вместо того чтобы предоставлять прямой доступ к различным таблицам, вы можете создать хранимые процедуры для обращения к этим таблицам и дать доступ к ним, а не к самим таблицам. Например, можно не давать привилегию INSERT для таблицы EMPLOYEE, а написать процедуру HIRE_ EMPLOYEE, которая принимает параметры, необходимые для заведения данных нового работника, и разрешить вызывать ее.

При обычном запуске хранимая процедура выполняется с теми правами доступа, которые предоставлены ее владельцу, то есть той схеме, в которой процедура находится. Если некоторая схема имеет доступ к объекту базы данных, то все находящиеся в ней хранимые процедуры обладают теми же правами, что и сама схема. Пользователь, вызывающий такую процедуру, будет иметь те же права доступа к объектам данных, что и процедура.

Предположим, например, что имеется схема HR_REP. Пусть ей разрешена запись в таблицу EMP. Тогда любая хранимая процедура в схеме
HR_REP тоже может писать в эту таблицу. Следовательно, дав пользователю право исполнять хранимую процедуру в схеме HR_REP, вы разрешаете ему запись в таблицу EMP независимо от привилегий, выданных самому этому пользователю. Однако доступ он будет иметь исключительно через процедуры, находящиеся в схеме HR_REP.

Если при компиляции хранимой процедуры указать ключевое слово AUTHID CURRENT_USER, то ограничения безопасности вычисляются исходя из имени пользователя, вызвавшего процедуру, а не схемы, в которой она находится (автора процедуры). Если пользователь имеет доступ к некоему объекту базы данных, поскольку ему лично предоставлена соответствующая привилегия, то этот доступ сохранится и при обращении через процедуру, откомпилированную в режиме AUTHID CURRENT_USER.

## Распределенная база данных и безопасность в многоуровневой системе

Все механизмы обеспечения безопасности, имеющиеся в стандартных базах данных Oracle, применимы и к распределенным базам. Однако в распределенном окружении возникают дополнительные требования к безопасности. Например, учетные записи, необходимые для установления соединений с серверами, должны существовать во всех базах данных, образующих распределенную систему. При создании связей баз данных (связь определяет соединение между экземплярами распределенной базы) необходимо позаботиться о наличии учетных записей и ролей на каждом узле.

## Управление безопасностью в распределенной системе

В крупной организации бывает необходимо сконфигурировать для пользователей и ролей глобальную аутентификацию в любой распределенной базе. Это позволяет поддерживать единственный список аутентификации для нескольких распределенных баз. Решение проблемы внешней аутентификации дает опция Advanced Security Option, рассматриваемая в следующем разделе.

Информацию о законных пользователях приложений обычно помещают в LDAP-совместимый каталог OID с помощью Enterprise Manager. Пользователь, обращающийся к приложению, для которого он не аутентифицирован, переадресуется на сервер входа в систему. Сервер запрашивает имя и пароль и отдает их на проверку серверу OID. В ответ он получает cookie и переадресует пользователя обратно на приложение.

Для управления безопасностью на разных платформах, поддерживающих собственные механизмы обеспечения безопасности, можно пользоваться подсистемой Oracle Identity Management.

### Безопасность в многоуровневой системе

В типичной трехуровневой системе присутствует сервер приложений Oracle Application Server, где исполняется часть логики приложения. Он играет роль интерфейса между клиентами и серверами баз данных, обеспечивая значительную часть инфраструктуры Oracle Identity Management (OIM). Oracle Internet Directory предоставляет службу каталогов, реализованную поверх базы данных Oracle. В состав OID входят служба синхронизации каталогов, интегрированная служба провизио- нирования и служба делегированного администрирования. В многоуровневых приложениях безопасность обеспечивается привилегиями приложения и сохранением идентичности клиента на всех уровнях.

Как и в случае крупных приложений или веб-приложений, при наличии нескольких уровней часто возникает необходимость в прокси- аутентификации. Приложение вызывает программу на промежуточном уровне, которая обращается к базе данных через прокси, нередко по разделяемому соединению. В некоторых СУБД безопасность ассоциируется с сеансом; это означает, что при смене идентификатора пользователя необходимо организовать новый сеанс. Из-за этого ограничения реализация многоуровневых решений усложняется.

В Oracle аутентификация отделена от сеансов, поэтому применение прокси на промежуточном уровне возможно. В одном сеансе можно поддерживать разных пользователей с разными идентификаторами. До выхода версии Oracle 10g Release 2 эту возможность предоставлял только интерфейс OCI, поэтому приходилось писать много кода. Версия Release 2 упразднила это ограничение, и теперь стандартные инструменты, такие как SQL*Plus, позволяют работать с прокси-аутентификацией.

## Опция Advanced Security Option

Опция Oracle Advanced Security Option (ASO), ранее называвшаяся Advanced Networking Option (ANO), применяется в распределенном окружении на базе Oracle Net, когда требуется обеспечить безопасность доступа и передачи данных. Она позволяет шифровать данные, передаваемые по сети Oracle Net, по протоколам Net/SSL, IIOP/SSL и между тонким JDBC-клиентом и базой данных, для защиты от несанкционированного просмотра. Поддерживаются следующие алгоритмы шифрования: RC4_40, RC4_56, RC4_128, RC4_256, DES, DES_40, 3DES112, 3DES168, AES128, AES192 и AES256. Для защиты пакетов данных от модификации, повторного воспроизведения транзакций и удаления применяются алгоритмы MD5 и SHA-1.
Начиная с версии Oracle Database 10g Release 2 в состав Advanced Security Option включен протокол Transparent Data Encryption (описан в следующем разделе), который обеспечивает простой способ шифрования данных в базе, а имеющийся в ASO механизм шифрования данных в сети защищает данные на этапе их передачи клиенту.

ASO также поддерживает различные методы аутентификации идентификаторов пользователей. Из сторонних служб аутентификации поддерживаются Kerberos, RADIUS и DCE. RADIUS обеспечивает поддержку сторонних устройств аутентификации, например смарт-карт и карт с переменным паролем (token card). Система аутентификации на базе инфраструктуры открытых ключей (Public Key Infrastructure, PKI), широко применяемая в интернет-приложениях электронной коммерции, основана на цифровых сертификатах стандарта X.509 v3 и может пользоваться профилями доверия (Entrust Profiles), хранящимися в бумажниках Oracle Wallets. В версии Oracle Database 10g добавлена возможность аутентификации пользователей, предъявляющих «верительные грамоты» (credentials) Kerberos, причем аутентификация на базе Kerberos работает через границы связей баз данных.

В типичной ситуации подсистема Oracle Enterprise Security Manager сконфигурирована так, что законные пользователи приложений указаны в LDAP-совместимом каталоге OID. Удостоверяющий центр создает пары ключей и публикует их в бумажниках Oracle (с помощью Oracle Wallet Manager), доступных по протоколу LDAP. Чтобы войти в базу данных, пользователю необходимы сертификат и закрытый ключ, который можно извлечь из защищенного паролем бумажника пользователя, находящегося в LDAP-каталоге. Ключ пользователя, посланный клиентским устройством серверу базы данных, сопоставляется с парным ключом, который сервер получает по SSL-соединению из LDAP-каталога, после чего пользователь считается аутентифицированным и получает доступ к базе.

## Шифрование

В предыдущих разделах мы говорили только о защите доступа к данным, хранящимся в базе Oracle. Но иногда требуется дополнительная мера по защите от несанкционированного

просмотра самих данных - шифрование.

Средства шифрования присутствовали в Oracle довольно давно, но только в версии Oracle Database 10g Release 2 появился протокол прозрачного шифрования данных Transparent Data Encryption. Раньше зашифрованные данные, хранящиеся в базе, необходимо было расшифровывать внутри приложения. Из-за этого возникали различные ограничения, самое существенное из которых то, что дешифрированием должно было заниматься приложение. Если вы решали зашифровать какую-то часть данных, то приходилось изменять процедуры доступа во всех приложениях, где использовались эти данные. Одного этого хватало, чтобы задуматься, стоит ли заниматься шифрованием.

С появлением Transparent Data Encryption СУБД стала шифровать и дешифрировать данные автоматически. Oracle шифрует данные, отправляемые в базу, и дешифрирует их, когда данные запрашиваются. Никакой дополнительный код не нужен, следовательно, можно зашифровать существующие данные, не изменяя SQL-команды.

В версии Oracle Database 1*lg* стало возможно шифровать целые табличные пространства, поэтому административные издержки значительно снизились.

## Безопасное резервное копирование Secure Backup

Описанные выше механизмы снабжают вас инструментами, позволяющими безопасно хранить данные в базе Oracle. Но что происходит, когда данные покидают базу, например, при резервном копировании?

Недавние события показали, что потеря и кража лент с резервными копиями - реальность. Опция Secure Backup, выпущенная между версиями Oracle Database 10g* Release 2 и Oracle Database *llg,* позволяет автоматически шифровать резервные копии. Расшифровать данные может только база, в которой они изначально находились, поэтому укравший ленту злоумышленник не сможет узнать, что на ней записано.

## Аудит

СУБД Oracle позволяет запретить неавторизованный доступ к ценным данным. Но любые меры безопасности хороши лишь настолько, насколько строго они соблюдаются, а людям свойственно ошибаться. Кроме того, часто хочется понимать, какие операции - законные или нет - производятся с данными. Аудирование работы с базой решает обе проблемы.

Имеющиеся в Oracle средства аудита позволяют отслеживать действия на уровне команд, привилегий или объектов схемы для всей базы данных или для отдельных пользователей. Кроме того, аудирование обеспечивает сбор данных о работе с базой для планирования и оптимизации. Аудит всех соединений с экземпляром базы данных от имени пользователей с административными привилегиями и всех событий запуска и останова экземпляра включен по умолчанию.

Можно также аудировать сеансы на уровне пользователя, при этом собирается общая, но очень полезная статистика о количестве логических и физических операций ввода/вывода и о суммарном времени работы с базой. Уже отмечалось, что накладные расходы на сбор статистики невелики, а начиная с версии Oracle Database 10g статистика автоматически сохраняется в репозитории Automatic Workload Repository (AWR).

Записи аудита всегда содержат следующую информацию:

- имя пользователя;
- идентификатор сеанса;
- идентификатор терминала;
- имя объекта схемы, к которому осуществлялся доступ;
- операция, которую выполнили или пытались выполнить;
- код завершения операции;
- дата и время.

Эти записи могут храниться в таблице словаря данных (таблица AUD$ в схеме SYS), которую еще называют журналом аудита базы данных, или в журнале аудита операционной системы.

В версии Oracle9i добавился механизм детального аудита, позволяющий включить избирательный аудит запросов SELECT с запоминанием переменных связывания при обращении к определенным столбцам. BOracle DatabaselOg* включена расширенная поддержка детального аудита. Теперь можно аудировать не только запросы SELECT, но и операции UPDATE, INSERT и DELETE.

В версии Oracle Database 1 *lg* аудит по умолчанию включен, а параметр AUDIT_TRAIL равен DB. По умолчанию аудируются следующие привилегии:

- ALTER ANY PROCEDURE

- ALTER ANY TABLE
- ALTER DATABASE
- ALTER PROFILE
- ALTER SYSTEM, ALTER USER
- AUDIT SYSTEM
- CREATE ANY JOB, CREATE ANY LIBRARY, CREATE ANY PROCEDURE, CREATE ANY TABLE, CREATE EXTERNAL JOB, CREATE PUBLIC DB LINK, CREATE SESSION, CREATE USER
- DROP ANY PROCEDURE, DROP ANY TABLE, DROP PROFILE, DROP USER
- EXEMPT ACCESS POLICY
- GRANT ANY OBJECT PRIVILEGE, GRANT ANY PRIVILEGE и GRANT ANY ROLE

## Соответствие требованиям

Лозунг «доверяй, но проверяй» применим и к функциям обеспечения безопасности и аудита. Идею проверки соответствия требованиям можно выразить, дополнив этот лозунг - «доверяй, проверяй и доказывай».

Мы опишем инструменты, необходимые для доказательства того, что данные используются надлежащим образом.

Механизм доказательства основан на описанных выше подсистемах обеспечения безопасности и аудита. По существу, необходимость в этом механизме возникла из-за появления нового элемента в деятельности корпораций - требований правительства. В США и других странах соответствие требованиям все чаще регулируется законодательством, поэтому способность Oracle облегчить решение этой задачи оказалась весьма важна. Доказывать соответствие требованиям совершенно необходимо многим организациям, а люди, отвечающие за это, необязательно работают в ИТ-департаменте. Поэтому реализацию схем безопасности и аудита пришлось упростить и увязать с новыми потребностями.

В Oracle есть две опции, специально предназначенные для доказательства соответствия, - Oracle Database Vault Option Option и Oracle Audit Vault Server, описанные ниже.

## Oracle Database Vault Option

Опция Oracle Database Vault Option появилась в 2006 году. Она ограничивает возможность администратора базы данных и других привилегированных пользователей обращаться к данным, к которым у них не должно быть доступа. Ее можно настроить так, что администратор одного приложения не сможет манипулировать данными, относящимися к другим приложениям. Администратор по безопасности может с помощью Oracle Database Vault Option описать схему обеспечения безопасности, принятую в организации, а система автоматически реализует эту схему, пользуясь описанными выше средствами.

Ключевые параметры, определяемые в Oracle Database Vault Option, называются *факторами*. По существу, это описательная информация, отражающаяся на безопасности всей базы данных. Фактором может быть конкретная прикладная программа, местоположение или время суток. В комплекте поставки есть более 40 готовых факторов, и пользователи могут определять дополнительные.

Факторы служат для определения прав доступа и аудирования различных характеристик безопасности. Можно создать правила, ограничивающие доступ к конкретному фактору, и наборы, объединяющие несколько правил. После того как наборы правил заданы, можно создать на их основе роли приложений, а также командные правила, определяющие, разрешено ли выполнять некоторые команды в базе данных, исходя из результата вычисления правила. Например, можно запретить удаление таблицы, если команда не исходила из конкретного места, определяемого фактором, или сказать, что новый пользователь может быть создан только совместными усилиями двух администраторов.

С помощью правил можно определить *область* (realm) базы данных, состоящую из подмножества схем и ролей, подведомственных конкретному администратору. Это существенно, если организация использует базу данных Oracle для обслуживания нескольких сообществ. Можно определить некую область и предоставить администратору привилегии в этой области, не подвергая риску данные в других схемах. Смысл областей заключается в том, чтобы обеспечить безопасное делегирование административной ответственности.

## Oracle Audit Vault Server

Oracle Audit Vault Server был представлен в 2007 году. Он собирает данные из журналов аудита Oracle и операционной системы. Данные объединяются в безопасном репозитории, а на выходе формируются отчеты о соответствии требованиям, в частности о доступе со стороны

привилегированных пользователей, об управлении учетными записями, о доступе к данным и о неудачных попытках входа в систему. Данные находятся в схеме хранилища данных и могут быть без труда обработаны различными инструментами бизнес-анализа, например Oracle BI Publisher.

Поскольку Oracle Audit Vault Server следит за всеми источниками данных аудита, то может генерировать предупреждения исходя из заданных политик, например, об изменении состава привилегированных пользователей или о доступе к секретным данным. Мониторинг возможен для версий СУБД не ниже Oracle 9*i* Release 2. Для построения специализированных сборщиков данных аудита предлагается комплект средств разработки ПО (SDK).

### Flashback Data Archive

Технология ретроспекции (Flashback) основана на сегментах отката. Впервые эта технология появилась в версии Oracle9i, а в Oracle Database *llg* она нашла полезное применение в доказательстве соответствия требованиям.

Решение Flashback Data Archive позволяет увидеть все изменения, происходившие с записью за все время ее существования. Эта история содержит важнейшую информацию, необходимую как для демонстрации соответствия требованиям, так и для выявления источника ошибок.

## Лекция 7. Производительность Oracle

### Основы настройки производительности

Производительность - один из самых сложных аспектов эксплуатации базы данных, так как на нее влияет множество факторов. Прежде всего, конечно, сама база данных. Но надо принимать во внимание и стратегии развертывания. В наши дни инфраструктура, скорее всего, размещается на нескольких платформах, включая серверы базы данных и приложений. Их связывает сеть, пропускную способность которой надо учитывать. Сложность решаемых пользователями задач тоже варьируется.

Производительности присущ один любопытный аспект; «хорошая производительность» - это, скорее, отсутствие, чем присутствие. Плохая производительность видна сразу, а хорошая определяется как отсутствие плохой. Тема производительности одновременно очень проста - любому новичку интуитивно понятно, о чем речь, и исключительно сложна - самым опытным администраторам баз данных иногда приходится проявлять недюжинную изобретательность.

Прежде чем начать обсуждать специфику производительности в Oracle, имеет смысл описать базовую методологию исследования связанных с ней проблем.

В контексте СУБД Oracle подход к вопросам производительности включает три основных шага:

1. Определить, что такое производительность и что такое проблема с производительностью.
2. Проверить производительность серверного ПО Oracle.
3. Проверить общую производительность машины, на которой работает сервер.

### Что такое производительность и как проявляются проблемы с ней

Первый шаг процедуры настройки производительности - понять, а есть ли *вообще* проблема. Выше мы сказали, что о плохой производительности чаще всего сообщают пользователи. Но что же такое плохая производительность?

Жалобы на плохую производительность - всегда результат разочарования; пользователю кажется, что система работает не так быстро, как он ожидает. Следовательно, прежде всего надо оценить, насколько реалистичны его ожидания.

Если ожидания небеспочвенны, например раньше система работала быстрее, и снижение скорости отразилось на бизнесе, то вам предстоит выяснить, какие компоненты привели к проблеме. Необходимо уточнить смысл фразы «система тормозит» и понять, какие операции выполняются слишком медленно, что означает «слишком медленно» и при каких условиях происходит замедление. Например, проблема может проявляться только для определенных транзакций и в определенное время, или для всех транзакций, или отчеты формируются дольше, чем рассчитывал пользователь.

Поняв, какой производительности пользователи ожидают от системы, можно начать выяснять, в чем заключается проблема. Вообще, все проблемы с производительностью объясняются тем, что спрос на некий ресурс выше, чем предложение этого ресурса, поэтому приложения вынуждены ждать, пока ресурс освободится.

### Производительность сервера Oracle

Начать поиск узких мест стоит с программного обеспечения сервера, пользуясь для этого программой Oracle Enterprise Manager. Задача - найти места, где внутренние ресурсы Oracle

используются не оптимально. Такая ситуация приводит к тому, что сеансы без необходимости ждут обслуживания, и настройка производительности должна быть нацелена на расшивку этих узких мест.

Динамические представления Oracle способны пролить свет на узкие места в вашей базе данных. До появления средств Automatic Workload Repository (AWR), Automatic Database Diagnostics Monitor (ADDM) nOracle Enterprise Manager Grid Control в версии Oracle Database 10g администраторы начинали с опроса представлений, относящихся к производительности. Имена всех таких представлений начинаются с префикса V$, а в версии Oracle9i появились еще и глобальные представления (для всех узлов кластера Real Application Cluster) с именами, начинающимися с GV$. Для идентификации источников задержек особенно полезны следующие представления, с которых следует начинать анализ проблемы:

*V$SYSTEM_EVENT*
: Дает агрегированную общесистемную информацию о ресурсах, которых ждет весь экземпляр.

*V$SESSION_EVENT*
: Дает накопительный список событий, которые приходилось ждать в каждом сеансе.

*V$SESSIONJVAIT*
: Дает детальную посеансовую информацию о ресурсах, которые сеанс ожидает в данный момент или ждал в последний раз.

*V$SESSION*
: Дает информацию о каждом сеансе, в том числе о событии, которое он ждет в данный момент или ждал в последний раз.

Эти представления позволяют выявить ресурсы, которых приходится ждать чаще всего. Пристальное внимание к этим ресурсам позволит добиться максимального повышения производительности.

Возможно, вы обнаружите, что причина проблемы тривиальна, например, коэффициент попаданий в кэш ниже ожидаемого. Коль скоро кэш работает не оптимально, можно попытаться увеличить параметр инициализации DB_BLOCK_BUFFERS, что приведет к увеличению размера кэша и, возможно, к повышению коэффициента попаданий. Этот показатель вы найдете в представлении V$METRICNAME.

Не всегда удается так легко очертить проблему, изучая параметры, доступные через представления. Например, вы обнаружили, что на выборку строк с диска уходит сравнительно много времени. Причина может заключаться в конкуренции за диски сервера или в неоптимальном размещении файлов Oracle на дисках или в помехах со стороны других приложений.

## AWR, ADDM и Enterprise Manager

Сегодня гораздо более эффективен подход с применением программы Enterprise Manager (в системах на базе RAC ее называют также Grid Control) как отправной точки для мониторинга и управления производительностью. Информация об использовании ресурсов накапливается в автоматическом репозитории рабочей нагрузки (Automatic Workload Repository, AWR). По умолчанию статистические данные собираются каждые 30 минут и хранятся 7 дней. Доступ к статистике дают представления, однако Enterprise Manager предлагает гораздо более удобный интерфейс.

AWR помогает выявлять потенциальные проблемы путем сравнения рабочей нагрузки за период времени. Он же служит основой для многих средств улучшения управляемости, появившихся начиная с версии Oracle Database 10g. Среди них - автоматический диагностический монитор базы данных (Automatic Database Diagnostic Monitor, ADDM).

ADDM автоматически выявляет и сообщает об узких местах, например, о конкуренции за процессор, о блокировках или о низкой производительности отдельных SQL-команд. В Oracle Database llg ADDM может анализировать работу всего кластера. Уведомления, которые ADDM посылает на инструментальную панель Enterprise Manager, могут указать на причину конкуренции в момент ее появления. Enterprise Manager предоставляет сводные и детальные сведения о потреблении ресурсов серверами Oracle, а из них вы можете быстро сделать выводы о причинах проблем с производительностью. Можно установить пороговые значения, и инструментальная панель будет уведомлять вас о том, что уровень потребления некоторого ресурса приближается к критическому. В состав Enterprise Manager входят различные консультанты, готовые подсказать, как лучше настроить приложения или оптимизировать производительность базы данных.

Для настройки приложений лучше всего подходит консультант SQL Advisor. Впервые появившись в версии Oracle Database llg, он объединяет функциональность SQL Tuning Advisor,

SQL Access Advisor и нового консультанта Partition Advisor. SQL Advisor вырабатывает рекомендации на основе хранящейся в AWR информации о потреблении процессора и ввода/вывода и сведений о проблематичных SQL-командах, обнаруженных ADDM. Консультант проверяет, что статистика не устарела, находит оптимальные пути, прибегая к профилированию SQL, определяет, будет ли эффективно добавление индексов, материализованных представлений и других структур данных, и подсказывает, какие изменения в особо нагружающих систему SQL-командах позволили бы повысить эффективность.

Перечислим основные консультанты по настройке базы данных. *Memory Advisor*

Полезен для оптимальной настройки параметров MEMORY_TAR- GET (автоматическое управление памятью в версии Oracle Database 11 *g*) и SGA_TARGET (управление разделяемой памятью).

*Segment Advisor*

Полезен для управления системой хранения во внешней памяти и выделения места.

*UndoAdvisor*

Полезен для управления транзакциями.

Есть и другие консультанты, такие как Mean Time to Recovery (MTTR) Advisor, помогающие оптимизировать конфигурацию Oracle, в том числе использование журналов.

## Использование машинных ресурсов

Проблемы с производительностью могут начаться и тогда, когда машине, где работает сервер базы данных, не хватает ресурсов. Если СУБД Oracle плохо развернута с самого начала, то добавление машинных ресурсов может поначалу способствовать расшивке узких мест, но это дорогостоящий способ решения проблемы. К тому же проблема, скорее всего, проявится вновь, когда и дополнительные ресурсы будут исчерпаны. Но если база данных правильно спроектирована и сконфигурирована, то добавление машинных ресурсов в случае их нехватки может прийтись кстати.

Производительность базы данных зависит от того, как используются имеющиеся машинные ресурсы. Речь идет о процессоре, оперативнойпамяти, подсистеме дискового ввода/вывода и пропускной способности сети. Может оказаться, что большая часть проблем объясняется нехваткой одного или нескольких из этих ресурсов.

Пропускная способность сети между сервером и клиентом в наши дни перестала быть серьезной проблемой, поскольку в большинстве организаций ее вполне хватает. При развертывании кластера следует обратить внимание на пропускную способность межсерверных линий, так как слишком напряженный трафик может привести к снижению производительности. Но быстродействие таких линий тоже растет, так что пропускная способность сети при правильном выборе проектных решений вряд ли станет камнем преткновения.

Памятуя об этом, далее мы уделим внимание тому, как Oracle использует три главных машинных ресурса: процессор, память и подсистему дискового ввода/вывода. Диск - самый медленный компонент системы, и большинство проблем производительности связано как раз с вводом/выводом.

Серверная машина может «тормозить» из-за конкуренции за несколько ресурсов. Вычислительные системы проектируются так, что нехватка одного ресурса может компенсироваться другим, что иногда влечет за собой и дефицит компенсирующего ресурса. В случае нехватки физической памяти операционная система выгружает часть ее содержимого на диск, что может привести к затору в подсистеме ввода/ вывода.

Понять, как используются машинные ресурсы, можно с помощью Oracle Enterprise Manager, инструментов, поставляемых производителем компьютера, или утилит операционной системы. В версию Enterprise Manager 10g включен анализатор производительности Automatic Performance Monitoring (APM). APM позволяет настроить *сигнальщики* (beacons) - клиентские процессы, которые периодически выполняют транзакции и сообщают о времени отклика. Это дает возможность оценить производительность с точки зрения пользователя и помогает выявить источники проблем, не связанные с работой сервера Oracle, например замедление передачи по сети.

## Когда уже ничего не помогает...

Если настройка не дает желаемых результатов, это может означать, что проблемы с производительностью вызваны приложением. Например, для ускорения ввода/вывода вы можете попробовать изменить расслоение дисков или повысить пропускную способность дисковой подсистемы. Но если ситуация обусловлена плохо написанным SQL- запросом, то лучше его переписать.

Дойдя до этой точки, вы должны проанализировать взаимодействие отдельных модулей сервера

базы данных и SQL-запросов, предъявляемых приложением. Возможно, обнаружится, что производительность снижается из-за нескольких отдельных запросов. Но скорее всего, придется переработать все приложение.

Enterprise Manager совместно с монитором Automatic Database Diagnostic Monitor (ADDM) могут автоматически выявить SQL-команды, которые потребляют большую часть ресурсов или ведут себя не оптимально, а консультант SQL Tuning Advisor способен даже предложить решение выявленных проблем.

Производительность непосредственно отражается на работе организации. Пытаясь решить проблему производительности, обязательно следите за поведением тех компонентов, которые вы стремитесь улучшить, - как до, так и после внесения изменений. В качестве эталона для сравнения используйте статистические данные, собираемые AWR о приложениях, базе данных, операционной системе, дисковом вводе/ выводе и работе сети.

Вы должны выработать систематический подход как к поиску источника проблемы, так и к реализации ее решения. Этот подход подразумевает сбор опорных данных об использовании ресурсов и времени отклика до внесения изменений. Сами изменения следует вносить мелкими порциями, каждый раз наблюдая за производительностью. Очень соблазнительно попытаться решить проблему одним махом, безо всяких измерений, но такая тактика обычно лишь приводит к новым проблемам.

В версии Oracle Database 1 lg выполнять такое сравнение производительности стало гораздо проще. Вы можете сохранить в качестве опорных данные, собранные AWR за определенный период. В качестве опорных можно брать данные за фиксированный период или в скользящем временном окне.

## Oracle и подсистема дискового ввода/вывода

С точки зрения машинных ресурсов, операцию ввода/вывода можно определить как считывание или запись операционной системой байтов в дисковой подсистеме сервера базы данных. Данные можно считы- вать/записывать мелкими порциями, скажем, по 4 Кбайт, или крупными - 64 или 128 Кбайт. Нижний и верхний пределы одной операции ввода/вывода зависят от операционной системы. В СУБД Oracle также имеется понятие *блока базы данных*, размер которого вы можете задать.

Сервер Oracle посылает подсистеме ввода/вывода запросы, в основном, двух видов:

*Ввод/вывод одиночного блока базы данных*
: Например, одного блока размером 8 Кбайт. В этом случае считывается или записывается один конкретный блок. Скажем, найдя строку в индексе, Oracle считывает нужный блок базы данных в память посредством одноблочного ввода/вывода.

*Многоблочный ввод/вывод*
: Например, 32 блоков базы данных по 8 Кбайт каждый. Общий объем ввода/вывода составит 256 Кбайт. Многоблочный ввод/вывод применяется в таких масштабных операциях, как полное сканирование таблицы. Количество блоков в одном запросе задается параметром инициализации DB_FILE_MULTIBLOCK_READ_COUNT.

Поскольку с помощью многоблочного ввода/вывода Oracle может читать большие объемы данных, то иногда полное сканирование таблицы оказывается быстрее поиска по индексу (особенно если избирательность индекса невелика). Многоблочная операция выполняется быстрее, чем эквивалентная последовательность одноблочных операций.

## Принципы планирования ввода/вывода в СУБД Oracle

При планировании разбиения диска и последующем размещении различных файлов, составляющих базу данных, следует принимать во внимание, для чего Oracle выполняет ввод/вывод и как различные причины влияют на производительность.

Вот перечень основных структур, служащих объектами ввода/вывода:

- журналы;
- данные в таблицах;
- индексы над таблицами;
- словарь данных, который находится в табличном пространстве SYSTEM;
- область сортировки, находящаяся в табличном пространстве TEMP того пользователя, который инициировал сортировку;
- информация для отката, разбросанная по файлам данных в табличном пространстве, которое содержит сегменты отката;
- архивные журналы, сохраняемые в указанных каталогах (в предположении, что база данных работает в режиме ARCHIVELOG).

Следующие простые принципы управления этими видами ввода/вывода помогут оптимизировать использование дисковой подсистемы сервером Oracle:

*Применяйте технологии расслоения дисков, чтобы равномерно распределить ввод/вывод по нескольким накопителям*

Эти технологии подробно описаны ниже в разделе «Технология дисковых массивов RAID». В Oracle Database 10g и последующих версиях процедура расслоения упрощена за счет того, что Enterprise Manager пользуется подсистемой ASM.

*Применяйте табличные пространства для четкого разделения различных видов ввода/вывода*

Отделяйте ввод/вывод в таблицы от ввода/вывода в индексы, помещая эти структуры в различные табличные пространства. Впоследствии файлы данных для этих табличных пространств можно будет перенести на отдельные диски, чтобы обеспечить более высокую производительность при конкурентном доступе.

Использование табличных пространств для разделения объектов также упрощает последующую настройку. В Oracle ввод/вывод реализуется на уровне файла данных, или физического объекта, который операционная система воспринимает как файл, а каждый такой файл является частью только одного табличного пространства. Помещение объектов в разные табличные пространства позволит точно измерить интенсивность ввода/ вывода для этих объектов и при необходимости перенаправить его, переместив файлы в другое место.

Рассмотрим, к примеру, базу данных с несколькими большими, активно используемыми таблицами. Если поместить б.ольшие таблицы в одно табличное пространство, то будет трудно определить, обращения к какой именно таблице вызывают особо интенсивный ввод/вывод в файлы данных. Разделение же объектов позволит вести мониторинг ввода/вывода, ассоциированного с каждым из них. В документации по Oracle рассматриваются и другие факторы, которые следует учитывать при отображении объектов на табличные пространства.

*Помещайте журналы и зеркальные копии журналов на разные и наименее загруженные устройства*

Такое размещение максимизирует пропускную способность в транзакционных системах. Oracle пишет во все копии журналов, и операция ввода/вывода завершается лишь после успешной записи. Если одна копия журнала находится на медленном устройстве, а другая - на быстром, то производительность ввода/вывода в журналы будет лимитирована скоростью медленного устройства.

В версии Oracle Database 10g* Release 2 есть возможность откладывать операции записи в журнал для транзакций. Это повышает производительность в системах, где выполняется очень много транзакций, но сопряжено с риском потери зафиксированных данных в случае краха системы.

*Распределяйте «системные издержки» равномерно по всем имеющимся накопителям*

Под системными издержками понимается ввод/вывод в табличное пространство SYSTEM, где находится словарь данных, в табличное пространство TEMP при сортировке и в табличные пространства, содержащие сегменты отката. Следует изучить, как системные издержки распределяются по накопителям. Например, если приложение чаще изменяет, чем читает данные, то нагрузка на сегменты отката может возрасти из-за большого количества операций записи изменений и операций считывания, необходимых для обеспечения согласованного чтения.

Сортировка тоже может влиять на дисковый ввод/вывод. До версии Oracle Database 10g изменением параметра инициализации SORT_ AREA_SIZE можно было добиться, чтобы большинство операций сортировки выполнялось в памяти. Oracle постоянно опрашивает и обновляет словарь данных, хранящийся в табличном пространстве SYSTEM, и эта информация кэшируется в разделяемом пуле в SGA, поэтому правильный выбор размера разделяемого пула - ключ к повышению общей производительности. В версии Oracle Database 10g* размеры пулов в SGA устанавливаются и изменяются динамически.

*Храните оперативные и архивные журналы на разных устройствах*

Во избежание проблем, связанных с конкурентным вводом/выво- дом, размещайте архивные журналы не на тех устройствах, куда пишутся оперативные журналы и их зеркальные копии.

А вот ряд соображений, которые следует учитывать для обеспечения доступности базы данных:

*Если вы сохраняете резервные копии непосредственно на диске, то выбирайте для них устройство, не содержащее больше никаких компонентов базы данных*

Это послужит защитой от одновременной утраты базы и копий в случае сбоя устройства ввода/вывода.

*Устройство, на которое записываются архивные журналы, не должно содержать никаких компонентов базы данных или резервных копий*

Если при сбое одного устройства теряются одновременно компоненты базы данных и архивные журналы или резервные копии и архивные журналы, то возможность восстановления оказывается под угрозой.

Отказоустойчивые дисковые массивы не отменяют необходимости в надежной стратегии резервного копирования и восстановления. Эта технология лишь понижает вероятность того, что базу данных придется восстанавливать после сбоя одиночного устройства.

## Технология дисковых массивов RAID

Один из самых действенных способов расшивки узких мест, связанных с производительностью ввода/вывода, - применение RAID-массивов. RAID (Redundant Array of Inexpensive, или Independent Disks) - это массив недорогих (или независимых) дисков с избыточностью. Есть две причины объединять группы дисков в массивы - резервирование и производительность.

Смысл объединения дисков в массивы заключается в том, что операции ввода/вывода автоматически распределяются по разным накопителям, уменьшая тем самым конкуренцию за отдельные диски. Представим, например, что вы поместили файл данных, содержащий индекс, на отдельный диск. Если к этому индексу одновременно обращаются несколько процессов, то все запросы ввода/вывода будут поставлены в очередь к одному накопителю, то есть возникнет конкуренция.

Но допустим, что тот же самый файл помещен на «диск», который на самом деле представляет собой массив из пяти дисков. Каждый физический диск в массиве способен независимо выполнять ввод/вывод различных блоков индекса и, следовательно, Oracle может выполнить больше операций ввода/вывода в единицу времени без конкуренции.

Само по себе применение дисковых массивов не даст вам оптимальную производительность ввода/вывода. Как уже отмечалось, вы должны разумно разместить различные виды файлов Oracle на имеющихся дисках, даже если они сгруппированы в массивы. С появлением версии Oracle Database 10g вопрос о расслоении решает подсистема Automatic Storage Management. ASM автоматически выполняет расслоение и перебалансировку полос. Поумолчанию ASM также обеспечивает зеркалирование.

### Менеджеры томов

Если применяется программное расслоение, то на сервере базы данных работает специальное программное обеспечение управления логическими томами. В старых версиях Oracle для этой цели частенько применялись продукты Logical Volume Manager (LVM) компании Hewlett Packard и Volume Manager компании Veritas Software. LVM играет роль интерфейса между операционной системой, запрашивающей ввод/ вывод, и физическими дисками. Менеджер томов объединяет диски в массив, который для операционной системы выглядит единым диском. В реальности диски обычно представляют собой либо отдельные устройства, подключенные к контроллерам, либо заранее собранный массив из нескольких дисков и контроллеров. Менеджер томов производит расслоение абсолютно прозрачно для Oracle.

В версию Oracle9i Release 2 для Linux и Windows включен менеджер томов, разработанный корпорацией Oracle. Начиная с версии Oracle Database 10g версии СУБД для всех поддерживаемых платформ содержат кластерную файловую систему и менеджер томов, которым пользуется подсистема ASM. При работе с ASM не рекомендуется применять менеджер томов, входящий в состав операционной системы.

### Специализированные запоминающие устройства

Системы, состоящие из специализированных запоминающих устройств, часто называют *дисковыми фермами* (disk farms). Они содержат диски, контроллеры, процессоры и (обычно) память, а используются в качестве кэша для подсистем ввода/вывода. Такие системы поставляют компании EMC, Network Appliance, Hewlett-Packard, IBM и Sun. Они избавляют сервер базы данных от необходимости управлять дисковыми массивами. Подсистема ввода/вывода подключается к серверу через контроллеры. Иногда такие специализированные устройства объединяются в *сети устройств хранения данных* (storage area network, SAN), дабы подчеркнуть, что логически они организованы как отдельный набор связанных сетью устройств. Внутри специализированной подсистемы ввода/вывода определяются дисковые массивы, аобразующиеся врезультате логические «диски» операционная система видит как обычные физические диски.

Такой способ управления дисковыми томами абсолютно прозрачен для сервера базы данных и обладает рядом преимуществ:

- сервер не тратит ресурсы процессора на управление дисковыми массивами;
- подсистема ввода/вывода кэширует ввод/вывод в памяти, в результате чего

производительность ввода/вывода в Oracle существенно возрастает (в среднем время одной операции снижается с 10-12 до 3-5 миллисекунд);

- операция записи считается завершенной, как только данные успешно записаны в кэш подсистемы ввода/вывода;
- подсистема ввода/вывода может сбросить данные из кэша на физический диск позже;
- считываемые данные могут уже находиться в кэше. Подсистема ввода/вывода может с помощью различных алгоритмов обнаруживать паттерны ввода/вывода и заранее загружать данные в кэш, предвидя, что они понадобятся в ближайшем будущем.

Отметим, что необходимо предусмотреть резервное питание кэша от батарей, чтобы случайный сбой питания не привел к потере данных, записанных в кэш, но еще не сброшенных на физический диск. В противном случае данные, которые Oracle считает записанными на диск, могут быть утрачены, и это приведет к порче базы данных. На рис. 7.2 показан сервер базы данных со специализированной подсистемой ввода/вывода.

### Комбинация программного управления томами и подсистемы ввода/вывода

В такой конфигурации диски сначала объединяются в массивы внутри подсистемы ввода/вывода, а затем - в более крупные массивы на уровне менеджера томов операционной системы. Например, в решении, предлагаемом компанией EMC, физические диски объединяются в зеркалированные пары на базе RAID-1 или в расслоенные массивы с четырьмя дисками в полосе на базе RAID-S. (Термин RAID-S применяется компанией EMC, http://www.emc.com, для обозначения разработанного ею программного и аппаратного обеспечения расслоения.)

В примере с технологией EMC операционная система видит горизонтальные секции общего дискового пространства, объединенного в диск RAID-1 или в массив RAID-S, как отдельные «диски». С помощью менеджера томов эти «диски» можно сгруппировать в массивы. В случае дисков RAID-1 достоинством такой конфигурации является сочетание кэша и вычислительных средств специализированной подсистемы ввода/вывода с простотой механизма расслоения. В случае массивов RAID-S вы получаете все выгоды от применения специализированной подсистемы ввода/вывода и при этом можете упростить управление дисками, увеличив коэффициент расслоения. Массив из пяти «дис ков», видимых операционной системе, может на самом деле отображаться на пять массивов по четыре диска в каждом на уровне подсистемы ввода/вывода. В результате один видимый Oracle логический диск состоит из 20 физических дисков в подсистеме ввода/вывода.

## Гибкость, управляемость и дисковые массивы

Во многих современных системах несколько дисковых накопителей объединяются в массивы с помощью той или иной разновидности технологии RAID. Затем при планировании ввода/вывода каждый дисковый массив рассматривается как единый логический диск. Техника расслоения позволяет распределить ввод/вывод по нескольким дискам, не тратя усилий на администрирование множества отдельных накопителей.

## О взаимодействии ввода/вывода и расслоенных массивов в Oracle

Почти во всех крупных базах данных расслоение дисков позволяет повысить производительность дискового ввода/вывода, не возлагая на администратора обязанность управлять множеством файлов данных, размещенных на многих отдельных дисках. Организовать RAID-массивы можно с помощью программного менеджера томов, специализированной подсистемы ввода/вывода или их комбинации.

Работая с версией Oracle, в которой еще нет ASM, при создании расслоенных дисковых массивов можно задать *размер порции* (chunk- size). Это количество данных, записываемых на один диск, перед тем как система переходит к следующему диску в массиве. Для оптимизации производительности ввода/вывода очень важно понимать взаимосвязь между размером порции и еще двумя размерами ввода/вывода Oracle.

Рассмотрим базу данных, в которой размер блока - 8 Кбайт, а значение параметра инициализации DB_FILE_MULTIBLOCK_READ_CO- UNT “ 32. Тогда в одноблочном режиме Oracle будет читать по 8 Кбайт, а в многоблочном режиме - по 256 Кбайт (32x8 Кбайт). Предположим, вы сконфигурировали массив из четырех дисков с размером порции 64 Кбайт, так что 256 Кбайт распределены по четырем накопителям - по 64 Кбайт на каждом.

Каждая операция ввода/вывода 8 Кбайт будет адресована одному накопителю, так как 8 Кбайт укладываются в порцию величиной 64 Кбайт. Расслоение может повысить производительность коротких операций ввода/вывода за счет улучшения конкурентности: каждый диск будет обслуживать различные запросы. Многоблочная операция ввода/вывода 256 Кбайт может затронуть все четыре диска. Если бы размер порции был не 64 Кбайт, а 256 Кбайт, то в среднем

каждая операция по 256 Кбайт была бы адресована одному диску. В этом случае для многоблочного ввода/вывода требуется меньше операций при большем размере порции. В любом случае операции ввода/вывода одного блока данных удовлетворяются одним диском. Расслоение может повысить производительность ввода/вывода при считывании большого объема данных за счет распределения одной операции на несколько дисков, что было проиллюстрировано на примере порции размером 64 Кбайт и многоблочной операции чтения 256 Кбайт.

## Oracle и параллелизм

Умение распараллеливать операции - одна из важнейших особенностей сверхбольших баз данных (Very Large Database, VLDB). Сейчас считается нормой, когда сервер базы данных работает на компьютере снесколькими процессорами - так называемом симметричном мультипроцессорном компьютере (symmetric multiprocessing, SMP), но механизм распараллеливания отлично работает и с многоядерными процессорами в одном чипе. По мере возрастания требований к производительности и роста объема данных вы все больше будете склоняться к использованию нескольких процессоров и дисков для ускорения решения задач. Oracle поддерживает параллелизм как на одном SMP- сервере, так и на нескольких узлах — с применением технологии Oracle Real Application Clusters. Параллельное выполнение SQL-команды потребляет больше машинных ресурсов - времени процессора, памяти, ввода/вывода, - но завершается быстрее.

Потребность в ресурсах (память, процессорное время), необходимых для параллельного выполнения задачи, практически пропорциональна числу параллельных процессов. Для каждого параллельного процесса выделяется программная глобальная область (PGA). Каждый параллельный процесс потребляет кванты процессора, но чем больше параллельных процессов, тем меньше общее время ожидания завершения дискового ввода/вывода, а именно оно - причина большинства узких мест.

В СУБД Oracle реализовано два вида параллелизма:

*Параллелизм по диапазонам блоков данных*

Управляется диапазонами блоков данных.

*Параллелизм по секциям*

Управляется количеством секций или подсекций, участвующих в операции.

Оба вида параллелизма описаны в следующих разделах.

## Параллелизм по диапазонам блоков

В 1994 году в версии Oracle 7.1 появилась возможность динамически распараллеливать операции сканирования таблиц и вычисления различных функций, основанных насканировании. Этот механизм был основан на понятии *диапазонов блоков* - сервер Oracle знал, что каждая таблица содержит диапазон блоков, в которых хранились данные из определенного диапазона. Параллелизм по диапазонам блоков в Oracle? был реализован путем динамического разбиения таблицы на куски, каждый из которых рассматривался как диапазон блоков, с последующим запуском нескольких процессов для параллельной обработки этих кусков. Такая реализация была уникальной в том смысле, что не требовала физического секционирования таблиц.

Клиентский сеанс, в котором была предъявлена SQL-команда, прозрачно становился координатором параллельного выполнения. Он динамически определял диапазоны блоков и назначал их набору параллельно выполняемых процессов (PE-процессов). Завершив обработку своего диапазона блоков, PE-процесс обращался к координатору за новой работой. Не все операции ввода/вывода выполняются с одинаковой скоростью, поэтому некоторые PE-процессы успевали обработать больше блоков, чем другие. Идея «перехвата работы» позволяет всем процессам наравне участвовать в выполнении задания, обеспечивая тем самым максимальное использование машинных ресурсов.

Параллелизм по диапазонам блоков линейно масштабируется путем увеличения количества PE-процессов при условии, что аппаратных ресурсов хватает. Ключ к масштабируемости в этом случае - простое наращивание аппаратных средств. Каждый PE-процесс работает на своем процессоре и посылает устройствам запросы на ввод/вывод. Если процессоров и дисков достаточно, то степень параллелизма будет выше. Если каких-то ресурсов не хватает, пострадает масштабируемость.

Например, наличие четырех процессоров, читающих два диска, не увеличит производительность по сравнению с двумя процессорами и может даже привести к снижению, если дополнительные процессоры начнут конкурировать за диски. Аналогично, наличие двух процессоров, обращающихся к 20 дискам, не даст 20-кратного повышения производительности. Эффект от параллелизма будет достигнут лишь при сбалансированном составе оборудования.
В большинстве крупных систем количество дисков намного превышает количество процессоров. В таких случаях распараллеливание приводит к случайному распределению ввода/вывода по всей подсистеме вво- да/вывода. Это полезно, когда имеется конкурентный доступ к данным, поскольку PE-процессы, запущенные от имени разных пользователей, читают данные с разных дисков в разные моменты времени, способствуя распределению нагрузки по всем имеющимся дискам.

Хорошая аналогия динамическому параллелизму - поедание пирога. Пирог можно считать набором блоков, которые нужно «прочитать», цель - съесть пирог как можно быстрее при заданном количестве едоков. Oracle раздает пирог порциями. Как только кто-то доел одну порцию, он может подойти за следующей. Не все едят одинаково быстро, поэтому кому-то достанется больше пирога. Вреальном мире такой подход трудно назвать справедливым, но он неплохо моделирует параллелизм, так как если все едят одновременно, пирог исчезнет быстрее. Альтернатива - дать каждому едоку по одинаковой порции и дожидаться, когда закончит самый медлительный. На рис. 7.5 показано разбиение набора блоков на диапазоны.

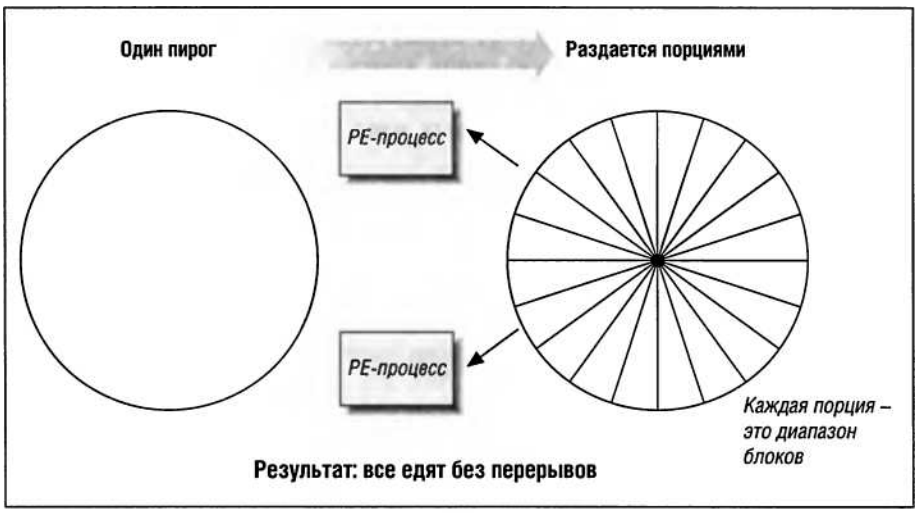



***Рис. 7.5. Динамический параллелизм по диапазонам блоков***

## Параллелизм для таблиц и секций таблиц

С появлением в версии Oracle8 *секционированных таблиц* операция может распространяться на одну, несколько или все секции. Между обычными и секционированными таблицами нет существенной разницы в том, как набор блоков динамически разбивается на диапазоны для параллельной обработки. После того как оптимизатор определил, какие секции следует обрабатывать, все блоки в них помещаются в общий пул для разбиения на диапазоны.

Это предположение оптимизатора ведет к важному замечанию о применении параллелизма к секционированным таблицам. Степень параллелизма (количество PE-процессов для таблицы в целом) относится к набору секций, участвующих в операции. Оптимизатор исключает те секции, в которых заведомо нет данных для обработки данной операцией. Например, если в одной из секций хранятся строки, в которых столбец ID меньше 1000, а в запросе указаны значения ID между 1100 и 5000, то оптимизатор понимает, что эту секцию обрабатывать не нужно.

Начиная с версии Oracle9i, можно также секционировать таблицы по списку конкретных значений, хотя такой способ обычно применяется для удобства обслуживания таблиц. Oracle добавляет все новые способы секционирования.

Если вы ожидаете, что в запросах будет задействован механизм отсечения секций, и планируете пользоваться параллелизмом, то следует расслаивать каждую секцию на достаточное количество дисков в целях обеспечения масштабируемости. Тогда решение будет масштабируемым вне зависимости от количества секций, к которым осуществляется обращение. Расслоение можно реализовать вручную за счет размещения разных файлов данных на разных дисках, или путем применения расслоенных дисковых массивов, или в виде комбинации того идругого.

## Что поддается распараллеливанию?

Oracle умеет распараллеливать отнюдь не только простые запросы. Вот перечень операций, к которым применим параллелизм по диапазонам блоков:

- создание табличных пространств;
- создание и перестроение индексов;
- оперативная реорганизация и перестроение индексов;
- реорганизация и перемещение индекс-таблиц;
- создание таблиц в результате запроса командой CREATE TAB- LE...AS SELECT (например, для агрегирования данных);
- операции обслуживания секций, например перемещение и расщепление;
- загрузкаданных;
- проверка ограничений целостности;
- сбор статистики (автоматический, начиная с версии Oracle Database 10^);
- резервное копирование и восстановление (в том числе очень больших файлов в версии Oracle Database *llg);*
- операции манипулирования данными (INSERT, UPDATE, DELETE);

- операции обработки запросов;
- OLAP-агрегирование (начиная с версии Oracle Database 10^f).

Также Oracle может распараллеливать отдельные шаги выполнения запроса:

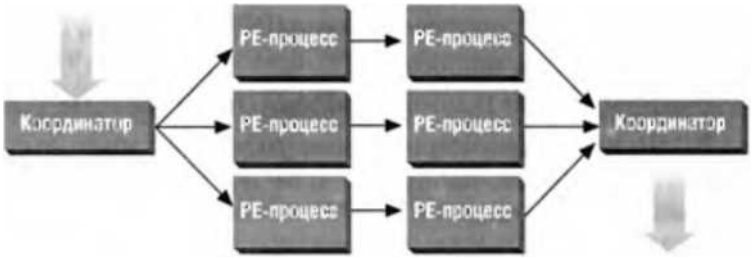
- сканирование таблицы;
- соединение методом вложенных циклов;
- соединение методом сортировки и слияния;
- соединение методом хеширования;
- соединение типа «звезда» с применением битовых индексов;
- сканирование индекса;
- соединение по секциям (partition-wise joins);
- антисоединение (NOT IN);
- SELECT DISTINCT;
- UNION и UNION ALL;
- ORDER BY;
- GROUP BY;
- агрегирование;
- импорт;
- функции, определенные пользователями.

## Степень параллелизма

У экземпляра Oracle имеется пул параллельно выполняемых процессов (PE-процессов), доступных пользователям базы данных. Управление количеством активных PE-процессов было немаловажно в старых версиях Oracle; если PE-процессов было слишком много, то они перегружали машину, а это вело к конкуренции за ресурсы и падению производительности. Кроме того, при высокой степени параллелизма предпочтение отдается полному сканированию таблиц, что не всегда подходит. На рис. 7.6 иллюстрируется прозрачный параллелизм внутри одного набора PE-процессов и между наборами.

Определение оптимальной степени параллелизма при наличии нескольких пользователей и изменяющейся рабочей нагрузке - нетривиальная задача. Например, при степени 8 обработка некоторого запроса могла бы показать великолепную производительность для одного или двух пользователей, но что если к той же таблице обратятся сразу 20 пользователей? Возникнет 160 PE-процессов (по 8 на каждого из 20 пользователей), что приведет к перегрузке машины.






***Результаты***

- Координатор выделяет PE-процессы и разбивает задачу на подзадачи
- Каждый «набор» PE-процессов выполняет отдельную задачу (сортировку, соединение и так далее)
- Результаты передаются «по конвейеру» от одного набора PE-процессов к следующему

*Рис. 7.6. Внутриоперационный и межоперационный параллелизм*

Если взять какое-нибудь значение (например, 2), обеспечивающее эффективный параллелизм для многих одновременно работающих пользователей, то ресурсы не будут использоваться в полной мере, когда количество пользователей невелико.

## Самонастраивающийся адаптивный параллелизм

В Oracle8i введено понятие самонастраивающегося адаптивного параллелизма. Смысл его в том, что степень параллелизма автоматически уменьшается при повышении нагрузки на систему и увеличивается при ее понижении. Если некоторая операция запрашивает определенную степень параллелизма, то Oracle смотрит, какова сейчас нагрузка, и во избежание перегрузки предоставляет меньшую степень. Чем больше пользователей выполняют параллельные операции, тем меньше степень параллелизма для каждого из них, в предельном

случае все операции выполняются последовательно. Если активность снижается, то последующие операции будут выполняться с большей степенью параллелизма. Такая адаптивность освобождает администратора от решения сложной задачи определения оптимальной степени параллелизма в условиях постоянно изменяющейся рабочей нагрузки.

Механизм адаптивного параллелизма принимает во внимание два фактора:

- нагрузка на систему;
- ограничения параллельного доступа к ресурсам для группы, к которой принадлежит пользователь (если работает менеджер ресурсов Database Resource Manager). Это важно, поскольку означает, что учитываются ресурсные планы, если они имеются.

## Параллелизм по секциям

Малая часть функциональности Oracle, связанной с параллелизмом, основана на количестве секций или подсекций, к которым обращается обрабатываемая SQL-команда. В случае параллелизма на основе диапазонов блоков PE-процесс работает над частью данных, описываемой диапазоном блоков. Адля параллелизма на основе секций такой частью является секция или подсекция таблицы. Такой вид параллелизма применим к следующим операциям:

- обновление и удаление;
- сканирование индексов;
- создание и перестроение индексов над секционированными таблицами.

### Параллелизм для секций и подсекций таблицы

В версии Oracle8 введена поддержка распараллеливания языка манипулирования данными (Data Manipulation Language, DML), то есть возможность параллельно выполнять команды INSERT, UPDATE и DELETE. При этом повышается производительность массовых операций (например, обновление всех строк очень большой таблицы).

В Oracle8 степень параллелизма для операций обновления и удаления увязана с количеством участвующих секций, а в Oracle8i и последующих версиях - с количеством участвующих секций или подсекций. Для таблицы с 12 секциями (например, по одной на каждый месяц года) в обновлении или удалении могут участвовать не больше 12 PE- процессов. При обновлении данных только за один месяц никакого распараллеливания не будет, так как участвует только одна секция. Если бы таблица создавалась с помощью механизма составного секционирования (например, 4 хешированных подсекции по столбцу PRODUCT_ID в каждом месяце), то максимальная степень параллелизма для всей таблицы составила бы 48 (12 секций по 4 подсекции в каждой). В обновлении данных за один месяц могло бы участвовать 4 PE-процесса, поскольку секция для одного месяца состоит из 4 подсекций. Для не секционированных таблиц Oracle не может выполнять обновление или удаление параллельно.

В версии Oracle8 и более поздних можно параллельно выполнять создание, перестроение и сканирование секционированных индексов, так же как для команд манипулирования данными - по одному PE-про- цессу для секции или подсекции.

### Быстрое полное сканирование не секционированных таблиц

Многие думают, что Oracle может параллельно сканировать только секционированные индексы. Но в версии Oracle 7.3 появилась возможность в некоторых случаях распараллеливать сканирование и не секционированных индексов. Если операция сканирования индекса «не ограничена», то есть для выполнения запроса необходимо просмотреть весь индекс, то для этого можно применить распараллеливание по диапазонам блоков. Хотя Oracle и умеет сканировать не секционированные индексы, это возможно только для узкого круга запросов. Распараллеливание по секциям индекса применимо к гораздо более широкому спектру запросов.

### Параллельная вставка в секционированные и не секционированные таблицы

Oracle может параллельно выполнять команды INSERT вида INSERT INTO *tableX* SELECT...FROM *tableY* как для секционированных, так и для не секционированных таблиц.

Для обработки предложения SELECT такой команды INSERT используются PE-процессы на базе параллелизма по диапазонам блоков. Эти процессы передают отобранные строки второму набору PE-процессов, которые осуществляют вставку в целевую таблицу. Целевая таблица может быть секционирована или нет. Распараллеливание вставки не основано ни на диапазонах блоков, ни на секциях.

# Oracle и оперативная память

Доступ к информации в оперативной памяти производится гораздо быстрее, чем к информации на диске. Экземпляр Oracle использует память сервера для кэширования прочитанной информации с целью повышения производительности. Для этого служит часть разделяемой памяти, называемая системной глобальной областью (System Global Area, SGA), а также область

в памяти каждого серверного процесса, называемая программной глобальной областью (Program Global Area, PGA).

До версии Oracle9i вы могли задать размер SGA или любой из ее частей - кэша буферов базы данных, разделяемого пула или большого пула - только в файле инициализации, и без перезапуска экземпляра изменить эти размеры было невозможно. В Oracle9i размер пулов стал изменяться динамически, исходя из минимальной единицы выделения памяти, называемой *гранулой*. Начиная с Oracle Database Ю^реализо- вано автоматическое управление разделяемой памятью. А в Oracle Database llg добавилось автоматическое управление SGA и PGA.

Исчерпание памяти сервера базы данных приводит к снижению производительности. Если вы работаете со старыми версиями Oracle, то должны периодически интересоваться размерами различных областей памяти и наращивать объем памяти по мере необходимости, чтобы избежать дефицита. Правильный размер той или иной области зависит от конкретного приложения, необходимых ему данных и ваших требований к производительности.

## Как Oracle использует системную глобальную область (SGA)

Oracle использует SGA для следующих операций:

- кэширование блоков, содержащих данные из таблиц и индексов, в кэше буферов;
- кэширование разобранных и оптимизированных SQL-команд, хранимых процедур и информации из словаря данных в разделяемом пуле;
- хранение записей в журнал в журнальном буфере до момента сброса на диск.

В версиях до Oracle9i объем памяти, выделяемой под каждую из этих частей SGA, определялся на этапе запуска экземпляра на основе параметров инициализации и не мог быть изменен без перезапуска.

Большинство возможностей настройки относились к оптимизации кэша буферов и разделяемого пула.

### Автоматический выбор размера SGA

В версии Oracle Database 10g* ручная настройка пулов SGA уступила место автоматическому выбору размера. Подсистема автоматического управления разделяемой памятью умеет определять подходящие размеры следующих пулов: кэш буферов, разделяемый пул, большой пул, Java-пул и Streams-пул. Нужно лишь задать полный объем памяти, выделяемой под SGA, в параметре инициализации SGA_TARGET.

Начиная с версии Oracle Database 10g* СУБД следит за тем, сколько памяти требуется каждому пулу, динамически изменяя их размеры по мере необходимости. Не отказываясь вовсе от автоматического определения размера SGA, вы также можете задать минимальные размеры пулов. Для этого предназначены параметры ииииализации DB_CA- CHE_SIZE, SHARED_POOL_SIZE, LARGE_POOL_SIZE, JAVA_POOL_ SIZE и STREAMS_POOL_SIZE. Размеры некоторых расположенных в SGA пулов - LOG_BUFFER, DB_KEEP_CACHE_SIZE и DB_RECYC- LE_CACHE_SIZE - по-прежнему можно задавать только вручную.

### Кэш буферов базы данных

Если вы решите отключить динамическое управление SGA, установив параметр SGA_TARGET в 0, тодолжны будете задать параметры инициализации для пулов памяти вручную (если не хотите использовать предыдущие размеры). Для кэша буферов нужно оценить, какая доля блоков, запрошенных пользователями, считывается из кэша, а какая - с диска. Отношение этих величин называется коэффициентом попадания в кэш. Если время отклика слишком велико, а коэффициент попадания меньше 90%, то имеет смысл увеличить значение параметра инициализации DB_CACHE_SIZE.

Можно было бы предположить, что всякое увеличение размера кэша буферов непременно повысит производительность. Однако это верно лишь в случае, когда блоки, находящиеся в кэше, действительно используются повторно. В большинстве OLTP-систем интенсивные обращения производятся лишь к относительно небольшому набору основных таблиц (например, справочных таблиц, в которых хранятся допустимые коды). Остальная часть ввода/вывода более-менее случайна - то из одного блока извлекаются одна-две строки, то из другого. Поэтому увеличение размера кэша может и не привести к повышению производительности.

Кроме того, не все операции читают данные из кэша буферов. Например, при выполнении полного сканирования таблицы допускается занять лишь очень небольшое количество буферов, иначе блоки, считываемые в ходе этой операции, займут весь кэш, что негативно отразится на работе других пользователей. Если ваше приложение часто сканирует таблицы, то увеличение размера кэша буферов ничего не даст, так как нужных блоков в кэше не окажется. При параллельном сканировании таблиц кэш вообще обходится, и отобранные строки передаются непосредственно запрашивающему пользовательскому процессу. Как и при решении любых вопросов, связанных с производительностью, чем лучше вы понимаете, как приложение

работает с данными, тем эффективнее сумеете настроить размер кэша буферов.

### Разделяемый пул

Разделяемый пул используется в нескольких местах при выполнении любой операции с базой данных Oracle. Например, в нем кэшируются предъявляемые клиентами SQL-команды и информация из словаря данных, необходимая для выполнения запросов. В силу его важности для работы базы данных занижение размера разделяемого пула может сильнее сказаться на производительности, чем задание слишком малого размера кэша буферов. Если запрошенный блок отсутствует в кэше буферов, Oracle выполнит операцию ввода/вывода и прочитает его, что приведет лишь к однократному замедлению работы.

Если же слишком мал разделяемый пул, то это отразится на производительности так, что почувствуют все пользователи. И причин тому несколько:

- В кэш помещается слишком мало информации из словаря данных, поэтому приходится часто обращаться к диску для запроса и обновления информации в словаре.
- Не удается кэшировать достаточное количество SQL-команд, что ведет к «перетряхиванию» памяти, когда полезные приложения вытесняются из кэша, чтобы освободить место для новых. Если памяти недостаточно для кэширования всех SQL-команд, предъявляемых приложением, то одни и те же команды приходится разбирать, кэшировать и выталкивать снова и снова, бессмысленно растрачивая ресурсы процессора и увеличивая накладные расходы на выполнение каждой транзакции.
- Не удается кэшировать достаточное количество хранимых процедур, что опять-таки ведет к «перетряхиванию» памяти и снижению производительности при выполнении кода, хранящегося в базе данных.

Если, управляя разделяемым пулом вручную, вы обнаружите какую- либо из этих неприятностей, решение не составит труда: просто увеличьте размер разделяемого пула с помощью параметра инициализации SHARED_POOL_SIZE. Для больших, активно используемых баз данных нет ничего необычного в том, что разделяемый пул занимает 150-250 Мбайт. Дополнительную информацию об исследовании работы разделяемого пула для выявления проблем вы найдете в официальном «Руководстве по настройке производительности Oracle» (Oracle Performance Tuning Guide) или в книгах, перечисленных в приложении 2.

### Журнальный буфер

Журнальный буфер занимает очень небольшую часть SGA по сравнению с кэшем буферов и разделяемым пулом, но с точки зрения производительности его важность трудно переоценить. Всякая транзакция, изменяющая данные в базе, записывает информацию, необходимую для повторного выполнения, в журнальный буфер в памяти. Этот буфер сбрасывается в журнал на диске в момент фиксации транзакции (нормальная ситуация) или когда уже заполнен на треть. Oracle «огораживает» часть журнального буфера, которая в данный момент переписывается на диск, чтобы содержимое этой части больше никто не изменял. В оставшуюся часть журнального буфера транзакции могут вносить новую информацию. В интенсивно используемой базе данных транзакции могут генерировать так много информации, что неогороженная часть журнального буфера заполнится еще до того, как закончится сброс на диск огороженной части. В этом случае транзакции вынуждены ждать завершения ввода/вывода, поскольку в буфере не осталось места. Такая ситуация может негативно сказаться на производительности. Чтобы разобраться в ней, полезен статистический показатель «redo buffer allocation retries» (количество повторных попыток получить место в журнальном буфере). Он доступен через представление V$SYSSTAT и говорит о том, как часто сеансу приходилось ждать освобождения места в журнальном буфере. Вот запрос для получения этой статистики:

```
SELECT name, value FROM VSSYSSTAT
    WHERE name = 'redo buffer allocation retries';
```

Чтобы увидеть тенденцию, нужно некоторое время отслеживать статистические данные. Значение в каждый момент показывает, сколько раз событие имело место со времени запуска экземпляра, и само по себе не очень осмысленно. Отметим, что это замечание относится ко всем статистикам, применяемым для настройки производительности. В идеале, значение показателя «redo buffer allocation retries» должно быть близко к 0. Если вы видите, что за время наблюдения оно выросло, то следует увеличить размер журнального буфера, изменив значение параметра инициализации LOG_BUFFER.

### Кэширование результатов запросов

Одно из самых важных нововведений в Oracle Database 11 *g* призвано повысить производительность выполнения повторяющихся запросов. Oracle кэширует блоки базы данных, устраняя необходимость выполнения дорогостоящих операций чтения с диска. Oracle кэширует планы выполнения SQL, избегая тем самым повторного разбора и оптимизации

запросов. Но до выхода версии Oracle Database 11 *g* кэшированный план все равно приходилось выполнять и получать результирующий набор.

Новый механизм позволяет кэшировать полученный результирующий набор в разделяемом пуле. Это означает, что при повторном выполнении того же запроса результаты можно будет просто извлечь из памяти. Так как для работы этого механизма нужно, чтобы результирующие наборы были в точности одинаковы, наибольшего эффекта от него можно достичь при возврате, например, многократно запрашиваемых веб-страниц. Эта возможность применима и к результатам работы PL/SQL-функций.

В версию Oracle Database *llg* включена также возможность кэшировать результирующие наборы на стороне клиента, при этом автоматически гарантируется согласованность кэшированного набора с любыми изменениями, которые могли бы на него повлиять. Это дает те же преимущества, что и кэширование результирующих наборов на стороне сервера, но без дополнительных запросов по сети.

## Как Oracle использует программную глобальную область (PGA)

У каждого серверного процесса есть своя частная программная глобальная область (PGA) памяти, в которой хранится информация о состоянии этого процесса. Общий объем памяти, отведенной под всю PGA, зависит от количества серверных процессов в одном экземпляре Oracle. Чем больше пользователей, тем больше серверных процессов и тем больше памяти требуется под их PGA. Использование многопоточного сервера (Multi-Threaded Server, или - начиная с версии Orac- le9i " разделяемого сервера) сокращает общее потребление памяти за счет уменьшения количества серверных процессов.

PGA служит рабочей областью для хранения различных временных переменных, используемых серверным процессом, информации об SQL-командах, выполняемых данным процессом, а также для сортировки. Начальным размером рабочей области для переменных (называемой *пространством стека)* напрямую управлять нельзя, так как он зависит от конкретной операционной системы. Для остальных частей PGA можно задавать размер, как описано в следующих разделах.

### Память для SQL-команд

Выполняя SQL-команду в интересах пользователя, серверный процесс сохраняет сеансовую информацию о самой команде и о ходе ее выполнения в части PGA, которая называется *приватной областью SQL* (private SQL area), или *курсором*. Не следует путать ее с разделяемой областью SQL в разделяемом пуле. Разделяемая область SQL содержит общую информацию об SQL-команде, например план ее выполнения, выработанный оптимизатором.

Приватная область SQL содержит специфическую для сеанса информацию о выполнении SQL-команды в данном сеансе, например количество уже извлеченных строк. По завершении обработки команды приватную область SQL можно повторно использовать для другой команды. Если приложение снова предъявляет команду, приватная область которой уже была использована для обработки другой команды, то эту область придется инициализировать заново.

При получении каждой новой SQL-команды нужно найти (или загрузить) соответствующую ей разделяемую область SQL в разделяемом пуле. Аналогично, приватная область SQL для новой команды ищется в PGA, а если не найдена, инициализируется серверным процессом. Такая повторная инициализация обходится довольно дорого с точки зрения процессорного времени.

Серверный процесс, в PGA которого находится много разных приватных областрй SQL, будет тратить меньше времени на повторную инициализацию этих областей для вновь поступающих команд. Если серверному процессу не приходится повторно использовать существующую приватную область SQL для новой команды, то уже инициализированную область можно оставить без изменения. Задание большого размера PGA позволяет избежать «перетряхивания» памяти из-за повторной инициализации приватных областей SQL. Чем больше степень повторного использования приватных областей SQL, тем меньше потребление процессорного времени и, значит, тем выше производительность. Разумеется, имеет место компромисс между выделением памяти в PGA для приватных областей SQL и общей производительностью.

В OLTP-системах обычно есть «рабочее множество» SQL-команд, предъявляемых каждым пользователем. Например, пользователь, отвечающий за бронь, снова и снова заполняет одну и ту же форму. Производительность можно повысить, если в серверном процессе, обслуживающем этого пользователя, достаточно памяти для SQL-команд, отправляемых в ходе обработки этой формы. И разработчики приложений должны писать SQL-команды с учетом повторного использования, с применением переменных связывания, а не «зашивая» значения параметров прямо в SQL-запрос.

### Память для сортировки внутри PGA

Для сортировки строк перед их отправкой пользователю серверный процесс использует память в своей области PGA. Если выделенной памяти недостаточно для размещения всех сортируемых строк, то сортировка производится в несколько *проходов*. Результаты промежуточных проходов сохраняются во временном табличном пространстве пользователя, что приводит к снижению производительности из-за необходимости обращений к диску.

До версии Oracle Database *10g* задание размера области сортировки в PGA было одним из важных аспектов настройки. Если эта область оказывалась слишком маленькой для типичных сортировок, то приходилось прибегать к временному табличному пространству, что снижало производительность. Если же область сортировки была слишком велика, то зря расходовалась память.

Начиная с версии Oracle Database 10g* СУБД может автоматически управлять размером PGA. По умолчанию автоматическое управление включено, и размер PGA устанавливается как 20% от размера SGA. Соглашаясь на автоматическое управление, вы избавляете себя от необходимости задавать параметры настройки отдельных частей PGA, например SORT_AREA_SIZE.

В версии Oracle Database 1*lg* подсистема автоматического управления памятью охватывает и SGA, и PGA. Если задать параметр инициализации MEMORY_TARGET, то (принимая во внимание, что размер PGA может быть выражен в процентах от размера SGA) размерам PGA и SGA будут автоматически присвоены начальные значения. Далее Oracle обеспечивает оптимальную производительность SGA и PGA исходя из текущей нагрузки.

## TimesTen

В 2005 году корпорация Oracle приобрела продукт TimesTen - лидирующую СУБД с размещением целиком в памяти. СУБД с размещением в памяти обеспечивают максимальную производительность, поскольку не тратят времени на обращение к диску. Встроенный в TimesTen оптимизатор знает о том, что данные находятся в памяти, учитывая это при создании плана выполнения. Кроме того, значительно сокращается объем кода, поскольку не нужно обрабатывать ситуацию, когда данные находятся на диске. TimesTen лучше всего подходит для высоконагруженных OLTP-систем, которые должны реагировать в масштабе реального времени.

Экземпляр TimesTen можно использовать как кэш для базы данных Oracle. Вы загружаете в экземпляр TimesTen подмножество таблиц Oracle, а механизм Cache Connect to Oracle обеспечивает синхронизацию данных.

Можно также включить репликацию между экземплярами TimesTen на разных машинах с целью балансирования нагрузки и обеспечения более высокой доступности. Дополнительную информацию о продукте TimesTen можно найти на сайте Oracle Technology Network.

## Oracle и ресурсы процессора

СУБД Oracle делит процессоры со всеми другими программами, работающими на сервере. Если мощности процессора не хватает, то сокращение потребления его времени со стороны Oracle или другого ПО может повысить производительность всех процессов, работающих на данном сервере.

Если все процессоры заняты, то процессы становятся в очередь, дожидаясь, когда им будет предоставлен процессор. Это называется *очередью на выполнение*, поскольку процессы ждут возможности начать выполнение. Чем сильнее загружены процессоры, тем дольше процессы вынуждены простаивать в очереди. Стоящий в очереди процесс ничего не делает, поэтому по мере роста очереди увеличивается и время отклика.

Оптимизация использования процессора сводится к настройке отдельных заданий: требуется либо уменьшить количество машинных команд, необходимых для выполнения задания, либо уменьшить количество выполняемых заданий. Добиться этого можно путем балансирования рабочей нагрузки, оптимизации SQL или пересмотра структуры приложения. Эта деятельность требует глубокого понимания природы заданий и того, как они выполняются.

Выше уже отмечалось, что подробное рассмотрение всех тонкостей настройки базы данных Oracle выходит за рамки настоящей книги. Однако мы расскажем о том, какие действия обычно приводят к повышенному потреблению времени процессора. Вот на что в первую очередь следует обратить внимание, когда серверу базы данных не хватает мощности процессора.

*Плохо составленные SQL-запросы*

Плохие SQL-запросы - главная причина неурядиц с производительностью. СУБД Oracle пытается оптимально выполнять запросы, поступающие от клиентов. Но если SQL-команды, которые приложение посылает серверу, написаны так, что даже самый лучший план

выполнения оказывается неэффективным, то Oracle будет потреблять больше ресурсов, чем необходимо. Настройка SQL может вылиться в сложный и длительный процесс, так как для этого требуется глубокое понимание принципов работы Oracle и задач приложения. Даже начальное расследование может обнаружить дефекты в проекте базы данных и, как следствие, необходимость изменить структуру таблиц, добавить индексы и так далее. После изменения SQL-команд требуется повторное тестирование и развертывание приложения. Но это если вы работаете с версией до Oracle Database 10g.

В Oracle Database 10g появился инструмент SQL Tuning Advisor, способный не только выявить плохо написанные SQL-запросы, но и создать план выполнения, позволяющий обойти проблему, и подменить им стандартный план, выработанный оптимизатором. Эта возможность позволит вам улучшить производительность плохо написанных запросов, не изменяя код приложения. Консультант SQL Advisor в версии Oracle Database 11 *g* объединяет функции SQL Tuning Advisor, SQL Access Advisor и Partition Advisor.

В версии Oracle Database 1 lg СУБД может автоматически выявлять запросы, создающие максимальную нагрузку, и создавать SQL-профили для повышения их производительности. Дополнительно СУБД может подсказать, какие индексы стоит построить для ускорения обработки этих запросов.

Oracle Database 1 lg также следит за изменениями в планах выполнения запросов. Оптимизатор может хранить историю планов выполнения и, обнаружив новый план, сравнивать его со старым. Убедившись, что новый план обеспечивает такую же или лучшую производительность, оптимизатор заменит им старый. Этот механизм не связан напрямую с плохо написанными запросами, но позволяет обнаружить непреднамеренные эффекты от изменения плана, в результате которых производительность могла бы снизиться.

*Чрезмерное количество синтаксических разборов*

В разделе «Память для SQL-команд» мы говорили, что перед тем как выполнить любую SQL-команду, Oracle нужно произвести ее синтаксический разбор. Разбор требует очень много процессорного времени и включает многочисленные обращения к словарю данных для проверки существования всех упоминаемых таблиц и столбцов. Для оценки стоимости различных планов выполнения и выбора оптимального применяются сложные алгоритмы и вычисления. Если приложение не пользуется переменными связывания, то серверу придется разбирать каждую предъявляемую команду. Такие лишние разборы - одна из основных причин снижения производительности. Еще одна типичная причина - недостаточный размер разделяемого пула, о чем уже говорилось выше. Имейте в виду, что можно избежать создания планов выполнения, если воспользоваться хранимыми схемами планов. А начиная с версии Oracle 9*i* появилась возможность редактировать подсказки, из которых состоит хранимая схема плана. Выше уже было сказано, что в версии Oracle Database *llg* есть возможность кэшировать результирующие наборы целиком, что также уменьшает потери от повторного выполнения одних и тех же запросов.

*Рабочая нагрузка на базу данных*

Если приложение хорошо спроектировано, и база данных работает в оптимальном режиме, то нехватка ресурсов процессора может быть обусловлена тем простым фактом, что мощности процессоров не хватает для выполнения всех возложенных на сервер задач. Нехватка может быть вызвана нагрузкой на одну базу данных (если машина является выделенным сервером) или комбинированной нагрузкой на все базы, расположенные на данном сервере. Недооценка требуемых ресурсов процессора - хроническая проблема, возникающая еще на этапе планирования аппаратных средств. К сожалению, для точной оценки ресурсов при определенном уровне активности необходимо очень хорошо понимать, сколько процессорного времени потребляет каждая транзакция и сколько транзакций в минуту или в секунду будет обрабатывать система - в периоды средней и пиковой нагрузки. Во многих организациях нет ни времени, ни возможностей для системного анализа и построения прототипа, а иначе получить ответы на эти вопросы невозможно. Наверное, именно по этой причине нехватка мощности ЦП - столь распространенная проблема, а чтобы ее решить, чаще всего просто добавляют процессоры, пока все не придет в норму. Кластерные системы на базе технологии Real Application Clusters и grid-системы представляют собой попытки хотя бы облегчить добавление новых обрабатывающих мощностей.

*Посторонняя нагрузка*

Не все организации могут позволить себе выделить целую машину только под базу данных Oracle, чтобы сервер мог получить в свое распоряжение все ресурсы процессоров. Применяйте утилиты операционной системы, чтобы выявить «пожирателей ЦП». Может обнаружиться, ч^о большую часть времени потребляют процессы, не имеющие отношения к Oracle, снижая тем самым производительность базы данных.

## Database Resource Manager

В предыдущем разделе были описаны некоторые причины снижения производительности из-за нехватки ресурсов процессора. В версии Oracle8i впервые появился менеджер ресурсов базы данных (Database Resource Manager, DRM), автоматически устраняющий некоторые из перечисленных проблем.

Работа DRM основана на идее определяемых вами групп потребителей. Для каждой группы можно установить предельные значения потребляемых ею машинных ресурсов. DRM гарантирует, что никакая группа или член группы не будет потреблять львиную долю ресурса. Кроме того, DRM стремится предоставить гарантированное обслуживание различным группам пользователей. Можно создавать иерархии DRM, в которых задаются уровни потребления ресурсов для групп, вложенных в другие группы.

Чтобы предотвратить снижение производительности, можно комбинировать следующие возможности DRM:

*Прогноз потребления ресурсов*
: DRM может пользоваться результатами расчетов стоимости, произведенных оптимизатором для предсказания объема необходимых для выполнения данного запроса ресурсов и времени его выполнения. Отметим, что начиная с версии Oracle Database 10g оптимизатор по умолчанию учитывает в целевой функции стоимость процессора и ввода/вывода. В версии Oracle9i применялась модель, основанная только на стоимости чтения одиночных блоков.

*Переключение групп потребителей*
: DRM умеет динамически переключать группы потребителей. Возможно, вы хотите выделить конкретной группе значительную долю ресурсов процессора. Но если оказывается, что какой-то запрос от членов этой группы потребит слишком много ресурсов ЦП, и это негативно отразится на производительности машины в целом, то DRM может переключиться на другую группу с более низким уровнем разрешенного потребления времени ЦП - например, на группу для пакетных операций.

*Ограничение количества соединений*
: DRM может ограничивать количество соединений для конкретной группы потребителей. Если лимит на количество соединений достигнут, то следующий запрос о соединении ставится в очередь и удовлетворяется лишь после закрытия какого-либо из уже установленных соединений. Ограничивая общее количество соединений для группы потребителей, вы вводите некие грубые лимиты на выделяемые этой группе ресурсы.

В версии Oracle Database 11 *g* при установке базы данных задается план DRM по умолчанию. Он призван ограничить объем ресурсов, выделяемых автоматическим заданиям обслуживания, например сборщику статистики для оптимизатора, а также консультантам Automatic Segment Advisor и Automatic SQL Tuning Advisor.

# Лекция 8. Конкурентный многопользовательский доступ в Oracle

## Основы конкурентного доступа

Чтобы осознать проблемы, возникающие при многопользовательском конкурентном доступе к данным, необходимо определить несколько базовых понятий.

## Транзакции

*Транзакция* - основа целостности данных в многопользовательских СУБД и фундамент всех моделей конкурентного доступа. Транзакцией называется единая и неделимая последовательность операций над данными. Все изменения данных, выполненные в рамках транзакции, единовременно применяются, к базе данных командой COMMIT либо все измененные данные единовременно возвращаются в исходное состояние командой ROLLBACK. После того как транзакция зафиксирована командой COMMIT, произведенные изменения становятся постоянными и видимыми другим транзакциям и пользователям.
Выполнение транзакции всегда занимает некоторое время, обычно очень небольшое. 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*. Блокировка не дает пользователям модифицировать данные. В СУБД блокировки применяются

для того, чтобы помешать одной транзакции затереть изменения, произведенные другой транзакцией.

Возможная проблема показана ниже. Транзакция *А* читает элемент данных. Транзакция *В* читает тот же самый элемент и фиксирует изменение данных. Когда транзакция *А* доходит до этапа фиксации, сделанные ею изменения непредумышленно затирают изменения, выполненные транзакцией Б, в результате чего утрачивается целостность данных.

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

В некоторых СУБД также применяются *блокировки чтения* (read locks), или *разделяемые блокировки* (shared locks). Блокировка чтения может удерживаться любым количеством пользователей, которые просто читают данные, так как доступ только для чтения не создает конфликтных ситуаций. Однако блокировка чтения означает, что к данным нельзя применить блокировку записи, поскольку последняя является монопольной. Если на рис. 8.1 установить блокировку чтения вмомент начала транзакцииА, то транзакцииБ будет разрешено читать те же данные, но ей не удастся заблокировать данные для записи до тех пор, пока транзакцияАне завершится.

В Oracle блокировки чтения применяются, только если пользователь явно запрашивает их в предложении FOR UPDATE команды SELECT. Не следует употреблять предложение FOR UPDATE во всех запросах, так как при этом возрастает вероятность того, что «читатели» будут мешать «писателям». Как вы вскоре убедитесь, такая ситуация в Oracle обычно не возникает.

## Конфликты блокировок при конкурентном доступе

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 производительность, так как установленные ею блокировки не дают завершиться другим транзакциям. Возникает так называемый *конфликт блокировок* (contention), который может отрицательно сказаться на производительности СУБД. Чем больше в базе данных конфликтов, тем ниже скорость отклика и общая пропускная способность.

В большинстве других СУБД повышение степени конкурентности доступа к данным приводит к увеличению числа конфликтов и снижению производительности. Реализованная в Oracle многоверсионная модель согласованности по чтению (Multiversion Read Consistency) существенно уменьшает количество конфликтов, как станет ясно из дальнейшего изложения.

## Проблемы целостности

Если механизмы изоляции транзакций применяются неправильно, то возможно нарушение целостности данных. Для многих СУБД характерны четыре ситуации подобного рода.

*Потерянные обновления (lost updates)*

Наиболее распространенная проблема целостности возникает, когда два писателя одновременно изменяют одни и те же данные. При этом изменения, произведенные одним писателем, затирают изменения, выполненные другим. Именно эту проблему призваны предотвратить монопольные блокировки.

*Грязное чтение (dirty reads )*

Считывание данных называется «грязным», если СУБД разрешает одной транзакции читать данные, измененные другой транзакцией, когда эти изменения еще не 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 условию. После того как данные получены, но до завершения первой транзакции, 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 другой результат, поэтому любые изменения, выполненные на основе первоначального результата, могут уже не быть корректными. При повторном чтении уже прочитанных данных в рамках одной и той же транзакции могут быть получены отличающиеся результаты.

*Фантомное чтение (phantom reads)*

Такой вид чтения также вызван изменениями, выполненными в другой транзакции. Одна транзакция выполняет запрос по какому-то условию. После того как данные получены, но до завершения первой транзакции, другая транзакция вставляет в базу данных новые строки, удовлетворяющие условию запроса в первой транзакции. Если в первой транзакции были затем произведены изменения над множеством строк, удовлетворяющих тому же критерию, что и в первоначальном запросе, то количество измененных строк будет отличаться от количества ранее прочитанных из-за новых фантомных строк.

## Сериализация

Полностью решить задачу конкурентного доступа - значит обеспечить наивысший уровень изоляции для действий различных пользователей, обращающихся к одним и тем же данным. Согласно стандарту SQL92, высший уровень изоляции называется *сериализуемым* (serializable). Сериализуемые транзакции для пользователя выглядят так, будто выполняются строго по порядку. Когда одна транзакция начинается, она изолируется от всех изменений, вносимых в обрабатываемые ею данные следующими транзакциями.

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 главные составляющие 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 Oracle с успехом проделывает этот трюк.

# Oracle и конкурентный доступ

В Oracle проблема конкурентного доступа решается с помощью *много- версионной модели согласованности по чтению* (multiversion read consistency, MVRC). Эта технология гарантирует, что пользователь всегда видит непротиворечивое представление запрошенных данных. Если другой пользователь изменит данные в процессе выполнения запроса, Oracle сохранит версию данных в том виде, в каком они находились на момент начала выполнения запроса. Если при этом какие-то транзакции уже выполняются, но еще не зафиксированы, сервер Oracle гарантирует, что запрос не увидит изменения, внесенные такими транзакциями. Возвращенные по запросу данные будут отражать результаты выполнения всех транзакций, зафиксированных на момент начала выполнения запроса.

Этот механизм существенно влияет на способ обработки запросов к базе данных. Прежде всего, Oracle не устанавливает никакие блокировки при чтении. То есть операция чтения никогда не блокирует операцию записи. Даже поставленная СУБД единственная блокировка всего одной строки во время чтения может привести к конфликтным ситуациям, особенно если учесть, что в таблицах базы данных операции обновления обычно выполняются внемногих «местах скопления» активных данных.

Далее, пользователь получает полный моментальный снимок данных, актуальный на момент начала выполнения запроса. Другие СУБД уменьшают количество конфликтов, блокируя отдельную строку только на время ее считывания, а не на все время транзакции, включающей доступ к этой строке. Но тогда строка, извлеченная в конце выполнения запроса, может измениться в промежутке с момента начала выполнения запроса до момента извлечения. Поскольку строки, которые будут прочитаны позже в ходе выполнения запроса, не блокируются и могут быть изменены другими пользователями, полученная картина данных может оказаться противоречивой.

# Уровни изоляции в Oracle

В Oracle, как и во многих других СУБД, для описания взаимодействий одной транзакции с другими применяется концепция *уровней изоляции*. По существу, уровень изоляции - это схема блокировки, реализованная базой данных и гарантирующая определенный тип изолированности транзакций.

Прикладной программист может задать уровень изоляции для всего сеанса (ALTER SESSION) или для отдельной транзакции (SET TRANSACTION). Чем строже уровень изоляции, тем больше вероятность возникновения конфликтов, но и тем выше степень защиты целостности данных.

Чаще всего в Oracle применяются два основных уровня изоляции: RE- AD COMMITTED и SERLUJZABLE. (Третий уровень, READ ONLY, описан в этом разделе ниже.) Оба уровня изоляции обеспечивают се- риализуемость операций с базой данных. Отличаются они временем гарантируемой сериализуемости.

*READ COMMITTED*

Обеспечивает 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 пока она выполняется, не исключено невоспроизводимое или фантомное чтение. Уровень изоляции READ COMMITTED устанавливается в Oracle по умолчанию.

*SERIALIZABLE*

Обеспечивает 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 в протяженности действия эти два уровня изоляции ведут себя по-разному, если какая-то другая транзакция поставила монопольную блокировку на запрошенную строку. Как только блокировка будет снята, транзакция, выполняемая с уровнем изоляции READ COMMITTED, просто повторяет попытку выполнить операцию. 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 транзакция, выполняющаяся с уровнем изоляции SE- RLALIZABLE, 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 невозможными. В подобной ситуации прикладная программа должна вернуться к началу транзакции SERLALIZABLE и выполнить ее заново.

Oracle поддерживает еще один уровень изоляции, READ ONLY, который тоже можно задать для сеанса в целом или для отдельной транзакции. Как видно из названия, такой уровень явно запрещает любые операции записи, гарантируя точное представление данных на момент начала транзакции.

## Механизмы обеспечения конкурентного доступа в Oracle

Для реализации многоверсионной согласованности по чтению в СУБД Oracle применяются три механизма:

*Сегменты отката*

Сегменты отката (rollback segments) служат для хранения информации, необходимой для возврата данных в исходное состояние в случае отката транзакции. Такая информация нужна для восстановления строк базы данных в состояние, в котором они находились в момент начала выполнения рассматриваемой транзакции. Перед тем как изменить данные в блоке, транзакция записывает старый образ данных в сегмент отката. Хранящаяся в сегменте отката информация нужна для предоставления необходимых для отката транзакции сведений и для поддержки многоверсионной согласованности.

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

Блоки сегментов отката кэшируются в SGA наряду с блоками таблиц и индексов. Те блоки сегментов отката, которые долгое время не используются, могут быть вытеснены из кэша и записаны на диск.

*Системные номера изменений (SCN)*

Для того чтобы обеспечить целостность данных в базе и гарантировать сериализацию, необходимо отслеживать порядок выполнения действий. Для этой цели в СУБД Oracle применяются системные номера изменений (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 корректном первоначальном порядке. На основе меток SCN Oracle также определяет, когда можно удалить уже ненужную информацию из сегментов отката.

*Блокировки в блоках данных*

СУБД должна как-то узнать о том, что некоторая строка заблокирована. Большинство СУБД хранит в памяти список блокировок, управляемый отдельным процессом - менеджером блокировок. В Oracle блокировки представлены индикатором в том же 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 весь содержащий ее блок. В блоке данных хранятся индикаторы блокировок, при этом каждая блокировка распространяется только на одну строку в блоке.

Кроме уже упомянутых механизмов, обеспечивающих многоверсион- ную согласованность

чтения, у Oracle есть еще одна особенность, позволяющая достичь большей степени конкурентности вусловиях большого количества пользователей:

*Нерасширяемые блокировки строк (nonescalating row locks)*

Для того чтобы снизить накладные расходы на управление блокировками, некоторые СУБД иногда распространяют блокировки на большее количество данных. Например, если какая-то часть строк таблицы заблокирована, СУБД расширяет блокировку до целой таблицы, блокируя все строки таблицы, в том числе и те, которые не затрагиваются выполняемой SQL-командой. Этот механизм уменьшает количество блокировок, которыми управляет менеджер блокировок, но приводит к блокировке не затрагиваемых операцией строк. Поскольку сервер Oracle хранит блокировку каждой строки в ее блоке данных, нагрузка на менеджер блокировок не увеличивается с возрастанием их количества. Поэтому Oracle никогда не прибегает к расширению.

Менеджер блокировок под названием Distributed Lock Manager (DLM) раньше применялся в продукте Oracle Parallel Server для отслеживания блокировок, общих для нескольких экземпляров. Но это совершенно отдельная схема блокирования, не имеющая никакого отношения к блокировке строк. Заимствованная из Oracle Parallel Server технология DLM была усовершенствована и теперь интегрирована в продукт
Real Application Clusters, являющийся неотъемлемой частью Oracle9i.

# Как Oracle реализует блокирование

Вы уже достаточно знаете о концепциях конкурентного доступа и о механизмах их реализации в СУБД Oracle. Однако чтобы прояснить, как взаимодействуют все эти механизмы, мы рассмотрим три сценария: простая запись в базу данных, попытка одновременной записи в одну и ту же строку таблицы со стороны двух пользователей и ситуация, когда между двумя конфликтующими операциями обновления вклинивается операция чтения.
Мы будем предполагать, что один или два пользователя модифицируют таблицу EMP с данными о работниках, включенную в стандартную демонстрационную схему, поставляемую вместе с СУБД Oracle.

## Простая операция записи

В этом примере рассматривается простая операция записи, когда один пользователь обновляет одну строку таблицы. Предположим, сотрудник отдела кадров хочет изменить фамилию работника. Пусть данные о работнике уже находятся на экране. Последовательность шагов начиная с этого момента:

1. Клиент модифицирует фамилию работника на экране и посылает SQL-команду UPDATE по сети серверному процессу.
2. Серверный процесс получает системный номер изменения и считывает блок данных, содержащий обновляемую строку.
3. Сервер записывает информацию о блокировке строки в блок данных.
4. Сервер сохраняет старый образ данных в сегменте отката, затем записывает изменения в журнальный буфер в памяти и модифицирует данные о работнике, в том числе заносит SCN в псевдостолбец ORA_ROWSCN (в версии Oracle Database 10g и выше).
5. Серверный процесс записывает данные из журнального буфера на диск, а затем сбрасывает на диск сегменты отката и измененные данные. Изменения, хранящиеся в сегментах отката, составляют только часть журнала, поскольку в журнале хранятся вообще все изменения, произведенные в ходе выполнения транзакции.
6. Сотрудник отдела кадров фиксирует транзакцию.
7. Процесс Log Writer (LGWR) переписывает информацию, необходимую для повторного выполнения всей транзакции, в том числе SCN-метку момента фиксации транзакции, из журнального буфера в текущий оперативный журнал на диске. Когда операционная система подтвердит, что запись в журнал успешно завершена, транзакция считается зафиксированной.
8. Серверный процесс посылает клиенту сообщение, подтверждающее факт фиксации.

В версии Oracle Database 10g* Release 2 серверный процесс может возвращать управление клиенту, не дожидаясь завершения записи в журнал. Положительный аспект этой возможности - увеличение производительности высоконагруженных OLTP-приложений. Отрицательный аспект - повышение уязвимости, поскольку сбой в базе данных может произойти после того, как транзакция зафиксирована, но до завершения записи в журнал. В этом случае восстановить зафиксированную транзакцию невозможно, поэтому работать в таком режиме следует с осторожностью.

## Конфликтующие операции записи

Описанная выше операция записи будет выглядеть несколько иначе, если КлиентАи Клиент *B*

попытаются одновременно модифицировать одну и ту же строку. В этом случае шаги будут такими:

1. КлиентА модифицирует фамилию работника на экране и посылает SQL-команду UPDATE по сети серверному процессу.
2. Серверный процесс получает системный номер изменения и считывает блок данных, содержащий обновляемую строку.
3. Сервер записывает информацию о блокировке строки в блок данных.
4. Серверный процесс записывает изменения в журнальный буфер.
5. Серверный процесс копирует старый образ данных о работнике в сегмент отката, а затем модифицирует данные о работнике, в частности заносит SCN в псевдостолбец ORA_ROWSCN (в версии Oracle Database *10gn* выше).
6. Клиент *B* модифицирует фамилию работника на экране и посылает SQL-команду UPDATE по сети серверу.
7. Серверный процесс получает системный номер изменения и считывает блок данных, содержащий обновляемую строку.
8. Серверный процесс видит в заголовке блока данных блокировку для обновляемой строки. Далее он может пойти одним из двух путей. Если уровень изоляции транзакции, начатой Клиентом Б, - READ COMMITTED, то сервер будет ждать завершения транзакции. Если же уровень изоляции этой транзакции - SERLALIZ- ABLE, то клиенту возвращается сообщение об ошибке.
9. КлиентА из отдела кадров фиксирует транзакцию, серверный процесс выполняет соответствующие действия и посылает КлиентуА сообщение, подтверждающее факт фиксации.
10. Если КлиентБ выполнял SQL-команду с уровнем изоляции READ COMMITTED, то он продолжает нормально работать.

Этот пример иллюстрирует стандартное поведение сервера Oracle в ситуации, чреватой потерей обновления. Поскольку уровень изоляции SERLALIZABLE приводит к более радикальным последствиям в случае конфликта записи, многие разработчики предпочитают уровень RE- AD COMMITTED. Избежать потенциальных конфликтов можно одним из двух способов: непосредственно перед обновлением проверить, не изменились ли данные (сравнив значения в строке или SCN строки в версиях Oracle Database 10g* и выше), или воспользоваться конструкцией SELECT FOR UPDATE, чтобы устранить проблему на корню.

## Операция чтения

Всю красоту модели согласованности по чтению в Oracle можно оценить, рассмотрев наиболее типичный сценарий, когда один пользователь читает данные, а другой одновременно пытается их обновить. Предположим, КлиентА читает последовательность строк из таблицы EMP, а Клиент *B* модифицирует некую строку перед тем, как КлиентА ее прочел, но после того, как он начал транзакцию.

1. Клиент А посылает SQL-запрос SELECT по сети серверному процессу.
2. Серверный процесс получает SCN для запроса и начинает считывать запрошенные данных. Для каждого прочитанного блока сервер сравнивает SCN, полученный для запроса, с SCN всех транзакций, которые затрагивают находящиеся в этом блоке и удовлетворяющие запросу строки. Если обнаруживается транзакция с более поздним SCN, чем у запроса, то сервер берет данные из сегментов отката, чтобы получить «согласованную почтению» версию блока данных, актуальную на момент отправки запроса. Именно в этом суть многоверсионной модели согласованности по чтению (MVRC), позволяющей не ставить на строки блокировки чтения. Если строка обновится после начала транзакции, то Oracle просто возьмет ее более раннюю версию.
3. Клиент В посылает SQL-команду UPDATE для обновления строки в таблице EMP, которая еще не была считана в ходе обработки запроса SELECT, посланного КлиентомА. Серверный процесс получает SCN для этой команды и начинает выполнять операцию.
4. Клиент *B* фиксирует свои изменения. Серверный процесс завершает операцию, в частности, записывает данные в блок, содержащий модифицированную строку, что позволяет Oracle определить SCN транзакции обновления.
5. Серверный процесс, выполняющий чтение от имени КлиентаА, доходит до только что модифицированного блока. Он видит, что в этом блоке имеются изменения, произведенные транзакцией с более поздним SCN, чем у запроса SELECT. В заголовке блока серверный процесс находит указатель на сегмент отката, содержащий данные в том виде, в каком они существовали на момент начала транзакции КлиентаА. В качестве результата запроса SELECT возвращаются строки из сегмента отката, то есть КлиентА получает согласованную версию данных.

Мы показали работу модели MVRC на примере двух пользователей. Но представьте себе базу

данных, поддерживающую одно или несколько корпоративных приложений, с которыми одновременно работают сотни пользователей. Механизмы обеспечения конкурентного доступа в Oracle позволяют избежать огромного количества конфликтов и, следовательно, снижения производительности в условиях интенсивной нагрузки. На самом деле, чем выше нагрузка, тем нагляднее проявляются достоинства модели MVRC.

## Конкурентный доступ и производительность

Прочитав описание всех шагов в предыдущих примерах, вы, возможно, решили, что Oracle - очень медленная СУБД. Но это не так. Все промышленные тесты доказывают, что Oracle - одна из самых быстрых (если не самая быстрая) СУБД, представленных сегодня на рынке.

Oracle обеспечивает высокую производительность при реализации мно- говерсионной модели согласованности по чтению, минимизируя и откладывая некоторые операции ввода/вывода. Чтобы гарантировать целостность данных, СУБД должна уметь восстанавливать базу данных в случае системного сбоя. Это означает, что должен быть способ привести базу в состояние, где правильно отражены все изменения, зафиксированные на момент сбоя. Oracle гарантирует это, записывая измененные данные в базу в момент фиксации транзакции. Но поскольку в журнале гораздо меньше информации, чем в целом блоке, включающем измененные данные, его запись на диск обходится гораздо дешевле. Oracle сбрасывает в журнал информацию, как только транзакция зафиксирована, но откладывает запись измененных блоков в базу до тех пор, пока не накопится несколько блоков, чтобы записать их разом. С помощью журналов Oracle способна восстановить базу, поэтому описанная стратегия уменьшает количество длительных операций ввода/вывода.

Однако говоря о производительности СУБД, имеют в виду не одни лишь операции ввода/вывода. Не важно, насколько быстро СУБД с ними справляется, если ваша транзакция вынуждена ожидать снятия блокировки, установленной в другой транзакции. Быстрая СУБД может завершить блокирующую транзакцию быстрее, но ваша-то транзакция все равно «стала, как вкопанная», пока это не произойдет.

Поскольку большинство СУБД выполняют и чтение, и запись, а Oracle - едва ли не единственная СУБД на рынке, в которой не применяются блокировки чтения, в ней практически всегда меньше конфликтов. А раз конфликтов меньше, то общая пропускная способность при смешанной нагрузке выше.

Кроме того, есть разные виды производительности. Производительность операций СУБД измеряется в миллисекундах, а производительность разработчиков приложений - в месяцах. Поскольку при работе с Oracle возникает меньше конфликтов, программистам не нужно тратить время на изобретение обходных путей для их обработки.

Мы не утверждаем, что Oracle - единственная СУБД, обеспечивающая решение проблем конкурентного доступа, которое позволяет гарантировать целостность данных. Но модель многоверсионной согласованности по чтению позволяет получить непротиворечивое представление данных без лишних конфликтов и не прибегая к различным хитростям на прикладном уровне. Если вы подумали, что мы фанаты схемы блокирования Oracle, - что ж, так оно и есть.

## Рабочие области

В версии Oracle9i появился новый механизм, имеющий отношение к конкурентному доступу, - менеджер рабочих областей Workspace Manager.
*Рабочая область* (workspace) - это способ изолировать данные от изменений в общем окружении базы данных. Workspace Manager решает эту задачу, создавая версии данных, специфичные для одной рабочей области. Создавая рабочую область, вы, по существу, создаете моментальный снимок данных в этой области на конкретный момент времени. Последующие изменения данных извне этой рабочей области не скажутся на том, как они видны в самой рабочей области. И наоборот, любые изменения, произведенные внутри рабочей области, не видны внешним по отношению к ней пользователям. Изменения данных в рабочей области видимы только другим пользователям в этой области.
Рабочие области позволяют создавать раздельные окружения для специальных целей. Можно взять данные на какой-то момент времени для анализа истории или выполнять анализ типа «что если», чтобы проверить, как отразятся на общей картине те или иные изменения, не затрагивая при этом основную промышленную базу данных. Обычно для таких операций приходится создавать копию базы данных, но рабочие области позволяют сэкономить время и ресурсы.

## Реализация рабочих областей

В основе рабочих областей - поддержка нескольких версий одних и тех же данных. Чтобы воспользоваться рабочей областью для хранения версии таблицы, нужно сначала разрешить для этой таблицы много- версионность. Это позволяет сделать Workspace Manager. Единицей многоверсионности является строка. Если для таблицы включена мно- говерсионность, то для каждой ее строки может поддерживаться несколько версий. Версии строк хранятся в той же таблице, что и исходные строки. Инфраструктура многоверсионности невидима пользователям базы данных; приложения запрашивают, обновляют, вставляют и удаляют данные точно так же, как и раньше. Для включения многоверсионности Workspace Manager переименовывает таблицу и добавляет в нее несколько столбцов для хранения метаданных, затем создает представление многоверсионной таблицы с тем же именем, что у исходной, и определяет над

этим представлением триггеры INSTEAD OF для выполнения DML-операций.

Чтобы сэкономить место и избежать дублирования, в рабочей области хранятся только изменения данных.

Разрешается создавать иерархии рабочих областей, причем у одного рабочей области может быть несколько родителей. Все описанные ниже операции с рабочими областями затрагивают как саму эту область, так и ее родителей. Многоуровневые рабочие области позволяют точнее управлять изоляцией изменений в многоверсионных таблицах.

Для реализации рабочих областей Oracle включает в строки таблицы метаданные. В состав метаданных может входить, в частности, временная метка, показывающая, когда было произведено изменение, что помогает анализировать активность в рабочей области. Метки применимы и к точкам сохранения, позволяя получить историю всех изменений для каждой версии строки, созданной в точке сохранения. Наличие временной метки дает пользователю возможность вернуться назад во времени и взглянуть на базу данных с точки зрения тех изменений, которые произошли в ней начиная с этого и кончая каким-то другим моментом. Можно считать это вариантом ретроспекции для ограниченного набора таблиц.

Дополнительно можно указать, что конкретная версия данных в рабочей области действительна только в течение определенного промежутка времени. Например, можно внести изменения, которые будут видны пользователям рабочей области в течение следующих 24 часов, а потом исчезнут.

У рабочих областей имеются собственные механизмы блокировки, применимые только к другим пользователям внутри данной области. Можно поставить монопольную блокировку на строку, но она предотвратит доступ к этой строке лишь для пользователей рабочей области. Пользователи, не входящие в рабочую область, по-прежнему могут читать и изменять исходные данные. Такой механизм имеет смысл, поскольку и блокировки, и рабочие области призваны изолировать данные от изменений. Рабочая область существует вне границ стандартной базы данных, поэтому блокировки в ней и в стандартной базе не должны зависеть друг от друга.

## Операции в рабочей области

К рабочим областям применимы три основных операции:

*Откат (rollback)*

Можно откатить изменения и вернуть рабочую область в состояние на момент ее создания. Разрешается также расставить точки сохранения, позволяющие выполнить откат к состоянию на предыдущий момент времени.

*Актуализация (refresh )*

Под актуализацией понимается синхронизация данных в рабочей области с данными в основной базе. Это может оказаться полезным, если требуется создать рабочую область как моментальный снимок данных на конец дня. В полночь рабочая область актуализируется и отражает состояние базы за предыдущий день.

*Слияние (merge)*

Операция слияния переносит изменения, произведенные в рабочей области, в ее родительскую область.

Легко понять, что операции актуализации и слияния могут приводить к конфликтам между данными в рабочей области и в ее родителе. Система управления рабочими областями отслеживает конфликты для каждой таблицы; разрешить их можно вручную.

## Усовершенствования в механизме рабочих областей

Workspace Manager тесно интегрирован с СУБД Oracle. В версии Oracle Database 10g менеджер рабочих областей был усовершенствован и теперь допускает экспорт и импорт многоверсионных таблиц и массовую загрузку данных в такие таблицы с помощью программы SQL*Loader. Также определены события, возникающие при операциях с рабочими областями, и разрешается создавать непрерывно актуализируемые рабочие области.

В версии Oracle Database llg эта тенденция получила развитие: добавлены подсказки оптимизатору и расширился спектр обслуживающих операций, применимых к многоверсионным таблицам.

# Лекция 9. Oracle и обработка транзакций.

## Основы OLTP

Прежде чем обсуждать специфику OLTP в Oracle, приведем стандартное определение оперативной обработки транзакций.

## Что такое транзакция

Напомним, что *транзакцией* называется логическая единица работы, которая должна быть выполнена полностью или не выполнена вовсе. Любая транзакция обычно состоит из одной или

нескольких команд языка манипулирования данными (DML), таких как INSERT, UPDATE, DELETE, и завершается либо командой COMMIT, делающей изменения постоянными, либо командой ROLLBACK, отменяющей изменения.

В классической книге об OLTP Джима Грея (Jim Gray) и Андреаса Ройтера (Andreas Reuter) «Transaction Processing: Concepts and Techniques» (издательство Morgan Kaufmann, см. приложение 2) было введено понятие ACID-свойств транзакции. Транзакция должна обладать следующими свойствами:

*Атомарность (Atomic)*

Транзакция выполняется или не выполняется как единое, неделимое целое.

*Непротиворечивость (Consistent)*

Завершенная транзакция оставляет измененные данные в непротиворечивом корректном состоянии.

*Изолированность (Isolated)*

Любая транзакция выполняется изолированно и не оказывает влияния на состояние других транзакций.

*Долговечность (Durable)*

Изменения, произведенные зафиксированной транзакцией, постоянны.

Если транзакции выполняются последовательно - одна за другой, - то гарантировать ACID-свойства относительно просто. Каждая транзакция начинается в непротиворечивом состоянии, оставленном предыдущей транзакцией, и в свою очередь оставляет данные в непротиворечивом состоянии для следующей транзакции. В случае конкурентного доступа для обеспечения ACID-свойств одновременно выполняющихся транзакций необходимы сложные механизмы блокировки и координации.

## Что означает OLTP

Оперативную обработку транзакций можно определять по-разному: как вид вычислений, обладающий определенными характеристиками, или как способ вычислений, противоположный традиционной пакетной обработке.

### Общие характеристики

Большинству OLTP-систем присущи следующие общие характеристики:

*Плотный поток транзакций и большое количество пользователей*

Во многих компаниях OLTP-системы - основа повседневной работы, поэтому нагрузка на них и количество пользователей больше, чем у любой другой используемой в организации системы.

*Четко определенные требования к производительности*

OLTP-системы - стержень важнейших бизнес-операций, поэтому время реакции на действия пользователя должно быть предсказуемым. При разработке OLTP-систем часто заключается соглашение об уровне обслуживания (Service Level Agreement), в котором оговорено ожидаемое время реакции.

*Высокая доступность*

Как правило, такие системы считаются ответственными для работы предприятия (mission-critical), простой которых обходится весь- мадорого.

*Масштабируемость*

Способность к наращиванию потока транзакций без существенного снижения производительности позволяет OLTP-системам переносить колебания активности пользователей.

В двух словах, OLTP-система должна обеспечивать предсказуемую производительность в любое время независимо от нагрузки. Любые неполадки в таких системах могут повлиять на работу организации в целом, что отразится на доходах и прибыльности.

### Оперативная и пакетная обработка

Оперативная обработка транзакций подразумевает прямое диалоговое взаимодействие между системой и ее пользователями. Пользователь вводит или запрашивает данные с помощью форм, взаимодействуя с обслуживающей базой данных. Редактирование и контроль данных производятся во время выполнения транзакций, инициируемых пользователями.

Пакетная обработка происходит без участия пользователей. Пакеты транзакций передаются обрабатывающей системе из файлов. Ошибки обычно записываются в протоколы и анализируются пользователями или операторами позднее. Практически во всех OLTP-системах имеются пакетные компоненты: задания, которые можно выполнить в периоды затишья. Это может быть составление отчетов, генерация платежных ведомостей, выполнение бухгалтерских проводок и так далее.

Во многих крупных компаниях большие ЭВМ, ориентированные на пакетную обработку, настолько глубоко интегрированы в корпоративную инфраструктуру, что отказ от них или замена чем-то другим невозможны. На практике часто создают «фронтальные» OLTP-системы, предоставляющие осовремененный интерфейс к таким унаследованным решениям. Пользователи вводят транзакции в OLTP-систему. На основе введенных данных создаются пакетные файлы, которые подаются на вход унаследованным приложениям.

По завершении пакетной обработки из пакетной системы извлекаются данные, с помощью которых актуализируется OLTP-система. В результате пользователи имеют развитый интерфейс, допускающий оперативную проверку и редактирование информации, а поток данных через модернизированную пакетную систему не прерывается. Хотя, на первый взгляд, такой процесс обходится дорого, обычно он все же более приемлем, чем радикальная замена старой системы. Проблема осложняется еще и тем, что старые системы часто плохо документированы, а те, кто знает, как они устроены, уже работают в другом месте или ушли на пенсию.

Индустрия финансовых услуг - лидер в технологии обработки транзакций, поэтому интерфейс с унаследованными приложениями нередко можно встретить в банках и страховых компаниях. Например, с помощью фронтальных оперативных систем обычно вводятся заявления о выплате страхового возмещения. После того как заявление введено и одобрено, оно передается в унаследованную систему для дальнейшей обработки и выплаты.

Цель таких средств Oracle, как переносимые табличные пространства и подсистема Streams, - обеспечить необходимую распределенным OLTP-системам функциональность в более краткие сроки, чем у традиционных пакетных систем.

## OLTP и средства бизнес-анализа

Смешанная рабочая нагрузка - OLTP и генерация отчетов - источник многих проблем обеспечения производительности и предмет жарких споров. Индустрия хранилищ данных зародилась в результате осознания того факта, что OLTP-системы не способны гарантировать требуемую пропускную способность, одновременно обслуживая нагрузку, связанную с обработкой огромных объемов исторических данных и выполнением произвольных запросов, которые часто предъявляют сотрудники, занятые, например, анализом многолетних трендов.

Проблема не в дефиците вычислительных мощностей; сами способы моделирования, хранения и доступа к данным принципиально различны. Цель проектирования OLTP-систем - анализ и автоматизация бизнес-процессов с гарантированной производительностью на хорошо известном наборе транзакций и пользователей. Рабочая нагрузка создается множеством коротких и четко определенных транзакций, в основном, транзакций записи.

Система бизнес-анализа обычно оперирует более крупными объемами данных, которые зачастую собираются из разных источников и содержат длинные истории. Схема хранилища данных, как правило, далека от полной нормализации, оптимальной для OLTP-систем. Кроме того, хранилища данных поддерживают произвольные запросы, даже малое количество которых - серьезная нагрузка для системы ввиду сложности и объема обрабатываемых данных.

Генерация отчетов и опрос данных есть и в составе OLTP-системы, но масштаб и частота обычно контролируются более строго, чем в хранилищах данных. Например, в банковской OLTP-системе могут встречаться запросы о состоянии клиентов и о балансах счетов, но не транзакции, охватывающие несколько лет.

Типичная OLTP-система строится из форм, с помощью которых можно предъявлять строго определенные запросы, выполняемые эффективно и не потребляющие ресурсы сверх меры. Однако поспешные выводы - например, что в OLTP-системе не может быть развитых средств запроса, - не всегда справедливы. Ввод/вывод в большинстве OLTP-систем примерно на 70-80% состоит из операций чтения и на 20-30% - из операций записи. Транзакции обычно запрашивают какие-то данные, например коды товаров, имена клиентов, балансы счетов, уровни складских остатков и так далее. Запросы, оптимизированные под конкретные бизнес-функции, - основной вид нагрузки на OLTP-систему. Произвольные запросы, для обработки которых нужно привлекать обширный набор данных, встречаются гораздо реже.

Системы бизнес-анализа на базе хранилищ данных и OLTP-системы могут обращаться, по сути, к одной и той же информации, но требования, предъявляемые к ним в плане мощности процессоров, объема оперативной памяти и организации хранения данных, как правило, совершенно различны. Поэтому попытка поддержать смешанную нагрузку оказывается гибельной для обеих систем. Продукт Real Application Clusters, появившийся в версии Oracle Database 10g* и реализующий динамическое предоставление обслуживания, позволяет выделять разные узлы под разные виды нагрузки. Кроме того, он в какой- то мере решает проблему смешанной нагрузки на одну базу данных (правда, с применением нескольких экземпляров).

## Развитие поддержки OLTP в Oracle

СУБД Oracle добилась широкого признания как основная СУБД для реализации OLTP-систем в вычислительных центрах среднего масштаба. В версии Oracle6 были реализованы нерасширяемые блокировки на уровне строк и согласованность по чтению (две важнейших особенности OLTP в Oracle), но настоящий рост числа покупателей Oracle для OLTP-обработки начался с выходом версии Oracle7. Именно в ней появились многие ключевые средства, а именно:

- многопоточный сервер Multi-Threaded Server (MTS), ныне - разделяемый сервер;
- разделяемый SQL;
- хранимые процедуры и триггеры;
- поддержка стандарта XA;
- распределенные транзакции и протокол двухфазной фиксации;
- репликация данных;
- Oracle Parallel Server (OPS).

В версии Oracle8 имеющаяся функциональность была расширена и дополнена новыми возможностями:

- пул соединений;
- мультиплексирование соединений;
- секционирование данных;
- Advanced Queuing (AQ);
- индекс-таблицы;
- распределенный менеджер блокировок Distributed Lock Manager (DLM) для Oracle Parallel Server;
- триггеры для реплицированных таблиц и параллельного распространения реплицированных транзакций.

В версию Oracle8i были добавлены следующие усовершенствования, имеющие отношение к OLTP:

- поддержка Java в ядре СУБД;
- поддержка распределенных компонентных технологий: CORBA V2.0 и Enterprise JavaBeans (EJB) vl.0;
- обмен сообщениями типа публикация/подписка на базе технологии Advanced Queuing;
- оперативное перестроение и реорганизация индексов;
- менеджер ресурсов Database Resource Manager (DRM);
- запросы к резервной базе;
- пакеты репликации, позволяющие применять транзакции на удаленных узлах.

В Oracle9i эта тенденция получила дальнейшее развитие с появлением технологии Real Application Clusters, распространившей достоинства Oracle Parallel Server на OLTP-приложения. Начиная с версии Oracle Database 10g технология Real Application Clusters поддерживает развертывание для модели grid-вычислений. Однако многие возможности, обеспечивающие оперативную обработку транзакций, много лет присутствовали в ядре Oracle.

## Архитектуры OLTP

Хотя все OLTP-системы ориентированы на решение одних и тех же задач, их внутренняя архитектура может отличаться, и это позволяет развертывать OLTP-приложения в традиционной двухуровневой модели, в трехуровневой модели с сервером приложений и в централизованной модели, применимой к веб- и grid-системам.

### Традиционная двухуровневая архитектура кпиент/сервер

Расцвет двухуровневых клиент-серверных приложений пришелся на конец 1980-х. В этой конфигурации ПК выступали в роли клиентов, обращающихся к серверу базы данных по сети. На стороне клиента работали графический интерфейс пользователя (ГИП) и прикладная логика, поэтому таких клиентов стали называть *толстыми*. Сервер обрабатывал SQL-команды и возвращал результаты клиентам. Имелись визуальные инструменты, упрощающие разработку клиентских приложений, но развертывать и обслуживать клиент-серверные системы было довольно трудно. Требовалась широкополосная сеть, а клиентское ПО приходилось устанавливать и регулярно обновлять на каждом пользовательском ПК.

Двухуровневая архитектура показана на рис. 9.1.

### Хранимые процедуры

В версии Oracle7 появилась возможность писать хранимые процедуры на языке PL/SQL,

разработанном корпорацией Oracle для программирования прикладной логики. Процедуры хранятся в базе данных, и для их выполнения клиент осуществляет вызов удаленной процедуры (RPC), а не посылает SQL-команды. Вместо того чтобы писать несколько команд, зачастую перемежающихся той или иной программной логикой, клиенту достаточно вызвать одну процедуру, передав ей параметры. Все SQL-команды и сопутствующие логические конструкции будут выполнены самой базой данных.

Кроме того, хранимая процедура скрывает от клиента внутренние изменения в структурах данных и логике программы. При условии, что параметры, которые клиент передает и получает назад, не изменяются, модифицировать клиентское ПО не нужно. Хранимые процедуры перемещают часть прикладной логики из клиента на сервер базы данных. Это позволяет заметно сократить сетевой трафик, а значит, повышает масштабируемость двухуровневых систем.

## Трехуровневые системы

OLTP-системы с большим количеством пользователей и транзакций обычно развертываются с применением трехуровневой архитектуры. В прошлом такая архитектура подразумевала наличие монитора транзакций, но теперь чаще применяется сервер приложений. Клиентыобращаются к монитору транзакций или к серверу приложений, размещенному на промежуточном уровне, который, в свою очередь, обращается к серверу базы данных. Идея монитора транзакций восходит к самым первым OLTP-системам, работавшим на больших ЭВМ. Разумеется, в этом окружении вся логика исполнялась на одной машине. В открытой же архитектуре сервер приложений, как правило, работает на отдельном компьютере (или нескольких компьютерах), образуя промежуточный уровень между клиентами и сервером базы данных.

Есть разные классы серверов приложений:

- Старые патентованные продукты типа Tuxedo от BEA Systems на платформах UNIX и Windows или CICS от IBM на больших ЭВМ.
- Стандартные серверы на базе Java 2 Enterprise Edition (J2EE).
- Сервер приложений на базе каркаса Microsoft .NET, поставляемый в составе серверных операционных систем Microsoft, например Windows 2000 или Windows 2003.

Серверы приложений организуют окружение для работы служб, вызываемых клиентами. Клиент не взаимодействует напрямую с сервером базы данных. Вызовы служб, предоставляемых монитором транзакций, во многом похожи на описанную выше архитектуру хранимых процедур, поэтому системы на базе хранимых процедур иногда называют «TP-Lite» (облегченный монитор транзакций).

Но серверы приложений предлагают и ряд дополнительных полезных служб, например:

*Суммирование (funneling)*
: Как и разделяемые серверы Oracle, серверы приложений поддерживают пул разделяемых служб, предоставляемых всем пользователям. Вместо того чтобы напрямую обращаться к базе данных, клиент вызывает службу, работающую под управлением монитора транзакций или сервера приложений, а эта служба связывается с сервером базы данных.

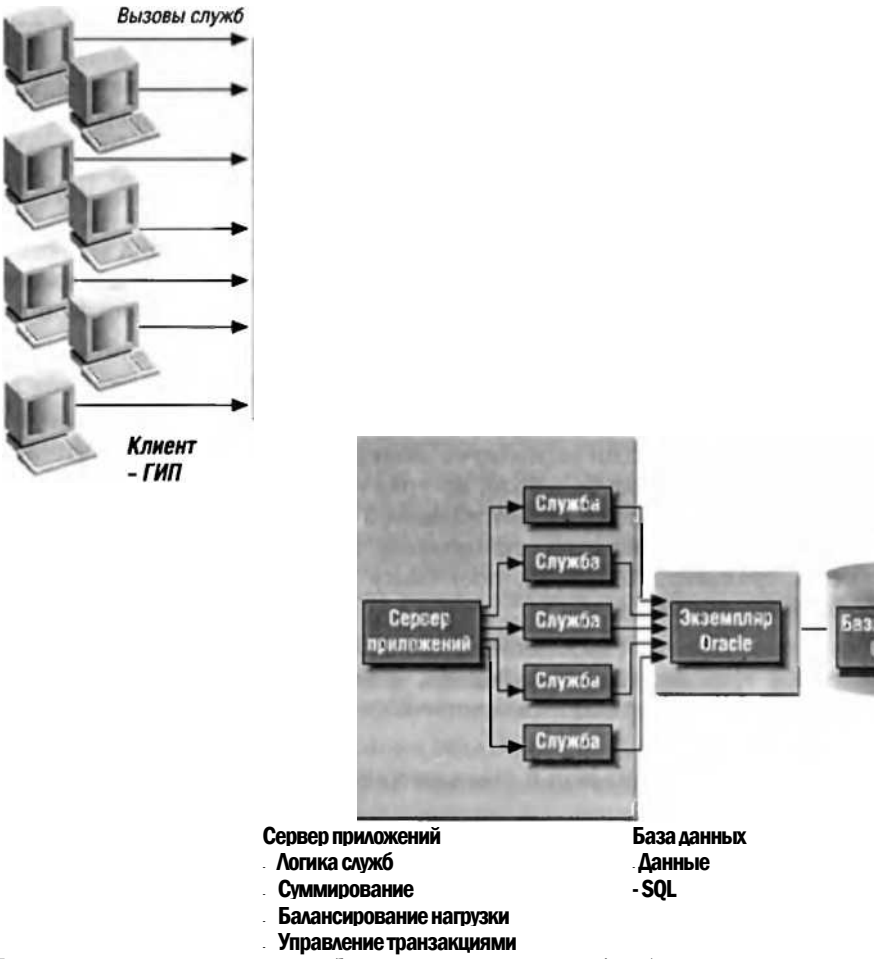
*Пул соединений (connection pool)*
: Сервер приложений организует пул разделяемых постоянных соединений с базой данных, которые использует от имени клиентов для передачи их запросов. Тем самым удается избежать накладных расходов на создание отдельного сеанса для каждого клиента.

*Балансирование нагрузки (load-balancing)*
: Запросы клиентов равномерно распределяются между несколькими разделяемыми серверами, работающими на одной или нескольких машинах. Сервер приложений может направлять запрос от имени клиента наименее загруженному серверу базы данных и при необходимости порождать дополнительные разделяемые серверы.

*Отказоустойчивость (fault-tolerance)*
: Сервер приложений играет роль менеджера транзакций; фиксацию или откат транзакции выполняет монитор. Собственно СУБД становится менеджером ресурсов, но транзакциями не управляет. Если сервер базы данных аварийно завершится во время выполнения транзакции, то после восстановления сервер приложений может начать транзакцию заново, поскольку управление транзакциями возложено именно на него.

  Такая способность к восстановлению была присуща уже старым мониторам транзакций типа Tuxedo, а более современные серверы приложений предлагают аналогичные возможности, описанные в стандартах.

*Маршрутизация транзакций (transaction routing)*
: Логические компоненты на промежуточном уровне могут отправлять транзакции конкретным серверам баз данных, что повышает масштабируемость.




*Гетерогенные транзакции (heterogeneous transactions)*

Серверы приложений могут управлять транзакциями, включающими разнородные серверы баз данных, например обновлять данные в базах, управляемых СУБД Oracle и DB2.

Хотя разработка трехуровневых OLTP-систем сложна и требует специальных навыков, преимущества весьма существенны. Системы, включающие серверы приложений, обеспечивают более высокую масштабируемость, доступность и гибкость по сравнению с простыми двухуровневыми системами. Для решения вопроса о том, какая архитектура в наибольшей степени отвечает потребностям вашей OLTP-системы, необходимо тщательно оценить и взвесить затраты, квалификацию имеющихся сотрудников, характеристики рабочей нагрузки, требования к масштабируемости и доступности.

На рис. 9.3 показана трехуровневая система с сервером приложений.

## Серверы приложений и веб-серверы

На промежуточном уровне системы, использующей Сеть, обычно находится сервер приложений и/или веб-сервер. Эти серверы предоставляют примерно те же службы, что и ранее описанный сервер приложений, но в большей степени ориентированы на Сеть, используя HTTP, HTML, CGI и Java.

*Рис. 9.3. Трехуровневая архитпектпура*

Серверы приложений на базе J2EE и .NET за последние десять лет сильно эволюционировали, заменяя устаревшие мониторы транзакций. Разные компании работают с различными стандартами - некоторые склонны использовать открытую технологию J2EE, другие предпочитают тесную интеграцию с предложениями Microsoft, в частности, с платформой .NET. Подробное обсуждение сравнительных достоинств J2EE и .NET, а также технологий построения серверов приложений выходит за рамки данной книги. Достаточно сказать, что серверы приложений играют чрезвычайно важную роль в архитектуре современных систем, и персонал отдела обслуживания баз данных должен понимать принципы создания N-уровневых архитектур.

На рис. 9.4 показана N-уровневая система, в которой имеются клиент, веб-сервер, сервер приложений и сервер базы данных.


## Решетка (grid)

Версия Oracle Database 10g ориентирована на другую разновидность архитектуры - grid-вычисления. Реальная топология решетки нас здесь не интересует, поскольку смысл этой технологии - предложить чрезвычайно простой интерфейс, прозрачно для пользователя выполняющий соединение с гибким источником вычислительных мощностей.

Таким образом, решетка позволяет ИТ-отделу получить все преимущества более сложных архитектур, не усложняя жизнь пользователей,
*Рис. 9.4. N-уровневая система*

а OLTP-системы развертываются на вычислительных ресурсах решетки.

# Поддержка OLTP в Oracle

В СУБД Oracle есть немало средств, обеспечивающих производительность, надежность, масштабируемость и доступность OLTP-систем. В этом разделе мы познакомимся с основными из них. Изложение ни в коем случае не является исчерпывающим, это лишь введение. Дополнительную информацию вы найдете в официальной документации Oracle и других источниках.

## Общие средства обеспечения конкурентного доступа и производительности

Уже отмечалось, что Oracle предлагает великолепную поддержку конкурентности и производительности в OLTP-системах. Перечислим некоторые основные средства, относящиеся к OLTP.

*Нерасширяемая блокировка на уровне строк*

Oracle блокирует только те строки, с которыми работает транзакция, и никогда не расширяет блокировку до уровня страницы или таблицы. В некоторых СУБД блокировка расширяется со строки на страницу, когда внутри страницы заблокировано достаточно много строк. Это приводит к неоправданным конфликтам, если пользователю нужны строки, которые больше никем не используются, но тем не менее недоступны из-за того, что содержащая их страница заблокирована.

*Многоверсионная согласованность по чтению*

Oracle обеспечивает согласованность данных на уровне одной команды или целой транзакции, не ставя блокировок чтения. Гарантируется, что запрос увидит только данные, зафиксированные на момент начала запроса. Изменения, произведенные транзакциями, которые еще не завершились в момент начала запроса, невидимы. Эффект транзакций, запущенных после начала запроса и зафиксированных до его окончания, также не виден в запросе. Для получения данных в том виде, в котором они существовали на момент начала запроса, Oracle пользуется сегментами отката. Тем самым удается избежать выбора между двумя равно неприятными вариантами: позволить запросу видеть незафиксированные данные (грязное чтение) или разрешить «читателям» блокировать «писателей» (и наоборот). Кроме того, в любой момент времени обеспечивается согласованный моментальный снимок данных.

*Разделяемые SQL-команды*

Разбор SQL-команд отнимает довольно много процессорного времени. Oracle кэширует разобранные и оптимизированные команды в специальной области внутри разделяемого пула. Если другой пользователь захочет выполнить уже кэшированную команду, то издержек на разбор и оптимизацию не будет. Однако повторно могут быть использованы только абсолютно идентичные команды- любые различия в количестве пробелов, символов новой строки или в регистре букв существенны. Для OLTP-систем характерно большое количество пользователей, работающих с одним и тем же приложением. Это идеальная ситуация для повторного использования разделяемых SQL-команд.

*Хранимые схемы планов выполнения*

В версии Oracle8i появилась поддержка устойчивых планов выполнения, которые иногда называют *связанными планами* (bound plans). Способ выполнения SQL-команды критически важен для производительности. После того как разработчик или администратор базы данных добился оптимальной эффективности некоторой команды, он может заставить оптимизатор Oracle применять тот же самый план вне зависимости от изменений в окружении. Это гарантирует стабильность и предсказуемость при модернизации ПО, изменениях схемы, объемов данных и так далее. В Oracle9i администраторам разрешено редактировать хранимые схемы планов.

Начиная с версии Oracle Database 10g можно заставить оптимизатор выбрать более удачный план выполнения плохо написанной SQL-команды, чтобы повысить производительность OLTP, не переписывая SQL. Консультант SQL Tuning Advisor выполняет такую оптимизацию SQL-команды и создает специально для нее улучшенный SQL-профиль. Во время выполнения созданный профиль используется вместо исходного плана.

## Масштабируемость

Разделяемый сервер и менеджер ресурсов Database Resource Manager помогают СУБД Oracle поддерживать много пользователей, решающих разные задачи.

### Многопоточный, или разделяемый, сервер

В версии Oracle7 был реализован многопоточный сервер (Multi-Threa- ded Server, MTS; в версии Oracle9i он стал называться разделяемым сервером) для поддержки большего числа одновременно работающих пользователей. Хотя разделяемый сервер позволил сократить количество серверных процессов, каждому клиенту по-преж- нему необходимо отдельное сетевое соединение. Количество сетевых соединений не безгранично, поэтому в Oracle8 предложено два решения по расширению возможностей сетевого уровня сокетов, являющегося частью операционной системы.

*Пул соединений Oracle Net*

Позволяет всем клиентам разделять общий пул физических сетевых соединений. Для простаивающих клиентов незаметно устраивается «тайм-аут», авыделенные им соединения возвращаются в пул, откуда могут быть переданы активным клиентам. Однако простаивающий клиент сохраняет виртуальное соединение с Oracle и, вернувшись к работе, получит новое физическое соединение. В модели безопасности Oracle аутентификация отделена от конкретного соединения, поэтому одно и то же соединение из пула может в разные моменты времени представлять разных пользователей. Пул соединений удобен для приложений, подключенных к базе, но не проявляющих высокую активность (например, почтовые системы).

*Oracle Net Connection Manager*

Уменьшает общее количество сетевых соединений с сервером базы данных. Клиенты подключаются к промежуточной машине, на которой работает менеджер соединений Oracle Net Connection Manager (CMAN). Тот мультиплексирует трафик от нескольких клиентов в одном соединении с диспетчером Oracle Net, работающим на сервере базы данных. В отличие от пула соединений, здесь нет понятия «тайм-аута», виртуального соединения клиента с сервером. В сети Oracle может быть несколько машин, где работает Connection Manager, что дает дополнительную масштабируемость и отказоустойчивость.

В терминах масштабируемости пул соединений можно считать средним, а мультиплексирование посредством Connection Manager - тяжеловесным решением. На рис. 9.5 представлены обе технологии масштабирования сети.

В версии Oracle Database 10g компонент Connection Manager стал более гибким; теперь допускается динамическое изменение его параметров без перезагрузки. Кроме того, был усовершенствован механизм правил доступа для фильтрации трафика CMAN.

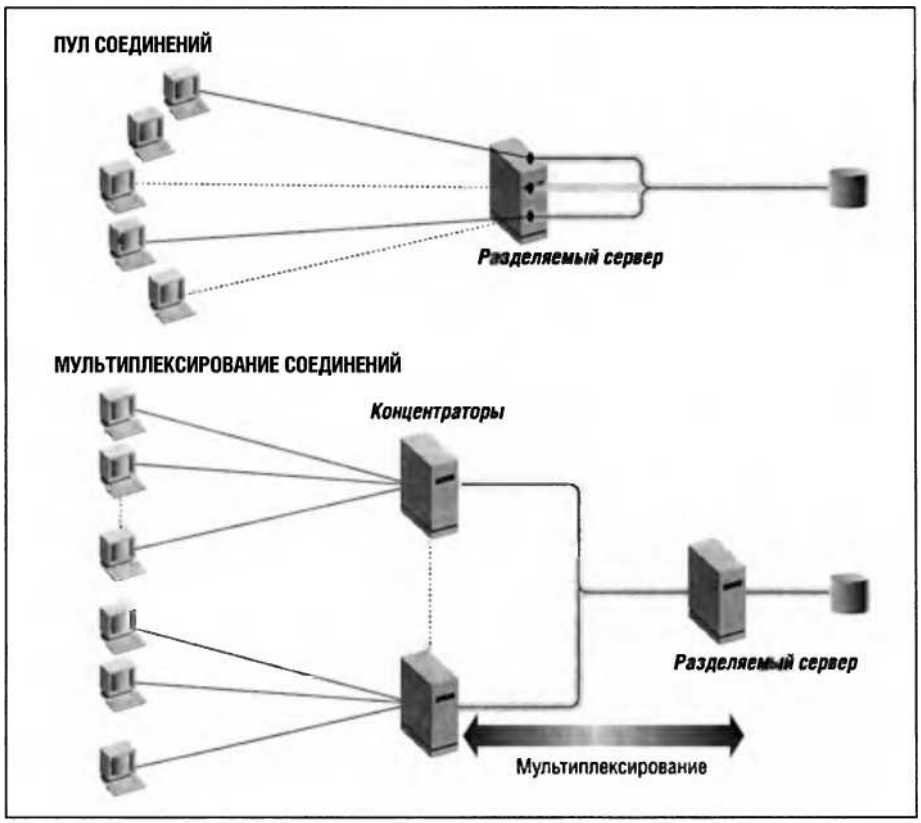



*Рис. 9.5. Масштабирование сети Oracle Net*

## Database Resource Manager

В состав версии Oracle8i включен менеджер ресурсов Database Resour- ce Manager (DRM), который упрощает и автоматизирует управление смешанной нагрузкой, когда разные пользователи обращаются к базе данных с разными целями. Можно определить группы потребителей, включив них отдельных пользователей или целые группы. DRM выделяет ресурсы процессора и определяет степень параллелизма для каждой группы потребителей исходя из планов распределения ресурсов. В этом плане задаются ограничения на объем конкретных вычислительных ресурсов, доступных каждой группе потребителей. Тем самым администратор базы данных может гарантировать, что определенные типы пользователей получат достаточно ресурсов для обеспечения необходимой производительности.

### Переменныесвязывания и разделяемые SQL-команды

Мы уже говорили, что механизм разделяемых SQL-команд - ключ к построению высокопроизводительных приложений. В OLTP- приложении часто встречаются похожие команды, но с разными условиями WHERE. Однако Oracle умеет разделять только абсолютно идентичные SQL-команды.

Чтобы воспользоваться этим механизмом для команд, которые отличаются только значениями параметров в условии WHERE, можно применить переменные связывания (bind variables). Значения, подставляемые вместо переменных связывания, могут отличаться, но команда при этом будет считаться одной и той же.

Рассмотрим приложение, в котором одна из функций - повышение зарплаты работникам. Для этого выполняются примерно такие команды:

```
UPDATE emp SET salary = salary * (1 + 0.1)
WHERE empno = 123;
UPDATE emp SET salary = salary * (1 + 0.15)
WHERE empno = 456;
```

Видно, что они различаются, - идентификатором работника и коэффициентом повышения зарплаты. Чтобы воспользоваться механизмом разделения, их следует переписать, включив переменные связывания:

```
UPDATE emp SET salary = salary * (1 + :v_incr)
WHERE empno = :v_empno;
UPDATE emp SET salary = salary * (1 + :v_incr)
WHERE empno = :v_empno;
```

Теперь команды идентичны, поэтому могут разделяться. Приложению нужно будет лишь

передать разные значения переменных `:v_incr` и `:v_empno` - ОДдля работника с идентификатором 123 и 0,15 для работника 456. Oracle подставит указанные значения вместо переменных связывания в SQL-команду. Подстановка производится на *фазе связывания*, которая следует за *фазой разбора* и *фазой оптимизации*. Дополнительную информацию вы найдете в официальном руководстве Oracle по вашему любимому языку разработки.

В версии Oracle Database 10g* и более поздних имеются инструменты, позволяющие отыскать в приложении места, поддающиеся такой оптимизации.

Например, можно выделить 80% процессорного времени пользователям, занятым вводом заказов, а оставшиеся 20% - пользователям, запрашивающим отчеты. Тогда генерация отчетов не займет все ресурсы, мешая вводить заказы. Впрочем, если пользователи, которые вводят заказы, не выбирают своей квоты, то остальные смогут потреблять больше ресурсов. Однако как только нагрузка от ввода заказов возрастет, квота для генерации отчетов вернется на уровень 20%. Другими словами, для ввода заказов отводится не больше 80% процессорного времени, а на генерацию отчетов - не меньше 20% или больше, если интенсивность ввода заказов невелика. DRM позволяет динамически изменять параметры плана, не останавливая экземпляр.

В версии Oracle9i компонент Database Resource Manager был серьезно усовершенствован. Теперь администратор может задать максимальное число активных сеансов, доступных группе потребителей. Все последующие запросы на установление соединения от членов этой группы ставятся в очередь. Ограничив число соединений, можно избежать ситуации, когда после очередного запроса оказывается превышен некий лимит для группы, вследствие чего страдают все ее члены.

Кроме того, Database Resource Manager теперь умеет прогнозировать потребление процессорного времени операцией. Если выясняется, что операция превысит лимит, заданный для группы потребителей, то она не выполняется. Иными словами, особо ресурсоемкие операции даже не начинаются.

Наконец, начиная с версии Oracle9i DRM может автоматически переключать группы потребителей, если некая группа слишком долго оставалась активной. Так, можно автоматически переключиться с группы, ориентированной на короткие OLTP-операции, на другую группу, более подходящую для выполнения пакетных операций.

В версии Oracle Database Ю^членом группы потребителей можно быть имя службы, приложение, конкретный компьютер или имя пользователя в операционной системе.

## Технология Real Application Clusters

Пожалуй, самым крупным нововведением в версии Oracle9i стала технология Real Application Clusters (RAC), пришедшая на смену Oracle Parallel Server (OPS).

В первом издании этой книги мы описывали OPS как средство повышения производительности и масштабируемости для некоторых видов приложений, устроенных по типу хранилищ данных, - таких, где данные можно естественным образом секционировать, а превалирующей операцией является чтение. Причиной такой ограниченной применимости OPS было явление, названное *вытеснением из кэша* (pinging).

OPS и RAC позволяют нескольким машинам обращаться к одним и тем же файлам базы данных, расположенным на общем диске, который подключен физически либо выглядит физически подключенным усилиями программного обеспечения (рис. 9.6).


*Рис. 9.6. Архитектура RAC*

Такая архитектура позволяет добавлять в кластер новые машины, увеличивая тем самым суммарную вычислительную мощность системы. Но ее реализация в OPS порождала проблему,

связанную с тем, что страница может содержать несколько строк. Если некоторая машина в кластере хочет изменить строку, принадлежащую странице, которая уже модифицируется другой машиной, то эта страница должна быть сброшена в файл базы данных, находящийся на общем диске. Вот этот сценарий и называется *вытеснением*. Такое развитие событий приводит к избыточному вводу/выводу, что в свою очередь снижает общую производительность.

Традиционно эта проблема решалась просто недопущением - OPS рекомендовалось применять лишь в тех случаях, когда база данных не вызывает вытеснения из-за большого количества операций записи или когда можно разделить операции записи, так чтобы никакие два узла не выполняли их одновременно. Это ограничение заставляло тщательно оценивать, подходит ли приложение для работы с OPS, а в некоторых случаях приходилось даже перепроектировать приложение, чтобы обойти особенности OPS.

Технология Real Application Clusters решила проблему вытеснения. RAC в полной мере поддерживает технологию Cache Fusion, идея которой в том, чтобы сделать данные, хранящиеся в кэше на одной машине, доступными всем остальным машинам в кластере. Если какой-то машине понадобился блок, который используется в данный момент другой машиной либо просто находится в кэше другой машины, то этот блок напрямую передается запрашивающему узлу, обычно по высокоскоростной шине.
Наличие технологии Cache Fusion означает, что никаких специальных мер по предотвращению вытеснения принимать не нужно. Внедрение Real Application Clusters заметно повышает масштабируемость практически всех приложений без каких бы то ни было модификаций. Но хотя это действительно так, все же в некоторых OLTP-приложени- ях, развернутых на кластере (характеризующихся частыми модификациями в небольшом наборе листовых узлов определенных индексов), было бы разумно применять индексы по реверсированному ключу для равномерного распределения ключей по листовым узлам. Тем самым удастся избежать возможного снижения производительности в этом специальном случае.

Технология Real Application Clusters дает те же преимущества в плане доступности, что и OPS. Поскольку все машины в кластере разделяют общий диск, выход из строя одной машины еще не означает сбой базы данных в целом. Пользователям, подключенным к отказавшей машине, придется перейти на другую машину кластера, но сам сервер базы данных будет работать.

В версии Oracle Database 10g модель RAC, реализованная для кластеров, была обобщена до grid-вычислений. Теперь Oracle предлагает все компоненты, необходимые для реализации кластеров на различных платформах, в том числе менеджер томов и кластерное ПО. В Oracle 10g Release 2 стало возможно вести мониторинг работы различных узлов кластера и давать рекомендации по более эффективному распределению нагрузки между узлами.

## Высокая доступность

С эксплуатационной точки зрения, OLTP-системы - это электронная центральная нервная система компании, поэтому поддерживающие их базы данных должны быть постоянно доступны. СУБД Oracle включает средства обеспечения высокой доступности.

*Резервная база данных*

Oracle может обеспечить резервирование, поддерживая копию основной базы данных на другой машине - обычно в другом узле. Журналы с основного сервера транспортируются на резервный и там применяются, чтобы продублировать все произведенные в основной базе изменения. BOracle8i реализованы механизм автоматической транспортировки журналов на резервный узел и возможность открывать резервную базу в режиме чтения для генерации отчетов.

В версии Oracle9i Release 2 появилось понятие *логической резервной базы данных* (logical standby). В этом случае изменения передаются в виде SQL-команд, а не журналов, что позволяет использовать резервную базу и для других целей.

*Прозрачное восстановление приложений при отказе (TransparentAp- plication Failover, TAF)*

TAF- это программный интерфейс, позволяющий автоматически присоединить сеанс пользователя к другому экземпляру Oracle, если обслуживающий его экземпляр вышел из строя. Все запросы, которые в этот момент обрабатывались, возобновляются с последней извлеченной строки.

*Опция Oracle Streams/Advanced Queuing (AQ)*

Опция AQ в подсистеме Oracle Streams предоставляет способ асинхронной, или отложенной, межсистемной коммуникации, что позволяет разным системам работать более независимо. Уход от зависимостей между системами позволяет избежать «каскадных» сбоев, то есть одна из взаимосвязанных систем может продолжать работу, даже если другая вышла из строя. Например, Streams позволяет организовать передачу изменений из одной базы данных Oracle в другую. С помощью шлюзов можно даже передавать изменения в базу данных

другого производителя.

*Репликация посредством Oracle Streams*

Для повышения избыточности можно воспользоваться встроенным механизмом репликации Oracle. Произведенные транзакциями изменения можно синхронно или асинхронно реплицировать в другие базы. Если основная база данных откажет, то данные можно будет получить из других баз. Начиная с версии Oracle9i Release 2 репликация на основе журналов входит в состав подсистемы Streams.

*RealApplication Clusters (RAC)*

Технология Real Application Clusters повышает масштабируемость базы данных Oracle, делая ее доступной из нескольких узлов, объединенных в кластер. Однако RAC не только позволяет нескольким экземплярам обращаться к одной и той же базе, но и обеспечивает высочайший уровень доступности, защищая систему от последствий сбоя любого узла кластера. Если какой-то узел вышел из строя, остальные продолжают предоставлять доступ к базе. Эта идея получила дальнейшее развитие в grid-системах.

В версии Oracle Database llg реализован ряд дополнительных мер по обеспечению высокой доступности, в том числе возможность собирать диагностическую информацию обо всех сбоях в базах данных.

## Oracle Streams и Advanced Queuing

Технологии передачи сообщений появились не вчера и широко применяются в OLTP-приложениях. Как правило, речь идет о предоставлении надежного транспортного уровня для передачи сообщений между машинами по сети. В версии Oracle8 служба Advanced Queuing (AQ) была интегрирована в СУБД. А в версии Oracle9i Release 2 объединение AQ с репликацией на основе журналов привело к созданию подсистемы Oracle Streams.

Oracle Streams AQ обладает всеми достоинствами простых систем передачи сообщений, добавляя к ним хранение очередей сообщений в базе данных. Информация в очередях сообщений описывает критически важные для бизнеса события и должна храниться в надежном, безопасном месте, допускающем масштабирование и восстановление. Храня очереди в базе данных, мы распространяем на них все преимущества, характерные для СУБД.

Данные, передаваемые через очереди, отражают приливы и отливы деловой активности. Анализируя тип и объем трафика, можно составить представление о работе и взаимодействии различных бизнес-функций, что в свою очередь способно пролить свет на важные аспекты функционирования компании. AQ поддерживает механизм *хранилищ сообщений*, позволяющий опрашивать и подробно анализировать содержимое сообщений, которые уже находятся в базе данных. Oracle может логически исключать сообщения из очереди, оставляя находившиеся в них данные для последующего анализа истории.

Помещение в очередь и извлечение сообщений из нее может осуществляться в составе транзакции или как отдельное событие, которое происходит в момент выполнения соответствующей команды. Операции с очередью, включенные в контекст транзакции, фиксируются или откатываются вместе с этой транзакцией. В случае сбоя состояние очередей восстанавливается наравне с остальной базой данных.

Oracle располагает механизмом маршрутизации трафика сообщений, позволяющим пересылать сообщения из одной очереди в другую. На рис. 9.7 показано применение очередей и механизма распространения сообщений.

В Oracle Database 10g и последующих версиях стало проще работать с подсистемой Streams из программы - сообщения разрешено помещать и извлекать из очереди пачками, а нового кода для работы с очередями требуется меньше.

## Применение Streams для реализации системных интерфейсов

В реализации OLTP-систем неизбежна задача организации интерфейсов с другими системами в рамках одного предприятия или в других компаниях. На проектирование, разработку и управление такими интерфейсами тратятся значительные усилия, в совокупности это может составить 40-60% стоимости крупномасштабной системы управления ресурсами предприятия (ERP). Более того, добавление новой или из-

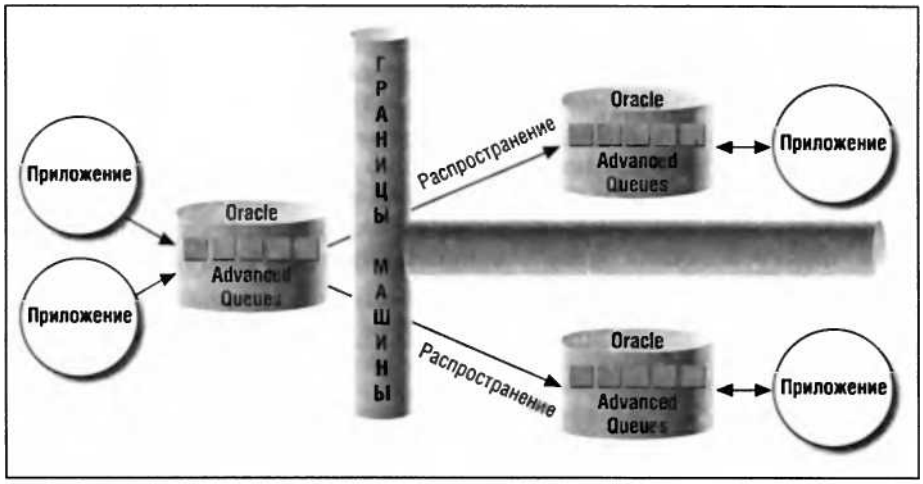



***Рис. 9.7.** Oracle Streams/Advanced Queuing*

менение имеющейся системы влечет за собой переработку интерфейсов, что ложится тяжким грузом на текущее сопровождение.

Подсистема Oracle Streams способствует решению проблемы интеграции засчет реализации архитектуры «колесо со спицами» (hub-and- spoke), применяющей сочетание технологий передачи сообщений, маршрутизации и трансформации. Традиционно разрабатывается специализированный интерфейс между двумя системами. Когда появляется третья система, приходится разрабатывать интерфейс между ней и остальными. Чем больше систем предстоит интегрировать, тем больше должно быть разработано нестандартных интерфейсов и тем сложнее оказывается сопровождение.

Но механизм очередей позволяет отдельным системам соединяться только со ступицей вдоль спицы, следовательно, разработки интерфейсов между каждой парой систем можно избежать. Спицы отправляют и принимают сообщения, а ступица обеспечивает маршрутизацию и трансформацию. Тем самым уменьшается количество интерфейсов, необходимых для объединения ряда систем. При добавлении новых систем не придется разрабатывать много новых интерфейсов. Нужно лишь подключить новую систему к ступице и настроить службы маршрутизации и трансформации. На рис. 9.8 показаны различия междутрадиционным подходом и архитектурой «колесо со спицами».

## Технология публикации/подписки в Oracle

В Oracle8i технология Advanced Queuing была дополнена механизмом публикации/подписки (publish-subscribe). Приложение может подписаться на очередь сообщений и указать атрибуты тех сообщений, в получении которых оно заинтересовано. Когда другое приложение публикует сообщение, помещая его в эту очередь, Oracle просматривает содержимое сообщения, выясняет, какие приложения в нем заинтересованы, и посылает им извещения. Например, приложение по отгрузке продукции может подписаться на очередь заказов, указав, что интерес для него представляют только сообщения о заказах в состоянии «Готов к отгрузке». И только такие сообщения оно и будет получать. Механизм публикации/подписки вкупе с механизмом распространения сообщений образует мощный каркас для организации потока информации между системами.

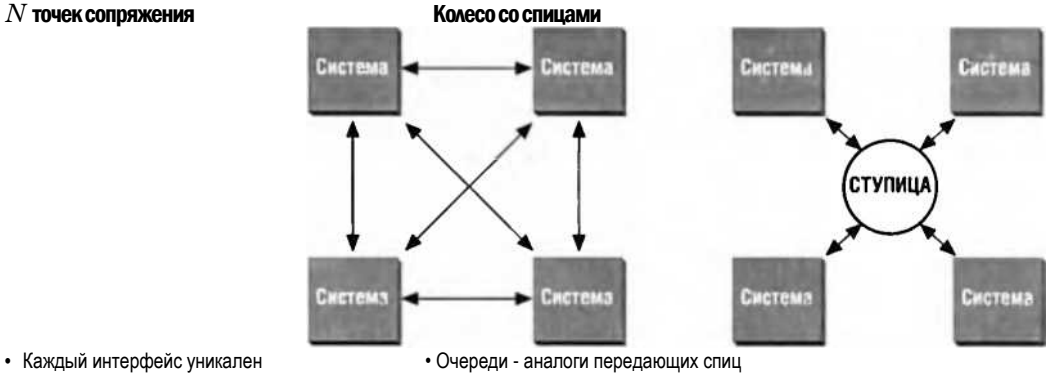


*Рис. 9.8. Специализированные интерфейсы и архитектура «колесо со спицами»*

## Объектные технологии и распределенные компоненты

Теоретически, чем больше объем информации, тем больше полезных сведений можно из нее извлечь. Но объединение информации, порождаемой в разных системах, может оказаться непосильной задачей, особенно принимая во внимание, что по мере добавления новых систем ее сложность возрастает в геометрической прогрессии.

Хотя технологии передачи сообщений могут помочь в организации интерфейса между различными системами, часто требуется и оперативное взаимодействие. Например, в отделе кадров имеется информация о работниках компании (скажем, о том, в каком отделе каждый работает и чем занимается), а отдел закупок мог бы воспользоваться этими данными на этапе оценки заказа на покупку. Система могла бы определить расходный лимит заказчика и определить отдел, на счет которого следует отнести затраты. На практике создание таких оперативных интерфейсов оказывается непростым делом, поскольку системы должны согласовать протокол обмена информацией и неукоснительно его придерживаться. Вкаждой системе имеются свои интерфейсы прикладного программирования (API), позволяющие обращаться к ней из других систем. Эти уникальные и зачастую противоречивые API ограничивают повторное использование имеющей функциональности.

Одно из возможных решений дают объектные технологии: системы взаимодействуют, вызывая методы объектов, а не функции конкретных API. Например, узнать, в каком отделе работает пользователь, поможет стандартный вызов метода объекта, представляющего этого работника в кадровой системе.

В Oracle8i и последующих версиях поддерживается целый ряд объектных технологий, включая использование объектно-ориентированного языка Java.

# Лекция 10. Хранилища данных и средства бизнес-анализа в Oracle.

## Основные понятия бизнес-анализа

Зачем создавать хранилище данных или средства бизнес-анализа? Почему данные в базе, поддерживающей оперативную обработку транзакций (OLTP), могут считаться только частью решения для бизнес- анализа? При проектировании хранилищ данных часто руководствуются следующими соображениями:

*В ходе стратегического и тактического анализа можно разглядеть в данных какие-то тренды*

Хранилища данных часто применяются для создания простых отчетов, основанных на агрегировании гигантских объемов данных. Для формирования таких агрегатов «на лету» с помощью OLTP-баз потребовалось бы столько ресурсов, что своевременная обработка транзакций стала бы невозможной. При выполнении подобных незапланированных запросов часто применяются встроенные аналитические функции СУБД, потребляющие очень много вычислительных ресурсов.

*Большая часть данных в хранилище предназначена только для чтения, обновления случаются редко*

Имеющиеся в СУБД средства позволяют создавать хранилища, содержащие сотни терабайтов данных, и даже обновлять некоторые данные почти в режиме реального времени.

*Данные в OLTP-системах не являются «чистыми» и могут быть рассогласованы*

Если ввод данных в OLTP-систему не очень тщательно контролируется, то возможны ошибки и дублирование. Нередко одним из важнейших этапов загрузки данных в хранилище является устранение таких ошибок. Кроме того, в разных OLTP-системах определения общих данных могут отличаться, и процедура загрузки может консолидировать определения.

*Для эффективной работы хранилища данных приходится отходить от стандартной нормализованной схемы, типичной для реляционной базы*

Обычно запросы предъявляются к некоей таблице фактов, содержащей сводные данные. Схема базы часто проектируется в виде *звезды*, что позволяет легко получать доступ к фактам в разрезе различных *измерений* (справочных данных). (Схема типа «звезда» подробнее

рассматривается ниже.) Например, пользователь может рассматривать объем продаж, хранящийся в таблице фактов, в разрезе регионов, региональных складов или отдельных наименований продукции - и все это можно назвать измерениями. В современных хранилищах данных часто встречаются и *гибридные схемы*, то есть комбинация схемы типа «звезда», характерной для предыдущего поколения «витрин данных» (data mart), и детальных данных, приведенных к третьей нормальной форме, как обычно делается в OLTP-системах.

## Эволюция средств бизнес-анализа

Идея сбора данных для бизнес-анализа не нова. Чуть ли не с момента появления первых компьютеров корпоративные данные применяются для принятия стратегических решений, а не просто для повседневных операций.

Довольно давно разработчики и пользователи оперативных систем осознали потенциал, который представляют для бизнеса сопутствующие системы анализа данных. В общем-то, на заре эры персональных компьютеров рост отрасли в немалой степени был обусловлен созданием электронных таблиц, занимающихся анализом данных, которые загружались из оперативных систем. Руководители стали направлять усилия ИТ-специалистов на разработку решений, позволяющих лучше понять функционирование предприятия и выработать новые стратегии развития. Ныне предлагаются готовые решения в таких областях, как управление взаимодействием с клиентами (CRM), анализ продаж и маркетинговых кампаний, управление продукцией, финансовый анализ, анализ цепочек поставщиков и анализ рисков и мошенничества.

В 1980-е во многих организациях стали выделять под такие приложения отдельные серверы, образующие так называемые *системы поддержки принятия решений* (decision support systems, DSS), которые дополняли обычную систему управления информацией. Запросы, характерные для DSS-систем, потребляли особенно много времени и памяти, осуществляя только чтение данных, тогда как в традиционных OLTP-системах обычно выполнялось много обновлений и, следовательно, операций ввода/вывода. Характеристики таких запросов были куда менее предсказуемы, чем в OLTP-системах. Это и привело к разработке хранилищ данных специально для систем поддержки принятия решений, отделенных от хранилищ, используемых в OLTP-системах.

Когда в начале 1990-х Билл Инмон (Bill Inmon), чьи работы упомянуты в приложении 2, и другие исследователи ввели в оборот термин «хранилище данных», возникла единая формализованная инфраструктура для построения требуемых решений. Топология систем бизнес-анализа продолжала развиваться, как мы увидим в следующем разделе. Современные решения часто включают инфраструктуру, позволяющую отражать в отчетах данные, получаемые как из хранилищ, так и из OLTP-систем. Прогресс в области аппаратных решений привел к тому, что теперь ввод/вывод стал более важным фактором при проектировании платформ для размещения хранилищ данных.

## Топология средств бизнес-анализа

Классическая топология хранилища данных, служащего источником всей корпоративной информации, состоит из нескольких уровней, представленных нарис. 10.1.

Есть ряд причин, по которым многолетние исследования привели именно к такой топологии. Первые попытки создать единое корпоративное хранилище данных оканчивались «аналитическим параличом» (analysis paralysis). Если определение корпоративной модели данных для OLTP-систем может длиться годами (из-за трений между отделами и масштабности задачи), то в случае хранилищ данных требуется гораздо больше времени, чем готовы были принять бизнесмены, финансирующие разработки. Дополнительная сложность заключалась в постоянном изменении требований, обусловленном изменяющимся рынком. Если элементы данных и требования к оперативным системам могут быть зафиксированы на достаточно длительное время, то уловить экономические тренды так же трудно, как поймать лису за хвост.

Поэтому любые попытки построить модель предприятия, которая устроила бы всех, приводят к результату, который не устраивает никого.

## Витрины данных

После того как попытки построить крупномасштабное единое для всего предприятия хранилище данных потерпели крах, наступило уныние и раздражительность. Нашлись люди, которые отреагировали на это, сосредоточившись на создании независимых *витрин данных* (da- ta mart), в которых были представлены данные, взятые из соответствующих оперативных систем. Многие витрины поначалу были вполне успешными, так как относительно быстро удовлетворили неотложные потребности бизнеса.

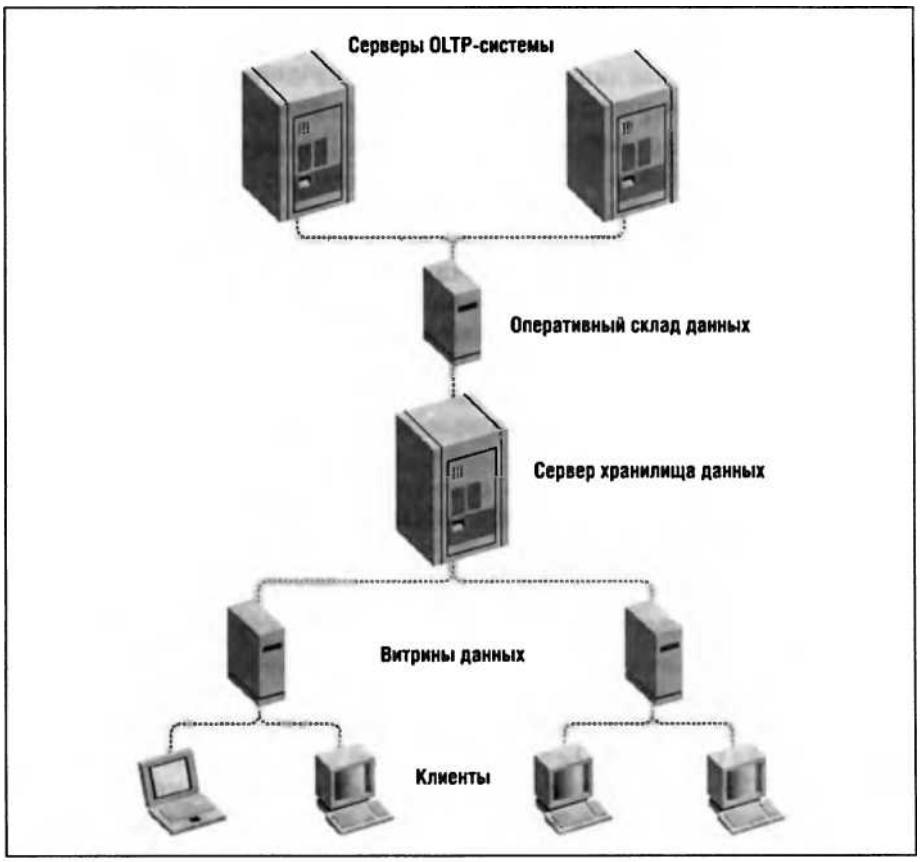



***Рис. 10.1.** Типичная начальная топология средств бизнес-анализа*

Однако вскоре начали проявляться и проблемы. Часто между отделами не было согласия относительно определения базовых понятий, например, что такое «клиент». Если руководитель высокого уровня задавал один и тот же вопрос главам нескольких отделов, то ответы, возвращаемые независимыми витринами данных, нередко оказывались различными. Аэто ставило под сомнение достоверность всех вообще витрин. Кроме того, у многих отделов постоянно возникали сложности с обслуживанием витрин и извлечением данных из оперативных источников (к тому же эти данные нередко дублировались).

Снова взглянув на свои решения, архитекторы начали осознавать, как важно иметь согласованное представление детальных данных на уровне корпоративного хранилища. Одновременно они увидели, что витрины данных могли бы решить проблемы бизнеса и постепенно дать возврат на инвестиции. Сегодня в наиболее успешных проектах одновременно строятся зависимые витрины данных, решающие по одной проблеме за раз, и инкрементно возводится здание корпоративного хранилища данных.

В настоящее время под витриной данных понимается просто узкоспециализированное хранилище данных, обычно реализуемое в рамках одного отдела. Как правило, витрины создаются с целью повышения производительности и могут включать много сводных таблиц. Первоначально витрины данных предполагались небольшими, поскольку нет необходимости загружать в них все данные одного отдела или данные из других отделов. Однако некоторые витрины необычайно разрослись, включая данные из внешних источников (иногда оплаченные), которые не имеют отношения к другим аспектам деятельности предприятия.

В некоторых организациях витрина данных создается для решения специфических задач какого-то проекта, и ее модель оптимизирована с учетом производительности именно для данного проекта. По завершении проекта такие витрины уничтожаются, а оборудование используется для других целей. По мере изменения требований, выдвигаемых бизнесом, топология хранилища данных может эволюционировать, и разработчики должны учитывать такую возможность.

Сосредоточение усилий на экономии затрат, управляемости и доказательстве соответствия заставило многих переосмыслить необходимость большого количества физически раздельных витрин данных. Поэтому наметилась тенденция к консолидации витрин в одно корпоративное хранилище данных. Последние версии Oracle позволяют эффективно управлять различными сообществами пользователей, что упрощает задачу консолидации.

## Оперативные склады данных и корпоративное хранилище данных

Концепция *оперативного склада данных* (operational data store, ODS) также приобрела популярность в 1990-е. Проще всего определить ODS как центр распределения текущих данных. Как и в случае OLTP-систем, схема нормализована и данные относительно свежие. ODS служит точкой консолидации для отчетов и тем местом, где можно посмотреть текущие данные, пересекающие границы отделов. Своей популярностью идея ODS отчасти обязана удобству применения в период слияний и приобретений одних компаний другими. ODS может выступать также в роли промежуточного пункта для трансформации данных перед отправкой в хранилище или на витрину.

Сервер хранилища, или *хранилище корпоративных данных*, - это склад исторической информации по разным предметам, поддерживающий несколько отделов и зачастую используемый как база данных о деятельности предприятия в целом. Если ODS уже имеется, то сервер хранилища может извлекать данные из него. В противном случае данные для хранилища извлекаются непосредственно из оперативных источников и по ходу дела трансформируются. В хранилище можно загружать и внешние данные.

Выше уже отмечалось, что сегодня популярна идея консолидации платформ. Хранилище корпоративных данных может стать местом консолидации ODS и различных витрин данных. Хотя различие в логических моделях остается, они консолидируются на единой платформе и СУБД.

## OLTP-системы и бизнес-анализ

В OLTP-системах хранятся текущие актуальные данные. Можно организовать порталы или инструментальные панели, где будут отображаться отчеты на основе информации, взятой как из систем обработки транзакций, так и из хранилищ данных. Ключ к успеху инструментальной панели - в предоставлении высококачественных данных с согласованной семантикой. Качество информации в OLTP-системе напрямую зависит от контроля входных данных с целью исключения дубликатов и ошибок.

Обеспечить согласованную семантику позволяют средства управления эталонными данными (master data management, MDM). Речь идет о концентраторах данных, которые служат единым эталоном для всех данных, необходимых для поддержки ключевых показателей бизнеса - клиентов, продукции или финансов. Oracle предлагает целый ряд концентраторов данных для этих и других аспектов бизнеса, чтобы на их основе можно было построить инфраструктуру.

Проекты, в которых используются данные из хранилищ, OLTP-систем и MDM-концентраторов, называются интеграционными. Во время работы над этой книгой в большинстве организаций, где были развернуты системы бизнес-анализа, основным источником исторических данных служила инфраструктура хранилища данных. Для устранения различий в общих элементах данных и их очистки применяются технологии извлечения, трансформации и загрузки (extraction, transformation and loading, ETL).

# Проектирование хранилища данных

Основой инфраструктуры бизнес-анализа служит база данных; это именно то место, в котором хранятся данные. Но бизнес-анализ одними данными не исчерпывается - инфраструктура приносит пользу, только когда пользователи могут получить из этих данных новые знания. Это может показаться тривиальной истиной, но нам встречались компании, которые создавали элегантную инфраструктуру, не спрашивая пользователей, каковы их потребности и какие ключевые показатели эффективности (КПЭ) следует измерять. Часто подобные проекты кончались печально - у них почти не было пользователей, к ним практически никто не обращался и никаких новых знаний о бизнесе они не порождали.

В предположении, что инфраструктура хорошо спланирована и на данные есть спрос, возникает следующая проблема - как этот спрос удовлетворить. Перед вами встанет задача спроектировать хранилище данных и другие компоненты инфраструктуры, обеспечивающие нужную пользователям производительность. Поначалу задача может даже показаться неподъемной, так как подразумевает обработку гигантских объемов исходных данных.

Приступая к проектированию, не забывайте также, что инфраструктура никогда не станет окончательной. По мере изменений требований бизнеса должны изменяться и составляющие ее компоненты. Таким образом, наличие средств для отслеживания изменения путем сохранения метаданных в некоем репозитории часто оказывается критически важным аспектом проекта.

Такие средства предоставляют различные инструменты проектирования. Программа Oracle Warehouse Builder (OWB), включенная в редакции СУБД Enterprise Edition, Standard Edition и Standard Edition One (начиная с 2006 года), содержит репозиторий метаданных и умеет импортировать метаданные из оперативных таблиц с последующим созданием новой схемы и

таблиц в ней. Проектировщик хранилища данных создает столбцы в новых таблицах и строит ограничения целостности для новой схемы. Между исходными и конечными столбцами создается отображение, включающее при необходимости трансформацию данных. DML-сценарии для создания новых таблиц, а также сценарии, ориентированные на PL/SQL или SQL*Loader, для процедуры извлечения, трансформации и загрузки генерируются автоматически.

Выше отмечалось, что исторически характеристики использования хранилищ данных были не такими, как у баз данных, предназначенных для OLTP-обработки. Одним из отличительных признаков хранилищ данных был высокий процент операций чтения. Это облегчало достижение приемлемой производительности. Модель блокировок Oracle идеально подходит для операций с хранилищами данных. Oracle не ставит блокировки на читаемые данные, что позволяет понизить количество конфликтов и требования к ресурсам в ситуации, когда есть много операций чтения. Поскольку блокировки не расширяются, Oracle прекрасно справляется с загрузкой данных в хранилище в режиме, близком в реальному времени. Возникающая при этом рабочая нагрузка не так уж сильно отличается от характерной для OLTP-систем.

Особенности работы с хранилищами данных диктуют развертывание иных видов схем. В OLTP-системах данные, участвующие в транзакции, обычно хранятся в нескольких таблицах, причем каждый элемент данных сохранен только один раз. Если требуется выбрать данные из нескольких таблиц, они соединяются. Как правило, оптимизатор решает, какую таблицу взять в качестве отправной точки для соединения, исходя из предположения, что данные во всех таблицах одинаково важны.

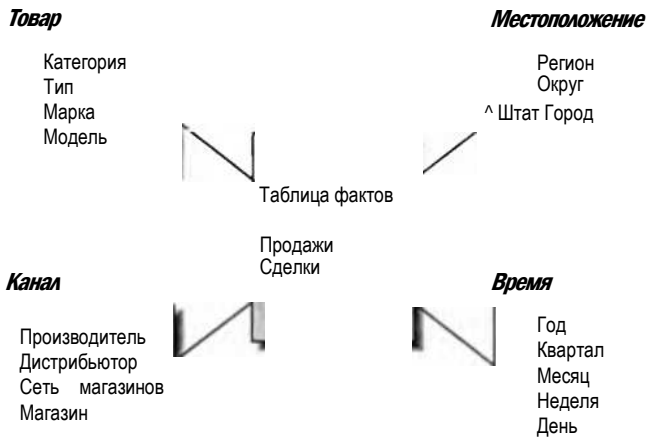
Хотя модель хранилища данных в Oracle иногда приводится к третьей нормальной форме, но если пользователям для формулирования запросов или аналитической обработки требуется более понятная схема, то основные данные лучше хранить в центральной таблице фактов, окруженной таблицами измерений, как показано на рис. 10.2. В таблицу фактов можно поместить сводные показатели для данных, представленных в хранилище и в другом виде, а таблицы измерений могут содержать различные иерархии. Выше уже отмечалось, что в наши дни многие организации, консолидирующие данные из витрин в хранилища, применяют гибридные схемы, в которых третья нормальная форма сочетается со «звездой».

Ральфу Кимбаллу (Ralph Kimball), автору широко известной книги «The Data Warehouse Toolkit» (издательство Wiley, см. приложение В), приписывают открытие того факта, что именно способ, которым большинство пользователей хранилищ данных формулируют запросы, делает схему типа «звезда» (см. рис. 10.2) наиболее подходящей моделью. Типичный запрос можно представить примерно так:

> Покажи, сколько компьютеров (товар) было продано через магазин (канал) в штате Висконсин (местоположение) за последние 6 месяцев (время).

В схеме на рис. 10.2 мы видим относительную большую таблицу данных о продажах (она называется *таблицей фактов*), окруженную

***Рис. 10.2. Типичная схема типа «звезда»***



меньшими таблицами (*измерения*, или *справочные таблицы)*. Приведенный выше запрос часто называют *многомерным*, поскольку в нем участвуют несколько измерений (и почти всегда одним из них оказывается время). Поскольку такие запросы типичны для хранилища данных, распознавание схемы типа «звезда» оптимизатором Oracle могло бы дать огромное повышение производительности.

## Оптимизация запросов

Распознавание схемы типа «звезда» оптимизатором впервые было добавлено в версию Oracle7 с прицелом на ускорение обработки запросов для систем бизнес-анализа в последующих версиях СУБД. В версии Oracle Database 10g* точность прогнозирования была улучшена за счет того, что оптимизатор сравнивает свой прогноз с реальной производительностью и автоматически вносит коррективы. Оптимизатор может также прозрачно переписывать запросы с агрегированием с помощью имеющихся в схеме материализованных представлений. В Oracle Database 1*lg* добавлена возможность переписывать запросы для опции OLAP Option, а также улучшилась обработка запросов с вложенными представлениями.

Как оптимизатор обрабатывает запрос к схеме типа «звезда»? Во-пер- вых, он находит таблицу фактов, содержащую данные о каждой продаже. Это служит признаком наличия звезды. По ходу эволюции версии Oracle7 оптимизатор научился порождать гораздо более разумные планы. В стандартной реляционной базе оптимизатор попытался бы соединить каждую таблицу измерений с таблицей фактов, по одной за раз. Но из-за того что таблица фактов обычно очень велика, многократные соединения с ней могут отнимать много времени.

В Oracle7 были добавлены соединения на основе декартова произведения, когда сначала соединяются все таблицы измерений, а на последнем шаге результат соединяется с большой таблицей фактов. Эта техника неплохо работает, если таблиц измерений не очень много (как правило, не больше шести, иначе декартово произведение получается слишком громоздким), а распределение данных сравнительно равномерно.

В некоторых случаях оказывается слишком много таблиц измерений или данные разрежены. Тогда оптимизатор может выбрать параллельное соединение типа «звезда» по битовым индексам.

В прежних версиях Oracle администратор базы данных должен был задавать некоторые параметры инициализации (например, STAR_TRA- NSFORMATION) и собирать статистику, чтобы оптимизатор сумел отыскать наилучший метод выполнения подобных запросов. Сегодня нужные параметры задаются на этапе установки по умолчанию, а СУБД собирает статистику автоматически.

## Битовые индексы и параллелизм

*Битовые индексы* впервые появились в версии Oracle7 для ускорения выборки данных и выполнения соединений в запросах, типичных для хранилищ данных. Как правило, битовые индексы строятся для столбцов, в которых кардинальность данных мала. *Кардинальностью* называется количество различных значений в столбце, поделенное на количество строк. Есть разные мнения относительно того, что считать малой кардинальностью. Некоторые считают небольшой даже кардинальность в 10%, но не забывайте, что в таблице смиллионом строк такая «малая» кардинальность означает наличие 100 000 различных значений!

Наличие 1 в битовом индексе означает, что соответствующее значение присутствует в строке, а 0 - что отсутствует. Битовая карта строится для каждого значения индексируемых столбцов. Поскольку компьютеры изначально оперируют нулями и единицами, такая техника может заметно повысить скорость выборки. Кроме того, логические операции типа AND сводятся к простым действиям над несколькими битовыми картами (в случае AND - к сложению). Дополнительное преимущество битовых индексов заключается в том, что они занимают гораздо меньше места на диске.

На рис. 10.3 иллюстрируется применение битового индекса для обработки составного условия WHERE. По существу, битовые индексы просто накладываются друг на друга, как перфокарты в колоде. Oracle ищет те позиции, в которых все биты подняты (то есть значение присутствует в обоих столбцах), как будто протыкает спицей колоду перфокарт в том месте, где все они пробиты.

В Oracle производительность запросов типа «звезда» улучшается, когда над таблицей фактов построены битовые индексы по столбцам внешних ключей, ссылающихся на таблицы измерений. При этом производится параллельное соединение по битовым индексам, так что из таблицы фактов извлекаются только необходимые строки, соединяемые с таблицами измерений. В процессе соединения автоматически распознается разреженность (то есть большое количество пустых значений), а число таблиц измерений не имеет значения. Этот алгоритм эффективно работает и для схемы типа «снежинка» - обобщения стандартной схемы типа «звезда», в которой для каждого измерения имеется несколько таблиц.

Чтобы еще повысить скорость обработки запросов, в версии Oracle9i добавлены битовые индексы соединения таблиц фактов с таблицами измерений. Битовый индекс соединения - это просто битовый индекс, построенный над соединением двух или более таблиц. Повышение производительности объясняется тем, что удается избежать фактического соединения таблиц, или сокращением объема соединяемых данных за счет того, что ограничения уже заранее учтены. Скорость выполнения запросов типа «звезда» с несколькими таблицами измере ний значительно повышается, так как от битовых операций при трансформациях звезды можно вообще отказаться.

Очевидно, что распараллеливание запроса также повышает производительность. В запросах, предъявляемых к системам поддержки принятия решений, операции сортировки и соединения встречаются часто.

Технология Real Application Clusters, заменившая Oracle Parallel Server в версии Oracle9Z, обобщила концепцию параллелизма, разрешив прозрачное выполнение запроса на разных узлах кластера или решетки.

## Сводные таблицы

Данные в таблицах измерений могут быть иерархическими (например, по временному измерению дни сводятся в недели, недели - в месяцы, месяцы “ в кварталы, а кварталы - в год). Если некоторому запросу нужны только помесячные итоги, то зачем просматривать данные за каждый день или неделю? Не лучше ли сразу обратиться к данным на этом или более высоком уровне иерархии? Уже давно специалисты по настройке производительности хранилищ данных придумали для этого сводные таблицы, в которых можно хранить несколько уровней заранее вычисленных итоговых данных. Например, для всех периодов, показанных на рис. 10.2, итоги можно вычислять на лету, задавая подходящую группировку. А чтобы ускорить запросы с участием разных временных рядов, можно заранее вычислить и сохранить в сводных таблицах итоги по неделям и месяцам.

## Материализованные представления

В версии Oracle8i было введено понятие материализованного представления для создания сводных таблиц для фактов и измерений, представляющих различные уровни иерархии итогов. В материализованном представлении хранятся заранее вычисленные итоговые данные. Еще важнее то, что материализованное представление автоматически подставляется вместо гораздо большей детальной таблицы там, где это оправданно. Оптимизатор по стоимости может прозрачно выполнить такую операцию переписывания запроса с подстановкой сводных таблиц, что часто приводит к радикальному повышению производительности. Например, если пользователь запрашивает данные о продажах с разбивкой по месяцам, то оптимизатор автоматически подставит вместо детальной таблицы материализованное представление. При обработке запроса с разбивкой по кварталам оптимизатор может использовать хранящиеся в материализованном представлении итоговые данные по месяцам, отбирая только те месяцы, которые входят в запрошенный квартал. В версии Oracle Database 10g механизм переписывания запросов умеет пользоваться сразу несколькими материализованными представлениями.

Материализованными представлениями позволяет управлять Oracle Enterprise Manager. В состав консультанта SQL Advisor, также доступного из Enterprise Manager, входит SQL Access Advisor, рекомендующий создать материализованное представление, когда это имеет смысл.

# Аналитические исследования, OLAP и добыча данных

Анализ больших массивов данных производится быстрее, если выполняется в той же СУБД, где хранятся данные. В этом разделе описаны встроенные функции СУБД и другие средства аналитических и статистических исследований, оперативной аналитической обработки (online analytical processing, OLAP) многомерных массивов и добычи данных (data mining).

Стоит отметить, что все более широкое применение Oracle для статистических расчетов стимулировало поддержку числовых типов с плавающей точкой, обеспечивающих точность в соответствии со стандартом IEEE 754-1985 (с мелкими отступлениями). Речь идет о типах данных BINARY_FLOAT и BINARY_DOUBLE, появившихся в версии Oracle Database 10g.

## Аналитические и статистические функции

Начиная с версии Oracle8i появляются все новые и новые аналитические и статистические функции. Они входят в редакции Oracle Enterprise Edition и Standard Edition как расширения SQL. В настоящее время имеются следующие функции.

*Функции ранжирования*

Служат для вычисления ранга записи относительно других записей. Вэту группу входят

функции RANK, DENSE_RANK, CUME_ DIST, PERCENT_RANK, NTILE и ROW_NUMBER. Поддерживается также условное ранжирование (hypothetical ranking).

*Функции агрегирования*
Служат для вычисления интегральных и скользящих средних значений. В эту группу входят функции SUM, AVG, MIN, MAX, COUNT, VARLANCE, STDDEV, а также FIRST_VALUE и LAST_VALUE.

*Функции запаздывания/опережения*
Часто применяются для сравнения значений за сходные периоды времени, например, за первый квартал 2006 года и первый квартал 2007 года.

*Функции агрегирования для отчетов*
SUM, AVG, MIN, MAX, COUNT, VARLANCE, STDDEV и RATIO_ TO_REPORT

*Функции линейной регрессии*
Сюда относятся функции REGR_COUNT, REGR_AVGX и REGR_ AVGY, REGR_SLOPE, REGR_INTERCEPT, REGR_R2 и другие, позволяющие построить линию регрессии для заданного набора пар чисел (например, координат X и Y).

Помимо этого, в Oracle поддерживаются операции поворота осей (pivoting), гистограммы (функция WIDTH_BUCKET), выражения CASE, восполнение отсутствующих данных и вычисления с временными рядами.

В состав СУБД включен статистический пакет DBMS_STATS_FUNCS. В него входят функции линейной алгебры, вычисления частых наборов признаков, описательной статистики, проверки гипотез (T-крите- рий, F-критерий, биномиальный критерий, знаково-ранговый критерий Вилкоксона, однофакторный дисперсионный анализ, критерий хи-квадрат, критерии Манна-Уитни и Колмогорова-Смирнова), перекрестной статистики (процентная, хи-квадрат, коэффициент фи, V-ста- тистика Крамера, коэффициент сопряженности и каппа Коэна) и непараметрической корреляции (коэффициенты корреляции Пирсона, Спирмена и Кендалла).

## Предложение MODEL в команде SELECT

Предложение MODEL впервые появилось в версии Oracle Database 10g как расширение синтаксиса SELECT. Оно позволяет трактовать реляционные данные как многомерные массивы (как в электронных таблицах) и используется также для определения формул, применяемых к таким массивам, для устранения множественных соединений и предложений UNION.

MODEL поддерживает аналитические запросы, включающие сравнение с прошлыми периодами и разделение общих предков. Это особенно удобно для формирования бюджета, прогнозирования и других статистических приложений. Например, с помощью MODEL можно вычислить разницу продаж в двух географических пунктах, относительное изменение в процентах и чистую приведенную стоимость. Предложение MODEL также позволяет одновременно использовать в расчетах уравнения и регрессию.

## Средства OLAP и добычи данных

Для хранения в реляционной базе кубов (объектов с предопределенными многомерными соединениями), фактов и измерений в Oracle9i была добавлена опция OLAP Option. Работа с OLAP обычно организуется на уровне SQL, хотя есть и Java API. В версии Oracle Database llg поддерживается переписывание запросов OLAP SQL.

В качестве альтернативы OLAP теперь предлагается технология, не зависящая от реляционной базы данных: Essbase от компании Hyperi- on (которую корпорация Oracle приобрела в 2007 году). Это решение особенно популярно для финансовых приложений Hyperion и в тех случаях, когда бизнес-аналитики хотят самостоятельно генерировать кубы. К кубам Essbase можно также обращаться с помощью инструментов, входящих в продукт Oracle Business Intelligence Enterprise Edition (OBI EE).

Алгоритмы добычи данных впервые были включены в версию Oracle9i как опция Data Mining Option. Первоначально к ним можно было обращаться только с помощью Java API, но позже появилась поддержка и для PL/SQL.

Для добычи данных на уровне приложения без привлечения баз данных в составе продукта OBl EE имеются инструменты для работы с алгоритмами добычи данных, которые корпорация Oracle приобрела вместе с компанией Sigma Dynamics в 2006 году. Эта технология называется Real-Time Decisions (RTD).

В следующих разделах описываются OLAP, добыча данных в базе и инструменты бизнес-анализа, предлагаемые Oracle.

## Расширяемость СУБД и хранилища данных

Втехнологии хранилищ данных набирает популярность тенденция хранить в базе данные

разнородных типов.

### Мультимедийные данные

Набор средств Multimedia (прежнее название mterMedia) позволяет включать в хранилище данных документы, аудио, видео и некоторые пространственные данные. На сегодня самым популярным является средство полнотекстового поиска (Oracle Text). Но количество организаций, хранящих данные других типов, например изображения, тоже растет. Зачастую хранение таких данных обусловлено желанием предоставить пользователям удаленный доступ.

### Опция Spatial Option

Эта возможность также важна для хранилищ, из которых данные выбираются по критерию географической близости к определенным пунктам. В состав пространственных данных входят те или иные географические координаты. Как правило, совместно с Spatial Option используются какие-то дополнительные продукты. Пример применения этого средства в хранилище данных - приложение для маркетингового анализа, определяющее, нужны ли розничные точки продаж в определенных местах.

### XML

Поддержка встроенного типа данных XML появилась в версии Orac- le9i вместе с взаимозаменяемыми механизмами поиска с помощью XML или SQL. Корпорация Oracle предоставила ключевую технологию на этапе разработки стандарта XQuery и начала поставлять промышленный вариант XQuery в составе версии Oracle Database 10g Release 2. В Oracle Database llg производительность работы с XML в базе данных значительно улучшена засчет реализации двоичной формы хранения XML-данных. По оценкам Oracle, производительность двоичного XML по сравнению с XML LOB выросла в 15 раз.

## Управление хранилищем данных

Определив топологию хранилища данных, вы можете развернуть несколько баз данных Oracle для реализации самого хранилища и его витрин. Корпоративные хранилища данных обычно создаются на UNIX- серверах, но все чаще их можно увидеть и на кластерных (RAC) платформах под управлением Linux. Витрины данных меньшего объема нередко размещаются на Windows- и Linux-машинах. Многие организации предпочитают консолидировать витрины данных и корпоративные хранилища на более масштабируемых платформах.

Oracle Enterprise Manager предоставляет общий графический интерфейс для управления всеми экземплярами, не зависящий от операционной системы. EM работает внутри броузера, поддерживая многопользовательский репозиторий для отслеживания и администрирования имеющихся экземпляров Oracle.

При работе с хранилищами данных помимо стандартного администрирования очень важно постоянно следить за производительностью и выполнять настройку. Начиная с версии Oracle Database 10g Enterprise Manager поддерживает многие средства автоматизированной диагностики и настройки.

Для больших хранилищ и витрин данных бывает желательно продолжать обслуживание или обеспечивать доступность некоторых данных даже если другие части базы выведены из оперативного режима. Опция Partitioning Option позволяет определять секции данных исходя из естественных диапазонов значений (например, по дате) или по дискретным значениям. Это повышает гибкость администрирования и заодно скорость выполнения запросов, поскольку оптимизатор умеет исключать из рассмотрения секции, заведомо не содержащие нужные данные. Например, можно определить «скользящее временное окно» для выполнения административных операций по добавлению новых или удалению старых данных. Новую секцию можно добавить, загрузить, параллельно проиндексировать и, возможно, удалить, не прерывая доступ к имеющимся данным.

*Секционирование по диапазону* впервые появилось в версии Oracle8. В версию Oracle8i было добавлено *хеш-секционирование*, позволяющее равномерно распределять данные с помощью алгоритма хеширования. Внутри секций по диапазону могут размещаться хеш-секции (*составное секционирование*), это повышает производительность, не принося в жертву то удобство администрирования, которое дают диапазоны. В версию Oracle9i добавилось *секционирование по списку*: секции создаются на основе дискретных значений, например географических пунктов. Возможность задавать секции по диапазону внутри секций по списку (например, дополнительно разбить региональную секцию по диапазонам дат) появилась в Oracle9i Release2. BOracle Database *llg* были реализованы и другие виды составного секционирования - *спи- сок-хеш*, *список-список*, *список-диапазон* и *диапазон-диапазон*. Тогда же был добавлен механизм *интервального секционирования*, позволяющий автоматически создавать секции по

диапазону, когда это требуется.

## Другое программное обеспечение хранилищ данных

Хранилища данных не обязательно создавать с помощью какого-то одного продукта. Они могут и не быть просто базой данных. Если вы собираетесь построить эффективную топологию хранилища данных, то помимо описанных выше возможностей ПО должно предоставлять следующую функциональность:

*Извлечение данных из оперативных источников*
: Перемещение данных из систем-источников для загрузки хранилища. Процедура может состоять в единовременной массовой выгрузке данных или организации постоянного потока инкрементных изменений.

*Трансформация и/или очистка данных*
: Поскольку данные могут попадать в хранилище из разных источников, их необходимо преобразовывать в единый формат. Кроме того, исходные данные могут нуждаться в очистке, то есть отбрасывании или исправлении неправильных значений.

*Загрузка данных в хранилище или на витрину*
: Этот процесс тоже может быть единовременным или потоковым. *Генерация отчетов*
Бизнес-аналитики, не искушенные в технических деталях, должны иметь простой доступ к стандартным отчетам. Их можно либо запускать из портала, к которому обращаются с помощью броузера, либо просто публиковать.

*Произвольные запросы и анализ*
: Инструменты, которыми бизнес-аналитики могут пользоваться для отбора необходимых данных и самостоятельного построения запросов. Результаты можно публиковать в виде отчетов.

*Развитые средства OLAP для многомерного анализа*
: Для выявления изменений и трендов, особенно при наличии большого числа измерений, необходимы более развитые аналитические средства.

*Добыча данных*
: Обычно применяется при большом количестве переменных, чтобы построить наилучшую модель по известным наблюдениям, а затем воспользоваться ею для прогнозирования новых результатов.

*Управление метаданными*
: Средства для хранения данных, описывающих собственно бизнес и технические детали. Предоставляются также дополнительные службы, например для управления версиями и анализа результатов воздействия.

В следующих разделах описано, как корпорация Oracle обеспечивает всю эту функциональность с помощью различных инструментов и механизмов СУБД.

## Извлечение, трансформация и загрузка

Три первых требования из приведенного выше списка часто реализуются так называемыми инструментами ETL (extraction - извлечение, transformation - трансформация, andloading-загрузка). Те, у кого есть опыт работы с хранилищами данных, понимают, что изучение структуры источников данных, проектирование их трансформации, тестирование процедуры загрузки и отладка - самая длительная часть процесса развертывания. В ходе трансформации обычно исключаются некорректные данные (записи, содержащие ошибки, и дубликаты), затем данные преобразуются к согласованному формату и отфильтровываются те из них, которые не должны попасть в хранилище. При этом не только повышается качество данных, но зачастую удается сократить их общий объем, то есть производительность хранилища улучшается.
Частота извлечения и загрузки определяется, главным образом, требованиями к актуальности данных в хранилище. Чаще всего эти операции производятся в пакетном режиме с заданной периодичностью (обычно один или несколько раз в час или один раз в сутки). Хранилища данных первого поколения, как правило, целиком обновлялись в процессе загрузки. Но по мере роста объема данных загрузка перестает укладываться в отведенные временные рамки. Сегодня чаще применяется обновление таблиц. Если данные в хранилище должны быть максимально близки к реальным, то используется также метод *струйной подачи* (trickle feed), обеспечивающий почти непрерывную загрузку данных.

В СУБД Oracle встроено несколько простых механизмов извлечения и транспортировки данных:

*Прозрачные шлюзы (Transparent Gateways) и гетерогенные службы (Heterogeneous Services )*

Средства для организации моста, позволяющего извлекать данные из сторонних (не-Oracle) источников и загружать их в базу данных Oracle, пользуясь диалектом Oracle SQL. Службы Heterogeneous Services обеспечивают доступ к другим реляционным источникам посредством интерфейса ODBC. Шлюзы в некоторых случаях позволяют повысить скорость извлечения данных из сторонних источников.

*Переносимые табличные пространства (Transportable Tablespaces)*

Еще одно средство перемещения данных, позволяющее быстро переносить данные из одного экземпляра Oracle в другой, не прибегая к экспорту/импорту. Сначала метаданные (словарь данных) экспортируются из исходного экземпляра и импортируются в конечный. После этого перенесенное табличное пространство можно смонтировать на конечном экземпляре. В версии Oracle Database Ю^появил- ся механизм кросс-платформенных переносимых табличных пространств, позволяющий перемещать табличное пространство с одной платформы (например, Solaris) надругую (например, Linux).

*Oracle Streams*

Подсистема Streams появилась в версии Oracle9i Release 2. Она включает репликацию на основе журналов, опцию Advanced Queues (AQ), а начиная с версии Oracle Database 10g также опцию Transportable Tablespaces. Подсистема Streams часто применяется для перемещения данных практически в режиме реального времени. В Oracle Database 10g добавлены поддержка захвата данных из исходного потока (downstream capture), предназначенная для получения изменений данных из журнальных файлов без дополнительных накладных расходов, связанных с использованием RMAN и Transportable Tablespaces, а также поддержка типов данных LONG, LONG RAW, NCLOB и механизм асинхронного сбора изменений, позволяющий перемещать из исходной базы в конечную только изменившиеся записи.

*Быстрый импорт/экспорт с помощью помпы данных*

В версии Oracle Database 10g появился новый формат импорта/экс- порта Data Pump (помпа данных), тесно связанный с поддержкой внешних таблиц. Поддерживаются параллельная загрузка и выгрузка в прямом режиме.

Все эти средства СУБД обычно применяются для высокопроизводительной передачи данных и сами по себе не годятся для выполнения сложных трансформаций. Корпорация Oracle предлагает также инструмент извлечения, трансформации и загрузки Oracle Warehouse Builder (OWB), позволяющий построить отображение источников на конечные таблицы в терминах предопределенных или определенных пользователем трансформаций. Затем OWB может автоматически сгенерировать сценарии, необходимые для выполнения ETL. На самом деле, инструмент OWB - это не просто средство для определения ETL; он также может выступать в роли инструмента проектирования хранилищ данных и служить репозиторием метаданных. Поддерживается также импорт файлов, созданных в других системах проектирования, например Oracle Designer, CA's ERwin, Sybase PowerDesigner и Business Objects Designer.

При построении большинства хранилищ данных сначала импортируются метаданные, описывающие исходные таблицы, - из базы данных Oracle (посредством связей баз данных), других РСУБД (с помощью ODBC или шлюзов) или плоских файлов. Затем проектируются с нуля или импортируются конечные таблицы, после чего определяется отображение исходных метаданных на конечные (возможно, с применением трансформации). В состав базового набора трансформаций, поддерживаемых OWB, входит оператор очистки имен и адресов, предназначенный для использования совместно с библиотеками партнеров Oracle и в приложениях для анализа рынка, сопоставления и объединения данных. В опции OWB Enterprise Option имеются также дополнительные функции, например поддержка медленно изменяющихся измерений и подключаемых отображений. Опция OWB Data Quality Option поддерживает профилирование данных и правила данных (data rules).

OWB умеет проверять корректность отображения между исходным и конечным представлением

После того как корректность отображения установлена, можно сгенерировать следующие элементы:

- DDL-команды, если требуется создать конечные таблицы;
- управляющие файлы SQL*Loader для загрузки данных из плоских файлов;
- PL/SQL-сценарии для извлечения, трансформации и загрузки из реляционных источников.

Сценарии развертываются и запускаются на стороне конечного хранилища данных, обычно по расписанию планировщика заданий, встроенного в Enterprise Manager. Таким образом, OWB - в большей степени инструмент ETL, поскольку для выполнения трансформаций задействуется конечная СУБД. Если требуется более сложный алгоритм планирования - с проверкой тех или иных условий, то OWB может использовать компоненты Oracle Workflow.

OWB предоставляет доступ к целому ряду сторонних источников. Для систем E-Business Suite и

PeopleSoft есть коннекторы, позволяющие включать созданные в этих ERP-приложениях объекты в отображения и обрабатывать потоки работ. Коннектор SAP Connector аналогичен, но включает также кодогенератор ABAP, позволяющий организовать доступ к любой таблице SAP над любой базой данных, в том числе к кластерным таблицам, с помощью RFC-соединения.

Для высокоскоростной загрузки плоских файлов компонент Oracle SQL*Loader поддерживает *загрузку в прямом режиме* (direct path lo- ad), когда выполняется запись прямо в файл данных, в обход кэша буферов и механизма отката. Если нужно дополнительно ускорить процесс загрузки таблиц (поскольку за ограниченный промежуток времени нужно загрузить данные в несколько хранилищ), то можно запустить несколько сеансов SQL*Loader параллельно. Многие популярные инструменты извлечения данных, в том числе OWB, генерируют сценарии для SQL*Loader.

В версии Oracle9i основная функциональность ETL впервые была включена в ядро СУБД. Это поддержка внешних таблиц, табличные функции, слияние данных (то есть вставка или обновление в зависимости от того, имеется ли уже рассматриваемый элемент данных), многотабличная вставка, технология Change Data Capture и возобновляемые команды. Внастоящее время вся эта функциональность доступна в OWB. Кроме того, OWB может создавать струйные каналы с помощью подсистем Streams и Advanced Queues.

Для выполнения ETL в базы данных Oracle и других производителей предлагается продукт Oracle Data Integrator (ODI). Он был приобретен в 2007 году, а ранее назывался Sunopsis. ODI основан на *модулях знаний* (Knowledge Modules), определяющих возможности интеграции, в том числе захват измененных данных, утилиты загрузки и выгрузки, средства загрузки и выгрузки с помощью SQL-команд и SQL-команды, описывающие логику трансформации. Модули знаний можно модифицировать.,В состав продукта входит среда разработки, позволяющая использовать модули знаний как шаблоны для процесса декларативного проектирования и в качестве агента оркестровки (orchestration agent).

Помимо организации гетерогенного механизма ETL ODI можно применять для развертывания и интеграции данных и служб трансформации в инфраструктуре SOA (Service-Oriented Architecture). ODI является ключевым компонентом решений на основе Oracle MDM и некоторых вновь разрабатываемых приложений для бизнес-анализа.

## Инструменты для генерации отчетов и произвольных запросов

Аналитики, занимающиеся маркетинговыми, финансовыми и иными проблемами, редко интересуются схемой, описывающей информацию, и методами ее хранения. Заинтересованность они начинают проявлять, когда речь заходит о предлагаемом инструментарии. Решение о покупке инструментов для бизнес-анализа часто принимается в интересах конкретных бизнес-отделов, иногда даже без консультаций с ИТ-отде- лом. Для систем на основе СУБД Oracle на выбор есть инструменты бизнес-анализа, предлагаемые самой корпорацией Oracle и независимыми компаниями, например Business Objects, Cognos и MicroStrategy.

Сегодня корпорация Oracle предлагает инструменты бизнес-анализа в трех вариантах комплектации - Oracle Business Intelligence Enterprise Edition, Standard Edition One и Standard Edition. Кроме того, купив компанию Hyperion, Oracle получила разработанные ею серверные и клиентские продукты, которые включены в редакцию Oracle Business Intelligence Enterprise Edition Plus. Стратегически важными корпорация Oracle считает редакции Enterprise Edition Plus и Standard Edition One, но все остальные продукты также продолжают разрабатываться и поддерживаться.

В редакцию Oracle Business Intelligence Enterprise Edition (OBI EE) входят инструменты, разработанные бывшей компанией Siebel Analytics, а также Oracle BI Publisher (прежнее название XML Publisher). Включены оптимизированные версии для СУБД производства Oracle и других компаний. В комплект входят:

*Интерактивные инструментальные панели*
: Интерактивное представление в броузере разнообразной информации, полученной от других компонентов OBI EE, например Answers. В частности, это могут быть аннотированные аналитические данные, предлагающие неискушенным бизнес-пользователям рекомендации по поиску дополнительной информации.

*Answers*
: Тонкий интерактивный клиент (на основе DHTML) для генерации произвольных запросов и анализа результатов. Компонент Answers может напрямую обращаться к реляционным базам данных или хранилищам данных в формате MOLAP. Сгенерированные отчеты можно выводить на инструментальную панель или подавать на вход компонента BI Publisher.

*Отчеты и публикации (BI Publisher)*

Этот компонент обеспечивает публикацию по шаблону. Данные извлекаются в формате XML, а отчеты могут быть представлены вразличных форматах, включая PDF, RTF, HTML, Excel, XML и eText. Редактировать отчеты можно в таких популярных программах, как Adobe Acrobat и Microsoft Word.

*Delivers*

Инфраструктура на основе сообщений «iBot», которые отсылаются при возникновении определяемых пользователем условий. Компонент Delivers можно настроить для рассылки сообщений по электронной почте в виде уведомлений на инструментальной панели, в виде SMS-сообщений или с применением иных механизмов оповещения. Этот компонент можно также связать с описанием бизнес- процесса на языке Business Process Execution Language (BPEL).

*DisconnectedAnalytics (Анализ без соединения)*

Позволяет бизнес-аналитику работать с комплектом инструментов автономно, обращаясь только к данным, хранящимся на локальном диске. После восстановления соединения с сетью производится синхронизация данных.

*Подключаемый модуль для Office*

Обеспечивает доступ к серверу BI Server из популярных продуктов корпорации Microsoft, например Excel.

*BI Server*

ПО промежуточного уровня для ранее описанных компонентов. Предоставляет бизнес-модель, слой извлечения данных, службы кэширования, движок вычислений и интеграции, а также оптимизированный доступ к данным в поддерживаемых источниках. К ним относятся СУБД Oracle и опция Oracle OLAP Option (аналитические рабочие области), Microsoft SQL Server и Analysis Services, IBM DB2, Teradata и иные источники, поддерживающие интерфейс ODBC. Есть и другие источники, например: Oracle Business Intelligence (BI) Applications, PeopleSoft EPM, E-Business Suite, Siebel, Fusion Business Intelligence Applications и SAP.

*BI ServerAdministrator*

Служит для управления уровнем представления, бизнес-моделью и отображением, а также физическим уровнем, определенным в компоненте BI Server. С помощью этого инструмента конфигурируются права доступа для отдельных пользователей и бизнес-аналитиков или их групп.

В редакцию OBI EE Plus включены также компоненты компании Hy- perion: Hyperion Foundation Services, Interactive Reporting, SQR production reporting, Financial Reporting, Smartview for Office и Web Analysis.

## OLAP и построение OLAP-приложений

Более квалифицированного бизнес-пользователя может интересовать не «что произошло», а «каковы тенденции и что может произойти в будущем». *Инструменты OLAP* могут выполнять математический анализ временных рядов для выявления прошлых трендов и прогнозирования будущих.

Изначально технология OLAP стала ответом на неспособность реляционных баз данных эффективно обрабатывать многомерные запросы (см. вышераздел «Проектирование хранилищаданных»). Поэтомуин- струменты OLAP поставлялись в комплекте с собственными «кубами» данных, в которые данные загружались из реляционных источников.

Эти совершенно отдельные СУБД получили название Multidimensional Online Analytical Processing, или *MOLAP* (система многомерной оперативной аналитической обработки). В качестве примеров можно привести Oracle Express Server и Oracle Hyperion Essbase, а также Microsoft Analysis Services. Такие системы способны очень быстро обрабатывать запросы, но оптимально работают при условии, что данные обновляются редко (поскольку генерация куба занимает время). Продукт Oracle Essbase содержит движок MOLAP, который можно использовать в сочетании с различными реляционными СУБД.

Теперь функциональность OLAP все чаще включается в реляционные СУБД, поскольку во многих системах в той или иной степени поддерживается схема типа «звезда» с различными уровнями агрегирования, а также из-за насущной потребности в обработке часто изменяющихся данных. В таком варианте технология называется *ROLAP* (Relational Online Analytical Processing - реляционный OLAP). Инструменты, способные работать как с реляционными базами данных, так и с MOLAP-системами, называются *гибридными*. При развертывании ROLAP-системы инструменты Oracle Business Intelligence и некоторые другие могут пользоваться определенными в стандарте ANSI аналитическими функциями, которые встроены в СУБД как расширения SQL, а также обращаться к опции OLAP Option, реализующей MO-

LAP-куб внутри реляционной базы данных, на языке SQL.
В версии Oracle Database llg гибкость доступа к опции OLAP Option была существенно повышена. На языке SQL запросы можно было формулировать и раньше, но при этом бизнес-пользователи должны были адресовать запросы именно кубам OLAP Option. При развертывании в рамках версии Oracle Database llg OLAP-кубы можно прозрачно использовать в качестве альтернативы материализованным представлениям, поскольку механизм перезаписи запросов в Oracle SQL распознает кубы. Начиная с версии Oracle Database П^обновление материализованного представления может вызывать и обновление OLAP-куба.
Кубы OLAP Option развертываются в так называемых *аналитических рабочих областях*. Их можно создать с помощью инструмента Oracle Warehouse Builder или более простого инструмента логического многомерного моделирования Analytic Workspace Manager (AWM). Тот и другой предлагают интерфейс для создания кубов и отображения реляционных таблиц на кубы.
Программа Oracle JDeveloper и технология *business intelligence beans* позволяют создавать собственные OLAP-приложения, хотя готовые продукты применяются гораздо чаще. В виде JavaBeans поставляются готовые компоненты для манипулирования таблицами, перекрестными таблицами и графами, а также для конструирования запросов и выполнения вычислений аналогично тому, как это раньше делалось в Express. JDeveloper генерирует на основе этих кирпичиков Java-код, включающий обращения к классам из библиотеки Java OLAP API, поставляемой в составе OLAP Option.

## Добыча данных

Термин *добыча данных* (data mining) употребляется в контексте хранилищ данных слишком часто и не всегда корректно. Он обозначает применение математических алгоритмов для моделирования таких взаимосвязей между данными, которые было бы трудно обнаружить иными способами. Мы не рекомендуем заниматься добычей данных, если в компании нет аналитиков, отвечающих следующим критериям:

- четкое понимание качества и семантики данных, присутствующих в хранилище;
- проникновение в суть бизнеса, достигнутое путем применения других инструментов;
- осознание того, что некоторый аспект бизнеса зависит от слишком многих переменных, поэтому промоделировать результаты воздействий каким-либо иным способом не представляется возможным.

Другими словами, инструменты добычи данных не могут служить заменой аналитическим способностям пользователей хранилища данных.
Для вскрытия взаимосвязей в инструментах добычи данных применяются различные методы, такие как:

- Обобщенные статистические алгоритмы, способные выявить статистические вариации.
- Методы кластеризации, показывающие, как результаты бизнеса можно объединить в группы, например при изучении зависимости количества страховых требований от времени для различных возрастных категорий. Вданном случае, отыскав группу с низким риском, можно продолжить изучение влияния других факторов, или «ассоциаций».
- Проверка логических моделей (если имеет место А, то возможен результат В или С) на небольших выборках с последующим применением к крупным наборам данных для прогнозирования. Обычно эту методику называют *деревьями решений*.
- Обучение нейронных сетей на небольших наборах данных с последующим применением полученных результатов к гораздо более объемным наборам.
- Алгоритмы обнаружения аномалий для выявления выбросов и редких событий.
- Применение методов визуализации для графического отображения переменных с целью понять, какие из них в первую очередь влияют на результаты.

Часто методы добычи данных применяются для решения трудных задач бизнеса, например обнаружения мошенничества и небольших колебаний в конъюнктуре рынка, а также в других областях, где результат зависит от многих переменных. Компании по обслуживанию кредитных карт применяют добычу данных для выявления необычных случаев употребления, например оплаты дорогостоящих ювелирных украшений в городе, где владелец карты ранее не бывал. Выявление кластеров нестандартного потребительского поведения в небольших группах людей может привести к решению провести маркетинговую микрокампанию, нацеленную на ограниченную аудиторию с высокой вероятностью покупки продуктов или услуг.
В последнее время среди поставщиков реляционных СУБД наблюдается отчетливая тенденция интегрировать алгоритмы добычи данных в свои продукты. Первоначально корпорация Oracle предлагала в качестве решения по добыче данных клиент-серверный продукт Oracle
Darwin, включающий алгоритмы моделирования ассоциаций, нейронные сети, деревья классификации и регрессии, а также кластеризацию данных, хранящихся в таблицах БД Oracle или в плоских файлах. В версии Oracle9i эти алгоритмы стали поставляться в виде опции Data

Mining Option. В настоящее время сюда входят следующие алгоритмы: наивная байесовская фильтрация, адаптивные байесовские сети, кластеризация, машины опорных векторов (SVM), факторизация неотрицательной матрицы (NMF), деревья решений и обобщенные линейные модели (версия Oracle Database 11 *g* поддерживает бинарную логическую регрессию и многомерную линейную регрессию). Имеются API для доступа к алгоритмам из языков Java и PL/SQL. Из других средств добычи данных упомянем классификацию и кластеризацию текстовых документов (text mining). а также поиск аналогов с помощью инструмента BLAST, основанного на алгоритмах SVM (применяется в генетических исследованиях).

Инструмент Oracle Data Miner позволяет создавать собственные приложения для добычи данных. Он предназначен для разработки, тестирования и оценивания моделей. В общем случае данные нужно сначала подготовить для добычи, то есть выполнить сортировку (binning), нормализацию и восполнить недостающие значения. В опции Data Mining Option для Oracle Database llg процедура подготовки данных автоматизирована. Data Miner позволяет определять метаданные, вручную оптимизировать сгенерированный Java-код, просматривать сгенерированные XML-файлы и тестировать компоненты приложения. Кроме того, к услугам аналитиков такие инструменты, как InforSense или SPSS Clementine, которые пользуются алгоритмами, реализованными в Data Mining Option, и предоставляют полную среду разработки.

## Приложения для бизнес-анализа

Приложения для бизнес-анализа - это набор готовых решений для генерации сложных отчетов и конструирования интерфейсов типа «инструментальная панель», позволяющих отображать тренды в развитии бизнеса. Эти приложения либо напрямую обращаются к OLTP-схеме (Oracle Daily Business Intelligence), либо - что более распространено - к решениям с инфраструктурой, напоминающей традиционные хранилища данных. В качестве примеров второго подхода можно отметить Oracle Business Intelligence Applications, PeopleSoft EPM и SAP Business Warehouse - все они часто развертываются поверх СУБД Oracle.

Приложения для бизнес-анализа обычно ориентированы на конкретные стороны бизнеса, например маркетинговый или финансовый анализ. Так, приложения, входящие в состав Oracle Hyperion Financial Performance Management, относятся к финансовому планированию и формированию бюджета. В таких приложениях имеются предопределенные запросы, отчеты и диаграммы, позволяющие получить относящуюся к рассматриваемому вопросу информацию, что избавляет пользователя от необходимости создавать эти объекты с нуля. В решения на основе хранилищ данных также включаются готовые процедуры извлечения, трансформации и загрузки данных из поддерживаемых источников.

Входящий в Oracle E-Business Suite продукт Daily Business Intelligence (DBI) предоставляет доступ к таблицам транзакций и материализованным представлениям Oracle. Доступ осуществляется с помощью предопределенных рабочих книг (workbook) Oracle Business Intelligence, уже заполненных метаданными о бизнесе. Последний поддерживаемый комплект инструментов - OBI EE. Oracle DBI включает модули: Compliance (Соответствие требованиям), Customer Support (Поддержка клиентов), Financials (Финансы), Human Resources (Персонал), Procurement (Закупки), Product Lifecycle (Жизненный цикл продукта), Projects (Проекты), Marketing (Маркетинг), Maintenance (Обслуживание), Sales (Продажи), Supply Chain (Цепочки поставок) и другие.

В Oracle Business Intelligence Applications в редакции OBI EE входит свыше 2500 ключевых показателей эффективности {КПЭ) и предопределенные отображения для извлечения, трансформации и загрузки данных из приложений Siebel, Oracle E-Business Suite, PeopleSoft, SAP и других. Ранее известные под названием Siebel Analytics, эти приложения охватывают следующие стороны бизнеса: Sales (Продажи), Service and Contact Center (Центр поддержки и обслуживания), Marketing (Маркетинг), Financials (Финансы), Supply Chain (Цепочки поставок) и Workforce (Трудовые ресурсы). Oracle Business Intelligence Applications позиционируется как флагманский продукт в области горизонтальных приложений для бизнес-анализа. Поэтому корпорация Oracle продолжает расширять набор предлагаемых КПЭ, отображений для ETL, охватывая все новые стороны бизнеса.

В состав продукта PeopleSoft EPM входит более 1200 метрик с готовыми отображениями для приложений PeopleSoft и JD Edwards. В комплекте с EPM поставляются хранилища данных для управления человеческими ресурсами (Human Capital Management), финансами (Financials), организации кампусов (Campus Solutions), управления цепочками поставок (Supply Chain) и взаимоотношениями с клиентами (Customer Relationship Management). Эти приложения также поддерживают использование OBI EE в качестве фронтального инструмента.

Готовые решения обеспечивают простоту развертывания вкупе с уже заложенной развитой функциональностью. Хотя без настройки все равно не обойтись, время на развертывание

начального полезного решения удается заметно сократить.

## Проблема метаданных

С одной стороны, роль *метаданных*, то есть данных, описывающих другие данные, невозможно переоценить. Практически для всех взаимодействий с базой данных необходимы метаданные - от типов данных до бизнес-семантики и истории полей данных.

С другой стороны, метаданные полезны лишь в том случае, когда их могут использовать заинтересованные клиенты и инструменты. Одна из самых сложных проблем состоит в том, чтобы построить единый набор определений метаданных, который позволил бы взаимодействовать инструментам и СУБД разных производителей.

Было предпринято несколько попыток достичь соглашения об общих определениях метаданных. В 2000 году ратифицирован стандарт, описывающий единый интерфейс для обмена реализациями метаданных. Этот стандарт, разработанный группой Object Management Group (OMG), называется Common Warehouse Metadata Interchange (CWMI) и основан на обмене данными в формате XML. Корпорация Oracle - один из первых сторонников и разработчиков технологии, описанной в этом стандарте. Например, имеется CWM-мост для обмена метаданными, хранящимися в репозитории Oracle Warehouse Builder. OWB также включает программу просмотра, позволяющую получить более детальную информацию о метаданных, и программу для просмотра данных аудита и построения диаграмм анализа последствий.

Выше отмечалась тенденция к развитию дополняющего решения, в котором ETL для одного хранилища данных не является окончательной целью. Речь идет об управлении эталонными данными и концентраторах данных. На сегодня большинство организаций все еще далеки от определения консолидированных метаданных, а попытки достичь цели путем внедрения передового опыта ИТ-отделом обычно оказывались безуспешными. Такие проекты принимаются лишь будучи включенными в бизнес-аналитический проект, обещающий практическую ценность для бизнеса.

## Передовой опыт

Те, у кого есть солидный опыт бизнес-анализа, скорее всего, согласятся, что вышеупомянутые попытки внедрения передового опыта проваливаются по следующим типичным причинам:

*Неспособность привлечь бизнес-пользователей, представителей ИТ- отдела, бюджетоформирующие органы и прочие заинтересованные стороны к участию в проекте на всем его протяжении*

Во-первых, от этих групп можно получить информацию, ценную для создания бизнес-аналитического решения, а во-вторых - отсутствие сотрудничества с любой из них может стать причиной провала проекта.

*Игнорирование важнейших причин для создания инфраструктуры бизнес-анализа*

На этапе планирования ИТ-архитекторы можно упустить из виду мотивы, движущие созданием решения.

*Незамеченные детали и неверные предположения*

Недостаточно пристальное исследование окружения может обречь весь проект на неудачу.

*Нереальные сроки и широта охвата*

Как и в любом другом проекте, слишком малый срок разработки и излишне широкий охват бизнес-аналитического решения заставляют разработчиков «срезать углы», что приводит к упомянутым выше ошибкам.

*Пренебрежение разъяснением ожидаемых результатов*

Как и любая технология, хранилища данных и бизнес-анализ не являются панацеей. Необходимо четко разъяснить всем членам команды и потенциальным конечным пользователям, чего можно ожидать от разрабатываемого решения.

*Принятие тактических решений вместо определения долгосрочной стратегии*

Хотя в самом начале эта задача может показаться слишком трудоемкой, тем не менее вы должны помнить о долгосрочных целях проекта и организации в целом на протяжении всего процесса проектирования и внедрения. Пренебрежение этим аспектом влечет двоякие последствия: относит на будущее выявление проблем, увеличивая при этом вероятность их возникновения и серьезность.

*Игнорирование чужого опыта*

Нет лучшей школы, чем изучение опыта успешных реализаций похожих проектов. Не менее полезно учиться на чужих ошибках; по крайней мере, вы сможете избежать действий, приведших к неудаче.

Чтобы проект по внедрению средств бизнес-анализа оказался успешным, необходимо постоянно сотрудничать с бизнес-аналитиками и пользователями, бюджетоформирующими органами и

ИТ-отделом. Игнорирование этой рекомендации - пожалуй, основная причина наиболее впечатляющих провалов. Организация подобной инфраструктуры может принести очевидную пользу бизнесу и заметный возврат на инвестиции (return on investment, ROI). Участие руководителей высшего звена во всем процессе- важнейший фактор, поскольку различные функции бизнес-анализа часто пересекают границы отделов, поэтому финансирование должно спускаться с верхнего уровня.

Ваш проект бизнес-анализа должен отвечать на вопросы, которые возникают в ходе проработки ключевых инициатив, связанных с бизнесом.

Безжалостно пресекайте все разработки, уводящие проект в сторону. В основе графика реализации должно быть желание получить ответы на критические для бизнеса вопросы. Постепенная реализация проекта должна показывать положительный возврат на инвестиции.

## Типичные заблуждения

Слишком упрощенное (недостаточно детализированное) представление о какой-либо части процесса может порождать многочисленные проблемы. Вот лишь несколько типичных (и обычно неверных) допущений, принимаемых в процессе разработки бизнес-аналитического решения.

- Источники данных очищены и непротиворечивы.
- Кто-то в организации понимает, что находится в исходных базах данных, и может ответить на вопросы о качестве данных и о том, где найти данные, представляющие интерес для бизнеса.
- Данные из оперативных источников можно извлечь и при необходимости отбросить, не оставляя никаких побочных записей.
- Сводных данных будет достаточно, и потому детальные данные можно не включать.
- В ИТ-отделе есть специалисты, способные разработать все необходимые процедуры извлечения, настроить базы данных, обслуживать системы и сеть и выполнять резервное копирование и восстановление в разумные сроки.
- Разработку можно вести без постоянной обратной связи и периодической демонстрации прототипа аналитикам и, возможно, руководителям, выделяющим бюджет.
- Хранилище данных не будет изменяться со временем, поэтому вопрос об управлении версиями не ставится.
- Квалификация аналитиков достаточна для полноценного использования инфраструктуры или инструментов бизнес-анализа.
- ИТ-отдел может контролировать выбор инструментов, которыми пользуются аналитики.
- Количество пользователей известно и предсказуемо.
- Виды запросов известны и предсказуемы.
- Аппаратное обеспечение бесконечно масштабируемо вне зависимости от принятых решений.
- Если какой-то отдел независимо создаст витрину данных или развернет некое приложение, то поддержка ИТ-отдела для него не потребуется.
- В случае крайней необходимости для разрешения оставшихся проблем консультант явится по первому зову.
- Метаданные и эталонные данные несущественны, к ним можно вернуться позднее.

## Эффективная стратегия

При разработке и внедрении большинства программных проектов не удается уложиться в график. Так как решения для бизнес-анализа очень сложны, зачастую на их разработку уходит гораздо больше времени, чем планировалось первоначально, а это как раз то, что категорически не хотят слышать руководители, которым информация необходима для принятия стратегических решений! Если вы ведете разработку постепенно, попутно создавая работоспособные прототипы, то проект может начать приносить ROI на ранних стадиях и корректировку графика можно будет оправдать изменением требований со стороны бизнеса, а не просто техническими сложностями (в которых топ-менеджеры обычно не разбираются).

Следует избегать чрезмерного разрастания функциональности проекта и возлагаемых на него надежд. Если со стороны бизнеса поступают предложения о каких-то изменениях или добавлениях, то следует удостовериться в том, что они принесут адекватный ROI. В противном случае вы будете долго и упорно работать над теми аспектами инфраструктуры, которые реально не окупятся. Аргументация представителей бизнеса должна быть составной частью процедуры выработки приоритетов; вы должны понимать, почему идете на те или иные компромиссы. Будучи вовлечены в «борьбу за влияние» между отделами по поводу того, кто является владельцем данных, привлекайте к арбитражу топ-менеджеров.

Нехватка времени и квалифицированных специалистов в сочетании с давлением со стороны бизнеса иногда заставляет принимать тактические решения об организации хранилища данных в ущерб долгосрочной стратегии. Вопреки оказываемому давлению вы обязаны в самом

начале проекта выработать долгосрочную стратегию и придерживаться ее или, по крайней мере, понимать, к каким последствиям приведет ее изменение. Следует учесть как можно больше деталей, чтобы не тратить зря время по ходу реализации. При этом стратегия должна быть достаточно гибкой и учитывать возможные слияния, приобретения и так далее.

Долгосрочная стратегия должна принимать во внимание новые тенденции, например, необходимость доказательства соответствия или потребность в решениях с высокой доступностью. Скорость изменений и объем появляющихся на рынке продуктов часто мешают отделить зерна от плевел. Большинство компаний стремятся идти в ногу с прогрессом. Традиционно источниками информации служат сами производители, консультанты и аналитики, специализирующиеся на индустрии обработки данных; при этом каждый из них стремится что-то продать. Поставщики мечтают продать свои продукты; консультанты - свои знания, а аналитики - свои благожелательные обзоры поставщиков и консультантов все тем же поставщикам и консультантам. Положившись на один-единственный источник информации, легко прийти к ложным выводам. Напротив, изучая информацию из нескольких источников, можно прийти к некоему консенсусу и получить ответы на свои вопросы.

Лучший способ разобраться в реальной ситуации - обсудить бизнес- аналитические проекты с представителями похожих компаний на конференции - хотя бы на стадии работоспособного прототипа. Отыскание работающих решений и установление новых контактов могут оправдать затраты на посещение конференции и оказаться ценнее докладов, прочитанных в рамках программы мероприятия.

# Лекция 11. Oracle и высокая доступность.

## Что такое высокая доступность?

Прежде чем обсуждать обеспечение высокого уровня доступности данных, уточним смысл термина *доступность* (availability).

Для разных организаций доступность может означать различные вещи. Ниже мы будем считать, что система доступна, когда она *запущена* (то есть пользователи могут обращаться к базе данных) и *работает* (то есть база данных предоставляет ожидаемую пользователями функциональность с ожидаемой производительностью).

Для нормальной работы большинства организаций доступность данных - необходимое условие. В последнее время, когда доступ к данным стал предоставляться через Интернет, любые сбои в работе СУБД могут оказаться фатальными для бизнеса. Отказ системы, к которой обращаются пользователи вне организации-владельца, немедленно становится виден широкой аудитории, и это может неблагоприятно отразиться на финансовом благополучии и имидже компании. Возьмем, к примеру, веб-сайт службы поддержки клиентов компании, занимающейся доставкой почтовых бандеролей. С помощью этого сайта клиент может отслеживать, на какой стадии обработки находится его бандероль. Поскольку клиенты зависят от бесперебойного функционирования этой службы, любой сбой в ее работе может побудить их обратиться к конкурентам.

Можно пойти дальше и подумать о том, с какими сложностями придется столкнуться, если данные находятся в нескольких системах. Интеграция различных систем повышает вероятность того, что отказ в одном звене приведет к недоступности всей цепочки поставки.

Для реализации высокодоступных баз данных следует проектировать инфраструктуру так, чтобы она компенсировала возможные сбои, например путем установки избыточного оборудования. Кроме того, необходимы методы, позволяющие восстановить состояние базы после сбоя, например надлежащим образом организованное резервное копирование.

## Измерение и планирование доступности

В большинстве организаций изначально предполагается, что данные должны быть доступны 24 часа в сутки 7 дней в неделю (доступность 24/7). Довольно часто это требование выдвигается без детального анализа функций, которые система призвана поддерживать. Поскольку стоимость аппаратных компонентов все время снижается, а их надежность растет, многие начинают думать, что достижение высокого уровня доступности - дело простое и дешевое.

К сожалению, доступность одного компонента - пусть даже недорогого и надежного - еще не означает доступность системы в целом. Сложность развертывания аппаратных и программных компонентов в современных двух- и трехуровневых системах порождает многочисленные зависимости и точки отказа. Достижение высокого уровня доступности системы, состоящей из разнообразных взаимозависимых компонентов, как правило, оказывается отнюдь не простой и не дешевой задачей.

Чтобы вам было проще сориентироваться, в табл. 11.1 мы показали, как различные уровни доступности транслируются во время простоя, выраженное в днях, часах и минутах, за год (365

дней).

*Таблица ПЛ.Доступностъ системы*

| Доступность, % | Время простоя системы в течение года | | |
|---|---|---|---|
| | Дни | Часы | Минуты |
| 95,000 | 18 | 6 | 0 |
| 96,000 | 14 | 14 | 24 |
| 97,000 | 10 | 23 | 48 |
| 98,000 | 7 | 7 | 12 |
| 99,000 | 3 | 16 | 36 |
| 99,500 | 1 | 20 | 48 |
| 99,900 | 0 | 9 | 46 |
| 99,990 | 0 | 1 | 53 |
| 99,999 | 0 | 0 | 5 |

Проектирование и внедрение крупномасштабных систем с уровнем доступности 99% и выше может обходиться в миллионы долларов, расходы на их эксплуатацию столь же высоки. Небольшое повышение доступности может потребовать солидных затрат на приобретение компонентов систем. Повышение доступности от 95 до 99% будет стоить дорого, но переход от 99 к 99,99%, скорее всего, обойдется еще дороже.

Еще один важный аспект измерения доступности - определение того, *когда* система должна быть доступна. Уровень доступности в 99% рабочего времени (например, с 8 утра до 5 вечера) - совсем не то же самое, что доступность 99% в течение всех 24 часов. Поэтому следует точно определять не только требуемые уровни доступности, но и периоды времени для каждого уровня. Например, многие компании принимают заказы только в «рабочее время». Цена недоступности системы ввода заказов в эти часы очень высока, но в остальное время куда меньше. Таким образом, имеет смысл планировать регламентное обслуживание в нерабочее время, а это, в свою очередь, поможет свести к минимуму незапланированные сбои в часы наибольшей активности. Разумеется, для транснациональных корпораций и компаний, имеющих представительство в Интернете, глобальное присутствие означает неограниченный «рабочий день».

Исходное требование о доступности 24/7 должно рассматриваться сучетом затрат на развертывание и обслуживание такой системы.
Детальный анализ сложности и стоимости систем с очень высокой доступностью иногда вынуждает идти на компромиссы, снижающие уровень требований и бюджет.

Но в некоторых случаях затраты на обеспечение высокой доступности безусловно оправданны. Какой-нибудь брокерской конторе каждый час простоя может обходиться в миллионы долларов. А, скажем, компания, торгующая товарами по каталогу, может потерять в час всего несколько тысяч долларов, причем на случай сбоя можно предусмотреть менее эффективную ручную процедуру. Но даже безотносительно к стоимости упущенной выгоды неожиданный отказ системы может снизить продуктивность работников и персонала ИТ-отдела.

## Причины незапланированного простоя

Незапланированные простои могут возникать по разным причинам. Некоторые предотвратить легко, тогда как для устранения других требуются значительные капиталовложения в инфраструктуру вычислительного центра, телекоммуникации, программное и аппаратное обеспечение, кадры.
Планируя меры по обеспечению доступности приложения, вы должны учитывать все перечисленные на этой диаграмме факторы, а равно идругие потенциальные причины прерывания работы системы, характерные для конкретного окружения. При любом планировании лучше принять во внимание все возможности (даже если некоторые из них потом будут отброшены), чем оказаться застигнутым врасплох неожиданным событием.

## Доступность системы и доступность компонентов

Система в целом состоит из аппаратных, программных и сетевых компонентов, организованных в виде технологического *стека*. Высокая доступность отдельных компонентов еще не гарантирует доступность всей системы. Есть разные стратегии и решения, направленные на обеспечение высокой доступности каждого из компонентов системы. На рис. 11.2 представлен технологический стек гипотетической системы.
Как видно из этого рисунка, для обеспечения работы приложения необходима кооперация различных физических и логических уровней. В некоторых системах компонентов может быть и

меньше; так, для двухуровневой системы типа клиент/сервер не нужны компоненты, относящиеся к серверу приложений.


***Рис. 11.2. Компоненты системы***

Отказ компонентов, расположенных выше уровня базы данных, может привести к потере доступа к базе, хотя сама база остается доступной. Сервер базы данных и сама база находятся в основании стека. Отказ базы данных тут же распространяется на верхние уровни стека. Если отказ приводит к потере или порче данных, то может оказаться под угрозой целостность самого приложения.

## Сбой системы

Внезапный отказ сервера, на котором работает база данных, - одна из наиболее распространенных причин незапланированного простоя. Выход сервера из строя может быть обусловлен аппаратной ошибкой, например сбоем источника питания, или программными проблемами, скажем, из-за того, что некоторый процесс начал потреблять все ресурсы процессора. Даже если операционная система работает нормально, может произойти сбой самого экземпляра Oracle. Какова бы ни была причина сбоя, видимый результат один и тот же - экземпляр Oracle перестает обеспечивать требуемую функциональность. Напомним, что когда мы говорим о сбое базы данных Oracle, имеется в виду обслуживающий ее экземпляр, а не сама база данных. Даже если и произойдет сбой системы, он не затронет данные, которые уже сохранены в дисковых файлах, составляющих базу данных Oracle.

Насколько далеко распространятся последствия сбоя, зависит от того, какие операции в этот момент выполнялись. Серверного процесса, обслуживавшего подключенные сеансы, уже нет. Активные запросы и транзакции аварийно завершаются. Процесс устранения возникшего хаоса называется *восстановлением экземпляра*, или *восстановлением после сбоя*.

## Что такое восстановление экземпляра?

При перезапуске экземпляра после сбоя Oracle обнаруживает, что имел место сбой, анализируя информацию в управляющем файле и в заголовках файлов базы данных. После этого Oracle автоматически инициирует процедуру восстановления экземпляра, используя при этом оперативные журналы, чтобы гарантировать восстановление данных в непротиворечивом состоянии, предшествующем моменту сбоя. Сама процедура состоит из двух фаз:

- все зафиксированные транзакции накатываются (выполняются);
- незавершенные транзакции откатываются (отменяются).

Отметим, что транзакция может оказаться незавершенной по одной из двух причин: пользователь не зафиксировал ее или она была зафиксирована пользователем, но еще не подтверждена сервером Oracle к моменту сбоя. Транзакция не считается зафиксированной, пока Oracle не запишет информацию о ней в текущий оперативный журнал и не пошлет клиенту сообщение, подтверждающее факт фиксации.

## Фазы восстановления экземпляра

Процедура восстановления экземпляра включает две фазы - накат (rollforward) и откат (rollback).

Для восстановления экземпляра необходимы журналы. В журналах хранится история всех физических изменений, произведенных в базе данных в результате выполнения транзакций - зафиксированных и незафиксированных.

Для понимания процесса восстановления после сбоя очень важно знать, что такое контрольная точка. В момент фиксации транзакции Oracle записывает все изменения в ассоциированных с ней блоках базы данных в текущий оперативный журнал. Сами же блоки могут быть сброшены на диск и позже. Это означает, что оперативный журнал может содержать изменения, еще не отраженные в файлах данных. Oracle периодически выполняет синхронизацию файлов данных с журналом, в результате чего все зафиксированные к некоторому моменту времени изменения оказываются записанными на диск. В процессе этой операции, которая называется *контрольной точкой*, все блоки базы данных, измененные зафиксированными транзакциями, переписываются в файлы данных на диске. Информация об успешно завершенных контрольных точках заносится в управляющий файл, в заголовки файлов данных и в журнал.

### Накат

В любой момент оперативные журналы опережают файлы данных на какой-то промежуток времени или на какое-то количество зафиксированных транзакций. В процессе восстановления экземпляра этот разрыв устраняется, так что в файлах данных оказываются отражены все транзакции, зафиксированные на момент сбоя. Для этого Oracle пробегает по всему оперативному журналу и воспроизводит изменения, внесенные с момента последней завершенной контрольной точки и до момента сбоя. Эта операция и называется *накатом*.

Для реализации фазы наката Oracle считывает нужные блоки базы данных в системную глобальную область и воспроизводит изменения, ранее внесенные в эти блоки. При этом воспроизводятся не только изменения данных, но и операции отката. Сегменты отката, как и таблицы, состоят из экстентов и блоков данных, и все изменения, сохраненные в блоках сегментов отката, являются частью операции отката данной транзакции. Предположим, например, что пользователь изменил имя работника с «John» на «Jonathan». Обрабатывая журнал, Oracle считывает блок, содержащий строку с информацией о работнике, в кэш и воспроизводит изменение имени. Кроме того, Oracle записывает старое имя «John» в сегмент отката, как и при обработке исходной транзакции.

По завершении фазы наката будут воспроизведены все изменения, выполненные зафиксированными и незафиксированными транзакциями. Незафиксированные транзакции окажутся незавершенными, как то и было в момент сбоя. Теперь наступает очередь следующей логической фазы восстановления экземпляра ~ отката. Но прежде чем переходить к обсуждению отката, посмотрим, как Oracle использует контрольные точки и как время восстановления может зависеть от интервалов между операциями записи контрольной точки.

### Технология быстрого восстановления после сбоя и ограниченное время восстановления

Запись контрольной точки может увеличить интенсивность ввода/вы- вода, поскольку процесс записи должен сбросить все измененные блоки на диск, чтобы актуализировать файлы данных. В версиях до Oracle8 администратор базы данных мог контролировать частоту записи контрольных точек с помощью параметров инициализации LOG_CHECK- POINT_INTERVAL (количество измененных блоков между контрольными точками) и LOG_CHECKPOINT_TIMEOUT (время между контрольными точками в секундах), а также задавая размер журнальных файлов. Кроме того, Oracle всегда записывает контрольную точку при переключении с одного журнала на другой.

Уменьшение интервала между контрольными точками (все равно, выраженного в блоках или в секундах) означает, что за это время накопится меньше изменений и, значит, восстановление пройдет быстрее. Но при этом также может увеличиться количество обращений к диску. Обычно для минимизации количества контрольных точек параметры инициализации устанавливали так, чтобы они записывались только при переключении журналов.

В версии Oracle8i был добавлен параметр инициализации, позволяющий управлять временем восстановления проще и точнее: FAST_ START_IO_TARGET. В ходе восстановления больше всего времени тратится на считывание блоков базы данных в кэш, где в них можно воспроизвести изменения. Этот параметр задает верхнюю границу количества блоков, которые сервер Oracle должен будет считать перед тем, как начать воспроизведение. В результате Oracle динамически изменяет частоту контрольных точек, стараясь уложиться в заданные рамки.

В версии Oracle9i процедура восстановления была еще ускорена. Начиная с последней контрольной точки сервер ищет в журнале блоки, которые содержат несохраненные изменения

и должны быть восстановлены. При втором сканировании изменения применяются только там, где необходимо. Поскольку второе сканирование - это последовательное чтение, а ненужные блоки пропускаются (ввод/вывод с произвольной выборкой), общее время восстановления сокращается.
В версии Oracle9i появилась интересная возможность быстрого восстановления с ограничением по времени. Администратор базы данных задает максимальное время восстановления в секундах (с помощью параметра инициализации FAST_START_MTTR_TARGET, где MTTR означает Mean Time to Recover - среднее время восстановления). Сервер автоматически вычисляет значения параметров FAST_START_IO_ TARGET и LOG_CHECKPOINT_INTERVAL. Оценка величины MTTR вычисляется и помещается в представление V$INSTANCE_RECOVE- RY, что дает возможность калибровки и уточнения оценки со временем.
Сегодня все это стало значительно проще вследствие появления технологии быстрого восстановления после сбоя. Сервер Oracle автоматически ограничивает время восстановления при запуске за счет автона- страиваемой обработки контрольных точек. Впервые эта возможность была реализована в версии Oracle Database 10g.

### Усовершенствования на фазе отката

После завершения фазы наката оказываются восстановленными все незафиксированные транзакции и относящаяся к ним информация об откате. Чтобы вернуться к непротиворечивому состоянию, эти незавершенные транзакции необходимо откатить.

В версиях Oracle до Version 7.3 база данных была недоступна до окончания отката всех незафиксированных транзакций. Хотя администратор базы данных и мог управлять частотой записи контрольных точек, то есть временем, требуемым для выполнения фазы наката, количество транзакций, оставшихся незавершенными в момент сбоя, могло заметно различаться, поэтому время отката нельзя было ни предсказать, ни контролировать. В активно используемой OLTP-системе после сбоя обычно остается довольно много незавершенных транзакций. Поэтому общее время восстановления после сбоя оказывалось переменным и непредсказуемым .

В версии Oracle 7.3 эта проблема была решена посредством *отложенного отката*. Сервер Oracle открывает базу данных сразу после фазы наката, а откат незафиксированных транзакций выполняет в фоновом режиме. Тем самым сокращается время простоя базы и удается справиться с переменностью времени восстановления.

А если транзакция пользователя обратится к блоку базы данных, который содержит изменения, оставшиеся от незафиксированной транзакции? В таком случае выполняется переход в приоритетный режим отката, в котором будут отменены все изменения, произведенные старой транзакцией, и только после этого новая транзакция возобновит работу. Это происходит прозрачно для пользователя, он не получает сообщений об ошибках и не должен запускать транзакцию повторно.

В версии Oracle8i процедура отложенного отката была еще оптимизирована - приоритетный откат, инициированный транзакцией пользователя, теперь ограничен только тем блоком, который этой транзакции нужен. Предположим, что имеется большая незафиксированная транзакция, затрагивающая 500 блоков базы данных. До выхода Oracle8i первая же транзакция, которая попытается обратиться к любому из этих 500 блоков, инициирует переход в приоритетный режим отката, после чего на нее лягут все издержки, связанные с откатом всей старой транзакции. А в режиме быстрого отката потребуется откатить только изменения в блоке, который интересует новую транзакцию пользователя. Новые транзакции не должны дожидаться полного отката больших незафиксированных транзакций.
В современных версиях управление откатом упрощено за счет автоматизации некоторых средств СУБД. Например, начиная с версии Oracle Database 10g автоматически контролируется объем информации, сохраняемой в сегментах отката. Консультант Redo Logfile Size Advisor определяет оптимальный наименьший размер журнального файла исходя из значения параметра FAST_START_MTTR_TARGET и собранной статистики базы данных.

## Защита от системных сбоев

Есть различные подходы к защите системы от последствий сбоев и аварийных остановов, например:

- резервирование компонентов;
- применение опции Real Application Clusters;
- применение программной службы прозрачного восстановления приложений при отказе Transparent Application Failover.

## Резервирование компонентов

Как минимум, различные аппаратные компоненты, составляющие сам сервер базы данных, должны быть отказоустойчивы. Под *отказоустойчивостью* (fault-tolerance) понимается способность системы продолжать работу даже после отказа одного из ее компонентов. Отсюда вытекает необходимость резервирования компонентов, причем система должна уметь распознавать отказ компонента и прозрачно подключать вместо него замену. К основным компонентам системы, нуждающимся в резервировании, относятся:

- дисковые накопители;
- контроллеры дисков;
- центральные процессоры;
- источники питания;
- вентиляторы;
- сетевые карты;
- системные шины.

Сбой диска - наиболее распространенный вид отказов, так как из всех компонентов вычислительной системы именно у дисков время наработки на отказ минимально. Но поскольку для дисков имеется особенно много вариантов резервирования, на их примере проще всего понять, как реализуется высокая доступность на уровне оборудования.

### Резервирование дисков

Среднее время наработки на отказ для отдельного диска довольно велико, но из-за того что современные базы данных размещаются на множестве дисков, отказы возникают относительно часто. Для защиты от сбоев дисков обычно применяют технологию RAID (массив недорогих дисков с избыточностью). Резервирование внешних запоминающих устройств применяется в большинстве систем всех типов и размеров по двум основным причинам - реальная опасность отказа и распространенность готовых сравнительно недорогих RAID-решений.

Технология RAID обеспечивает избыточность двумя методами: *Зеркалирование*
: Все данные дублируются на другом диске.

*Расслоение с контролем четности*
: Данные записываются на несколько дисков, но вместо дублирования вычисляется математическая функция, называемая *четностью*, и результат вычисления сохраняется на отдельном диске. Можно считать, что четность - это сумма расслоенных данных. Если какой-то диск выходит из строя, то хранившиеся на нем данные можно реконструировать, используя оставшиеся диски и данные о четности. Потерянные данные соответствуют единственной переменной уравнения, допускающего однозначное решение.

Есть несколько конфигураций дисков, или типов RAID, формально называемых *уровнями*.

На рис. 11.3 приведены конфигурации дисков для различных уровней RAID.

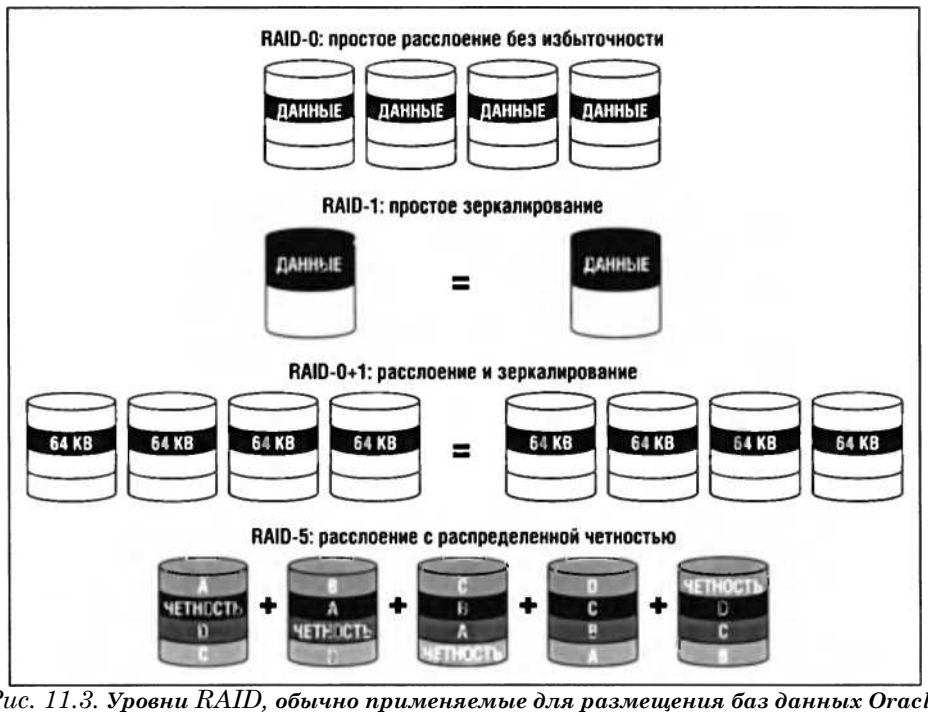



*Рис. 11.3.* ***Уровни RAID, обычно применяемые для размещения баз данных Oracle***

## Автоматическое управление хранением

В версию Oracle Database 10g и более поздние включена подсистема автоматического управления хранением (Automatic Storage Management, ASM). ASM позволяет создать пул внешней памяти на группах дисков и управляет размещением в ней файлов базы данных. Тем самым ASM заменяет сторонние файловые системы и менеджеры логических томов для файлов, управляемых СУБД Oracle. Один экземпляр ASM управляет всеми дисками в одной группе, а в случае RAC- кластера на каждом узле устанавливается отдельный экземпляр ASM.

ASM обеспечивачет режим SAME (Striping and Mirroring Everything - расслоение и "Зеркалирование всего) для многих типов дисков, в том числе для массивов JBOD (Just a Bunch of Disks - простой пакет дисков). Вы можете определить группы дисков и назначить специальную группу, которая будет использоваться в случае отказа какого-либо диска. Зеркалирование также можно организовать на уровне файлов, причем можно задать одно или несколько зеркал. ASM способна распознавать «горячие участки» на диске и перераспределять данные, так чтобы не возникало узких мест. Кроме того, эта подсистема позволяет добавлять новые диски в группу, не прерывая работу. Администратор базы данных может добавлять и удалять диски из группы с помощью Oracle Enterprise Manager.

При добавлении или удалении дисков хранимые данные автоматически перераспределяются. В случае выхода диска из строя данные автоматически переносятся на оставшиеся зеркала. Все это делает подсистему ASM идеальным решением для организации решетки внешних запоминающих устройств и позволяет использовать более дешевые дисковые системы, не снижая уровень доступности и производительности.

В версии Oracle Database 11 g появилась возможность быстрой ресинхронизации зеркал, позволяющая быстрее восстанавливаться после кратковременного сбоя. Теперь ASM можно настроить так, чтобы в случае кратковременных сбоев ресинхронизировались только изменившиеся экстенты.

## Простой перехват управления при отказе

База данных Oracle автоматически восстанавливается после сбоя системы. Механизм автоматического восстановления гарантирует целостность данных, что очень важно для реляционной СУБД, но пока выполняется восстановление, база данных недоступна. Для минимизации времени простоя очень важно уметь быстро обнаруживать сбой системы и запускать процедуру восстановления.

Если отказывает какой-то сервер, то становится бесполезен и работавший на нем экземпляр. В зависимости от причины сбоя отказавший узел не всегда можно быстро вернуть в нормальный режим или уведомить незамедлительно. Желая защитить систему от сбоя единичного узла, компании обычно организуют кластер машин, чтобы реализовать простой *перехват управления при отказе* (failover). В результате уцелевший узел принимает на себя обязанности вышедшего

из строя. Хотя перехват управления при отказе напрямую не связан с надежностью оборудования, автоматизация этой процедуры может сократить время простоя в случае аппаратных сбоев.

Идея очень проста: комбинация программных и аппаратных средств «наблюдает» за кластером. Обычно мониторинг осуществляется путем периодической проверки *тактового сообщения* (hearbeat), которым обмениваются входящие в кластер компьютеры. Если компьютер *A* выходит из строя, то компьютер *B* обнаруживает это, не получая тактового сообщения, и тут же выполняет сценарий, который перехватывает управление дисками, назначает себе сетевой адрес компьютера *A* и запускает процессы, работавшие на компьютере A. С точки зрения СУБД Oracle все происходит так, будто произошел сбой и последующее восстановление экземпляра. Для восстановления после сбоя экземпляр использует управляющие файлы, журналы и файлы базы данных. Тот факт, что теперь экземпляр работает на другой машине, значения не имеет " важны лишь различные файлы Oracle, хранящиеся на диске.

В большинстве решений, обеспечивающих перехват управления при отказе, применяется программа, которая отслеживает состояние определенных процессов, к примеру, фоновых процессов экземпляра Oracle. Если отказал не сам основной узел, а какой-то процесс на нем, то монитор обнаружит этот сбой и запустит сценарий, заданный администратором. Например, в случае аварийного завершения экземпляра Oracle монитор может предпринять три попытки перезапуска экземпляра. Если все три попытки оказались безуспешными, программа может инициировать физическую аппаратную передачу управления другому узлу кластера.

### Время простоя при аппаратном перехвате управления

Время на перехват управления при отказе и, следовательно, продолжительность простоя базы данных зависят от следующих факторов.

*Время, необходимое альтернативному узлу для обнаружения отказа основного узла*

*Время, необходимое альтернативному узлу для выполнения различных действий по запуску*

Время на выполнение таких действий (например, перехват управления дисками, на которых хранится база данных Oracle) может зависеть от системы и определяется путем тестирования. Одна из важных составляющих - время, затрачиваемое на проверку файловых систем. Чем больше база данных, тем больше файловых систем, на которых она размещается. Принимая на себя управление дисками, альтернативный узел должен проверить состояние различных файловых систем на этих дисках.

*Время, необходимое серверу Oracle для восстановления после сбоя*

Как уже отмечалось, этим временем можно управлять с помощью контрольных точек. Oracle предоставляет простой способ контролировать время восстановления путем задания параметра инициализации FAST_START_IO_TARGET или появившегося позднее параметра FAST_START_MTTR_TARGET.

После сбоя экземпляра пользователи обычно получают какое-то сообщение об ошибке и, как правило, пытаются заново войти в систему.

Разработчики приложений могут либо отреагировать на последовательность событий, возникающих при перехвате управления, в единой или специализированной процедуре обработки ошибок либо воспользоваться функцией Transparent Application Failover, описанной ниже.

### Перехват управления при отказе и операционная система

Описанный тип перехвата управления реализован много лет назад. На UNIX-платформах производители обычно предлагают простое решение: две машины, объединенные частной сетью для мониторинга тактовых сообщений, разделяемый диск и кластерное ПО. От СУБД Oracle никакого дополнительного ПО не требуется.

На платформе Windows в дистрибутив Oracle входит подсистема Fail Safe, предоставляющая графический интерфейс для конфигурирования перехвата управления на базе продукта Microsoft Cluster Server. Механизм перехвата управления точно такой же - графический интерфейс служит лишь для удобства. (В последних версиях программы Fail Safe Manager и Real Applications Cluster Guard Manager объединены.)

## Опция Real Application Clusters

Продукт Oracle Parallel Server (OPS) - предшественник Real Application Clusters " появился в Oracle в 1989 году на кластерах VAX производства компании Digital Equipment Corporation, работавших под управлением ОС VMS, и в 1993 году - на платформе UNIX. OPS-кла- стеры развертывались прежде всего для обеспечения высокой доступности базы данных. В версии Oracle9i, вышедшей в 2001 году, OPS был заменен опцией Real Application Clusters.

На первый взгляд, опция Real Application Clusters (RAC) похожа на кластерные решения, описанные в предыдущем разделе. И RAC, и система перехвата управления при отказе нуждаются в кластерном оборудовании с доступом к общим дискам с различных узлов. Основное

различие между ними состоит в том, что в случае простого аппаратного перехвата управления в каждый момент времени может быть активен только один узел. А в случае RAC база данных Oracle распределена по нескольким узлам, на каждом из которых работает экземпляр Oracle. Для обращения к одной и той же базе клиент может подключиться к любому экземпляру.
Поскольку каждый экземпляр Oracle работает на собственном узле, при сбое этого узла он также становится недоступен. Но база данных остается доступной, поскольку экземпляры на уцелевших узлах продолжают работать.
На рис. 11.6 показан кластер под управлением Real Application Clusters.

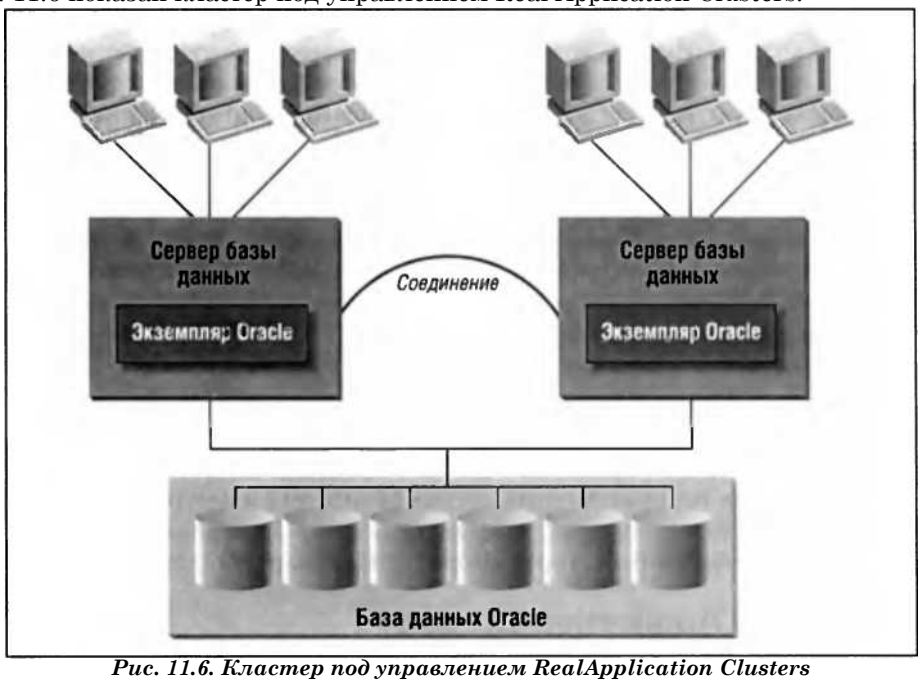



***Рис. 11.6. Кластер под управлением RealApplication Clusters***

## Real Application Clusters и простой перехват управления при отказе

Опция Real Application Clusters, как правило, обеспечивает более высокий уровень доступности, чем простой перехват управления при отказе. Кроме того, она допускает большую гибкость при масштабировании приложений, работающих с базой, на несколько машин. С появлением элемента Enterprise Manager Grid Control в версии Oracle Database 10g администрирование заметно упростилось.
Опция Real Application Clusters повышает уровень доступности, позволяя избежать простоя базы данных. В случае аппаратного перехвата управления база недоступна в течение всего времени, пока альтернативный узел принимает на себя управление, запускает экземпляр и выполняет восстановление после сбоя. При наличии RAC клиент может в любой момент подключиться к уцелевшему экземпляру. Более того, клиент даже не прервет работу, если только не был подключен котказавшему экземпляру. Можно считать, что отказ экземпляра, управляемого RAC, - аналог «частичного затемнения» в отличие от «полного затемнения», происходящего при аппаратном перехвате управления.
Между двумя этими решениями есть и другие существенные отличия:

• Опция RAC позволяет избежать выполнения различных действий, связанных с передачей управления дисками: монтирования томов, проверки целостности файловых систем, открытия файлов базы данных Oracle и так далее. Поэтому время восстановления доступности системы сильно сокращается.

• При использовании RAC не требуется создавать и сопровождать сложные сценарии для контроля над аппаратным перехватом управления. Например, не нужно описывать в сценарии, какими дисковыми томами должен управлять уцелевший узел. Поскольку RAC работает полностью автоматически, не требуется сложная начальная настройка окружения для перехвата, а также текущее сопровождение при добавлении новых дисковых томов. В случае же аппаратного перехвата стоит вам добавить новые тома, забыв прописать их в различных сценариях перехвата, как все решение само перестанет работать!

В простом аппаратном кластере из двух машин обе машины должны обладать одинаковой вычислительной мощностью, так чтобы они могли справиться с полной рабочей нагрузкой. Эквивалентность необходима потому, что в каждый момент времени обработкой занимается только один узел кластера. Если этот узел откажет, то второй должен суметь справиться с той же нагрузкой без потери производительности.

При использовании Real Application Clusters обработка может быть ра- пределена между обоими узлами кластера, тем самым нагрузка на каждую машину сокращается. Тем не менее, для удовлетворения требований бизнеса необходимо гарантировать, что вычислительная мощность каждой машины достаточна для обработки полной нагрузки в одиночку (пусть даже с пониженной производительностью).

Разумеется, применение RAC для распределения нагрузки на несколько машин, в результате чего ресурсы каждой машины используются не на полную мощность, обычно обходится дороже, чем загрузка машин «под завязку». Каждая входящая в кластер машина должна нести определенные накладные расходы на поддержание своей роли в кластере, хотя эти издержки и минимальны. Вы должны решить, что выгоднее - смириться с некоторой потерей производительности в случае отказа узла или потратиться на закупку большего количества узлов или более мощных компьютеров. Анализ ситуации с экономической точки зрения может показать, что падение производительности в случае отказа узла целесообразнее, чем первоначальные затраты на приобретение большего числа узлов или организацию более крупной системы.

В версии Oracle9i настройка и программирование с учетом масштабируемости были существенно упрощены. В версии Oracle Database 10g* споявлением интегрированного кластерного ПО упростилось и развертывание.

## Отказ узла и Real Application Clusters

Экземпляры базы данных защищают друг друга - если какой-то экземпляр откажет, то один из оставшихся обнаружит отказ и автоматически инициирует восстановление RAC. Но сама процедура восстановления совершенно иная, чем в случае перехвата управления. Фактически никакого «перехвата» не происходит - не нужно передавать управление дисками, так как все узлы и так уже имеют доступ к дискам, на которых размещена база данных. Не требуется и запускать экземпляры Oracle на уцелевших узлах, поскольку они и так работают. Сервер Oracle выполняет все необходимые действия без каких-либо сценариев; все, что нужно, уже встроено в ПО Real Application Clusters.

Восстановление Real Application Clusters включает следующие фазы. *Реорганизация кластера*

В случае сбоя экземпляра опция Real Application Clusters должна сначала понять, какие узлы кластеры еще работоспособны. В версии Oracle9i был реализован механизм тактовых сообщений на базе диска: каждый член группы голосует за то, какие члены входят в текущую группу. С помощью алгоритмов арбитража определяется новая конфигурация группы. Эта операция выполняется очень быстро.

*Перестройка базы данных о блокировках*

База данных о блокировках содержит информацию, необходимую для координирования трафика Real Application Clusters. Она распределена по нескольким активным экземплярам. Следовательно, в случае отказа узла часть этой информации теряется. Но оставшиеся узлы обладают избыточной информацией, пользуясь которой они могут воссоздать утраченные данные. После того как членство в кластере установлено, оставшиеся экземпляры перестраивают базу данных о блокировках. Как долго продлится эта операция, зависит от количества подлежащих восстановлению блокировок, а также от того, сколько узлов уцелело - один или несколько. Oracle ускоряет процедуру восстановления блокировок за счет того, что оптимизация размещения эталонных данных о блокировках производится в фоновом режиме, когда пользователи уже обращаются к базе. Если кластер состоит из двух узлов, то после сбоя одного из них второй действует как диктатор, поэтому операции с блокировками выполняются очень быстро.

*Восстановление экземпляра*

После того как база данных о блокировках перестроена, выполняется процедура восстановления отказавшего экземпляра с использованием его журцалов. Она аналогична восстановлению после сбоя одного экземпляра - сначала производится фаза наката, а затем не блокирующая, отложенная фаза отката. Ключевое отличие в том, что восстановление не ициииируется перезапуском отказавшего экземпляра. Вместо этого процедуру восстановления выполняет экземпляр, обнаруживший отказ.

Пока происходит восстановление RAC, клиенты, подключенные к уцелевшим экземплярам, сохраняют соединение и могут продолжать работу. В некоторых случаях пользователи могут заметить небольшое замедление реакции, но их сеансы не прерываются. Клиенты, которые были подключены к отказавшему экземпляру, могут переподключиться к любому из уцелевших экземпляров и возобновить работу. Незафиксированные транзакции будут откачены, поэтому их придется повторить. Обработка запросов, выполнявшихся в момент сбоя, также прекращается, однако опция Transparent Application Failover (TAF) позволяет автоматически

продолжить выполнение запроса на уцелевшем узле, не вынуждая пользователей повторять запрос. Кроме того, TAF можно использовать и для повторного запуска транзакций без вмешательства пользователя.

### Опция Parallel Fail Safe/RACGuard

Опция Parallel Fail Safe в версии Oracle9i была переименована в RAC- Guard и интегрирована в ядро RAC в Oracle Database 10^. До выхода Oracle Database 10g* это была одна из функций RAC, которая опиралась на кластерное ПО, предлагаемое поставщиками системы. В Oracle Database 10^ в состав СУБД входит кластерная файловая система.

RACGuard обладала следующими возможностями:

• автоматизированное быстрое восстановление с ограничением по времени после сбоя экземпляра Oracle;

• автоматический сбор диагностической информации;
• гарантированные основная и резервная конфигурация;
• поддержка таких функций, как Transparent Application Failover (см. следующий раздел);

• заблаговременное установление соединения клиента с резервными экземплярами для ускорения переподключения.

## Опция Oracle Transparent Application Failover (TAF)

Опция прозрачного восстановления приложений (Transparent Application Failover, TAF) впервые появилась в версии Oracle8. Она позволяет незаметно для пользователя передавать открытый им сеанс с одного экземпляра Oracle на другой. TAF позволяет скрыть отказ экземпляра в целях повышения уровня доступности и перевести пользователей сперегруженного экземпляра на менее занятый. На рис. 11.7 показана совместная работа TAF и Real Application Clusters.

Как видно из рисунка, TAF может автоматически переподключать клиентов к другому экземпляру, обслуживающему ту же базу данных, что и исходный. Достоинства TAF с точки зрения высокой доступности заключаются в следующем.

*Прозрачное переподключение*

Клиенту не нужно вручную устанавливать новое соединение с уцелевшим экземпляром. В целях оптимизации можно так сконфигурировать TAF, чтобы в момент входа клиента в систему сразу создавалось соединение не только с основным, но и с альтернативным экземпляром. Таким образом, удается избавиться от накладных расходов на создание нового соединения на этапе автоматического перехвата управления. Если в системе одновременно работает много клиентов, то заблаговременное установление соединения позволяет избежать задержек из-за того, что после сбоя основного экземпляра альтернативный «утонет» в огромном количестве запросов на открытие соединения.

*Автоматический повтор запросов*

TAF может автоматически повторять запросы, выполнявшиеся в момент отказа экземпляра, и возобновлять отправку результатов клиенту. Oracle повторно выполняет запрос к базе данных в том состоянии, которое существовало в момент его первоначального предъявления. Механизм согласованности по чтению позволяет дать правильный ответ вне зависимости от того, что происходило с базой с момента начала выполнения запроса. Однако если пользователь запрашивает «следующую» строкурезультатов, Oracle понадобится обработать все предшествующие строки, что может приводить к задержке.

*Функции обратного вызова*

В версии Oracle8i разработчики приложений получили возможность зарегистрировать «функцию обратного вызова» TAF. После того как TAF успешно завершит переподключение клиента к альтернативному экземпляру, зарегистрированная функция будет вызвана автоматически. Разработчик может использовать ее для повторной инициализации различных аспектов сеанса.

*Уведомление приложений об отказе*

Разработчик может использовать TAF для написания приложений, уведомляемых об отказе. Такое приложение может самостоятельно повторять транзакции, прерванные из-за отказа основного экземпляра, что позволяет дополнительно сгладить последствия отказа. Отметим, что в отличие от повтора запросов, TAF не умеет автоматически повторять прерванные транзакции, а лишь предоставляет прикладному программисту средства для прозрачного восстановления.

### Принцип работы TAF

Опция TAF реализована на уровне Oracle Call Interface (OCI) - низкоуровневого API для создания и управления соединениями с базой данных Oracle. Если экземпляр, к которому подсоединился клиент, отказывает, то обслуживающий клиента серверный процесс прекращает существование. Уровень OCI на стороне клиента способен обнаружить отсутствие серверного процесса на другом конце канала и

автоматически создать новое соединение с другим экземпляром. Альтернативный экземпляр, к которому TAF переподключает пользователей, задается в конфигурационных файлах Oracle Net, описанных в документации.

Поскольку OCI " это низкоуровневый API, программирование для него требует от разработчика больше усилий и знаний. К счастью, корпорация Oracle использует OCI для написания клиентских инструментов и различных драйверов, поэтому приложения могут задействовать TAF, применяя эти инструменты. Особенно полезна поддержка TAF в драйверах ODBC и JDBC, поскольку это означает, что средства TAF доступны любому клиентскому приложению, применяющему эти драйверы для соединения с Oracle. Например, TAF может обеспечить автоматическое переподключение для любого написанного сторонней фирмой инструмента запросов, работающего на уровне ODBC. Для реализации TAF с помощью ODBC следует настроить источник данных ODBC, так чтобы он обращался к имени службы Oracle Net, для которого в конфигурационных файлах Oracle Net прописано использование TAF. Поскольку интерфейс ODBC работает с Oracle Net, ему становятся доступны и средства TAF.

### TAF и различные конфигурации Oracle

Хотя комбинация TAF с Real Application Clusters - самый очевидный способ достижения высокой доступности, TAF можно также использо-

вать с единственным экземпляром Oracle или с несколькими базами данных, доступ к которым предоставляет единственный экземпляр. Вот несколько возможных конфигураций:

• TAF может автоматически переподключать клиенты к их исходным экземплярам в случае, когда экземпляр отказал, а узел - нет. Автоматизированная система мониторинга, например Oracle Enterprise Manager, может быстро обнаружить отказ экземпляра и перезапустить его. Технология быстрого восстановления позволяет свести к минимуму время восстановления после сбоя. Если пользователь не вводит данные беспрерывно, то он даже может не узнать, что экземпляр аварийно завершился и был автоматически перезапущен.

• В простых кластерах TAF может переподключить пользователей к экземпляру, запущенному на уцелевшем узле в результате перехвата управления после сбоя. Переподключение произойдет только после того, как на альтернативном узле запустится Oracle и закончится процедура восстановления после сбоя.

• Если имеются две разных базы данных и каждая обслуживается единственным экземпляром, то TAF может переподключать клиенты к тому экземпляру, который дает доступ к другой базе, находящейся в другом центре обработки данных. Очевидно, для этого необходимо реплицировать соответствующие данные из одной базы в другую. Впрочем, в Oracle имеется автоматизированный механизм репликации данных, который рассматривается в разделе «Полный отказ центра обработки данных» ниже.

## Восстановление после сбоев

Несмотря на широкую распространенность резервированной или защищенной дисковой памяти, сбои носителей все же случаются. Если врезультате сбоя диска один или несколько файлов данных Oracle пропали, необходимо восстановить утраченные данные из резервных копий.

Привести к потере данных могут не только сбои носителей, но и простые ошибки человека или компьютера. Например, администратор может случайно удалить файл данных, или данные на дисках могут оказаться испорченными из-за сбоя подсистемы ввода/вывода. Чтобы встретить такого рода проблемы во всеоружии, необходимо заранее подготовить эффективную стратегию резервного копирования и восстановления, тщательно изучив новые возможности Oracle, например ретроспекцию (Flashback).

## Разработка стратегии резервного копирования и восстановления

Правильно организованные разработка, документирование и тестирование стратегии резервного копирования и восстановления - один из важнейших аспектов обслуживания базы данных Oracle. Следует тщательно протестировать все этапы процедуры резервного копирования и восстановления, убедившись, что все работает как надо. Когда гром грянет, процедура восстановления *должна* работать безукоризненно.

В некоторых компаниях тестируют только процедуру резервного копирования, забывая проверить, как работает восстановление с имеющихся копий. И лишь при сбое, когда приходится доставать резервные копии, обнаруживается, что по какой-то причине они не работают. Абсолютно необходимо протестировать весь цикл начиная с резервного копирования и заканчивая восстановлением.

## Снятие резервных копий в Oracle

В Oracle есть два основных типа резервного копирования.

*Горячее резервное копирование*

Файлы данных для одного или нескольких табличных пространств копируются без останова базы данных.

*Холодное резервное копирование*

База данных останавливается, после чего копируются все файлы данных, журналы и управляющие файлы.

При снятии горячей резервной копии необязательно копировать сразу все файлы данных. Например, можно каждую ночь копировать по одной группе файлов. Но необходимо иметь копии архивных журналов за все дни, начиная с даты самого старого скопированного файла данных и до текущей, поскольку они понадобятся для выполнения наката с момента снятия этой копии.

Иногда администраторы, обслуживающие сверхбольшие базы данных, копируют некоторые файлы данных в несколько проходов. Бывает, что файлы с часто изменяемыми данными копируются чаще (например, каждый день), а файлы с относительно статичными данные - реже (например, раз в неделю). Есть команды, позволяющие снять резервные копии управляющих файлов; это следует делать после того, как скопированы все файлы данных.

Если архивные журналы не используются, то можно снимать только полные холодные копии. Если же архивные журналы ведутся, то резервное копирование можно выполнять, не останавливая базу данных.

Вне зависимости от типа резервного копирования следует включать в копию файл *INIT.ORA* или *SPFILE* и файлы паролей - без них СУБД Oracle работать не будет. Хотя это и необязательно, рекомендуется также скопировать различные сценарии создания и последующего развития базы данных. Это важная часть документации по структуре и эволюции базы.

Дополнительную информацию о различных типах и вариантах резервного копирования вы найдете в официальной документации Oracle и в книгах, указанных в приложении В.

## Восстановление из резервных копий

В Oracle есть два типа восстановления - с применением и без применения архивных журналов.

*Полное восстановление базы данных*

Если архивные журналы не ведутся, то возможно только полное холодное резервное копирование. Соответственно, можно выполнить лишь полное восстановление базы данных. Из резервной копии восстанавливаются файлы базы данных, журналы и управляющие файлы. База данных оказывается в том же состоянии, в котором находилась на момент снятия копии. Все, что было сделано после снятия копии, теряется, и восстанавливать базу нужно целиком, даже если испорчен только один файл данных. Из-за возможности потери данных вкупе с необходимостью восстанавливать всю базу для исправления последствий частичного сбоя большинство организаций предпочитают эксплуатировать базу в режиме ARCHIVELOG. На рис. 11.8 показана схема резервного копирования и восстановления для базы данных без архивных журналов.

*Частичное восстановление с накатом*

Если база данных работает в режиме ARCHIVELOG, то можно восстановить только поврежденные файлы данных и накатить информацию из журналов за период с момента снятия копии и до момента сбоя. С помощью архивных и оперативных журналов можно вос произвести все изменения в восстановленных файлах данных и привести их к состоянию на тот же момент, что и в остальной базе. Эта процедура минимизирует время выполнения операций восстановления. Подобное частичное восстановление выполняется при остановленной базе данных. Альтернативно можно вывести требуемые табличные пространства из оперативного режима и восстановить их, пока остальная база данных работает.

# Лекция 12. Oracle и аппаратная архитектура.

## Основные компоненты системы

Всякое обсуждение аппаратной части системы начинается с обзора компонентов платформы и влияния, оказываемого ими на систему в целом. Внутри любого компьютера вы обнаружите одни и те же базовые компоненты:

- один или несколько процессоров, исполняющих машинные команды, возможно, с несколькими ядрами для повышения производительности;
- оперативную память, в которой хранятся исполняемые команды и данные;
- подсистему ввода/вывода, в которую обычно входят диски, контроллеры устройств для считывания данных и программ с физических носителей и сетевые контроллеры.

Количество и характеристики различных компонентов определяют стоимость системы и возможность ее масштабирования. Машина с четырьмя процессорами обычно дороже и способна выполнить больше работы в единицу времени, чем однопроцессорная; современные компоненты, например процессоры, как правило, быстрее и дешевле прежних моделей.

При проектировании систем оперативной обработки транзакций (OLTP) во главу угла часто ставится пропускная способность. Что касается систем для бизнес-анализа и работы с хранилищами данных, то раньше считалось, что их производительность лимитирована мощностью процессоров и объемом памяти. Однако то и другое за последние годы (особенно прошедшие с момента выхода прежних изданий этой книги) существенно возросло, и адекватные характеристики ввода/вывода теперь также заслуживают повышенного внимания.

Каждый компонент системы характеризуется временем доступа и доставки данных - *задержкой*. В табл. 12.1 различные компоненты классифицированы по уровням задержки. Кроме того, каждый компонент характеризуется его емкостью.

Меньше всего задержка у процессора и кэша памяти первого уровня (Ll-кэш), но и емкость у них тоже наименьшая. Наибольшей емкостью обладают диски, им же свойственна и максимальная задержка.

При настройке любой базы данных Oracle очень важно минимизировать количество операций чтения с источников, имеющих большую задержку (например, с диска). А когда к дискам все же приходится обращаться, следует избегать образования узких мест в подсистеме ввода/вывода. Если СУБД Oracle большую часть данных получает из памяти, а не с диска, то общая задержка системы уменьшается, а производительность возрастает.

*Таблица 12.1. Типичные значения емкости и задержки различных компонентов системы*

| Компонент | Типичная емкость | Типичная задержка |
|---|---|---|
| Центральный процессор | Нет | Нет |
| Кэш первого уровня (внутри ЦП) | От десятков до сотен килобайт | 10 наносекунд |
| Кэш второго уровня (рядом с ЦП) | Мегабайты | 40-60 наносекунд |
| Кэш третьего уровня (на системной плате) | Десятки мегабайт | 120 наносекунд |
| Основная память | От мегабайта до терабайта и выше | 1-10 тыс. наносекунд |
| Диск | От гигабайта до сотен терабайт | 1-10 млн наносекунд |

## Системы с симметричной многопроцессорной обработкой

Одним из факторов, ограничивающих полезность однопроцессорных систем, является быстродействие процессора - ведь это ресурс, разделяемый всеми приложениями. Системы с симметричной многопроцессорной обработкой (SMP) были придуманы в попытке преодолеть это ограничение путем подключения к шине памяти дополнительных процессоров.

У каждого процессора имеется свой кэш. Иногда данные, находящиеся в кэше одного процессора, бывают нужны другому. Из-за возможности такого разделения данных процессоры должны уметь «подсматривать» за шиной памяти, чтобы знать, где находятся копии данных и не обновляются ли данные в текущий момент. «Подсматривание» прозрачно реализуется

операционной системой, управляющей SMP-компьюте- ром. На таких платформах может работать Oracle редакций Standard Edition One, Standard Edition или Enterprise Edition. (В редакциях Standard Edition One и Standard Edition количество поддерживаемых процессоров ограничено, а в Enterprise Edition может быть любым.)

SMP-системы появились еще в 1980-х как платформы для систем среднего масштаба, преимущественно под управлением UNIX. Сегодня они считаются серверами начального уровня и в основном оборудуются 64-разрядными процессорами (сменившими 32-разрядные). Самые популярные ОС этой категории - различные варианты Windows и Linux.

Более масштабируемые SMP-серверы, поставляемые такими компаниями, как HP, IBM и Sun, отличаются конструкцией. Так, есть SMP- системы с многоядерными процессорами, увеличенным кэшем второго уровня, более быстрой шиной памяти или высокоскоростными каналами ввода/вывода. Все ухищрения призваны исключить потенциальные узкие места, снижающие производительность. Для развертывания Oracle на SMP-серверах высокой ценовой категории обычно устанавливают ОС UNIX или Linux.

Количество процессоров в SMP-системе ограничено масштабируемостью системной шины (памяти). По мере добавления к ней все новых процессоров наблюдается насыщение шины трафиком между ними и памятью.
Системы с 64-разрядными процессорами способны более эффективно обрабатывать большие объемы данных, чем с 32-разрядными; они поддерживают десятки процессоров и сотни гигабайтов основной памяти.
Конечно же, чтобы воспользоваться всеми преимуществами SMP-ар- хитектуры, СУБД должна поддерживать параллелизм. Выполнение запросов и другие операции манипулирования данными (DML), а также загрузку данных, Oracle может распараллеливать, обращая себе на пользу достоинства мультипроцессорных систем. Как и любая программная система, Oracle выигрывает от распараллеливания операций, что следует из закона Амдала.

Один из пионеров разработки больших ЭВМ Джин Амдал (Gene Amdahl) в 1967 году сформулировал этот закон для описания производительности обработки нагрузки, включающей параллельные и последовательные вычисления. Он ясно показывает, что чем больше вычислений удается в распараллелить, тем больше выигрыш от увеличения числа процессоров. Наоборот, чем больше в приложении доля последовательных вычислений, тем больше полное время выполнения, поскольку выигрыш от увеличения числа процессоров незаметен на фоне времени, затрачиваемого напоследовательные вычисления. Иными словами, добавление процессоров не ускоряет выполнение последовательных операций.
В каждой новой версии Oracle увеличивалось количество распараллеливаемых операций. Это относится как к выполнению запросов, так и к операциям настройки и обслуживания базы данных.
При выполнении параллельных операций Oracle задействует все доступные процессорные ресурсы. Если в вашей системе возможности процессоров полностью исчерпаны, то распараллеливание не повысит производительность; на самом деле, ситуация может даже ухудшиться из-за того, что ресурсы процессоров расходуются и на управление параллельными процессами. Механизм адаптивного параллелизма в Oracle позволяет автоматически уменьшать степень параллелизма, чтобы избежать такого развития событий.

## Кластерные системы

Кластерные системы, появившиеся в 1980-х на платформе DEC VAX- cluster, позволили разработать решения с высоким уровнем доступности и масштабируемости. Кластеры позволяют объединить все компоненты отдельных машин, включая процессоры, память и подсистемы

ввода/вывода, в единую аппаратную сущность. Однако обычно кластер строится на базе разделяемых дисков, подключаемых к нескольким «узлам» (вычислительным системам). Высокоскоростной канал связи между системами позволяет обмениваться данными и командами без записи на диск (рис. 12.3). В каждой системе работает своя копия ОС и свой экземпляр Oracle. Решетки, описанные далее, как правило, строятся из нескольких очень больших кластеров.

*Рис.12.3. Типичный кластер (показаны только две системы)*


Поддержка кластеров в Oracle восходит еще к платформе VAXcluster. В Oracle применяется изощренная модель блокировок, позволяющая нескольким узлам обращаться к общим данным на дисках. Такая модель необходима, поскольку каждая машина должна знать о блокировках, поставленных другими, физически отдельными машинами в кластере.

В настоящее время в качестве кластерного решения корпорация Oracle предлагает опцию Real Application Clusters (RAC) (она заменила продукт Oracle Parallel Server - OPS, доступный до выхода версии Oracle9i). Чаще всего RAC применяется в кластерах под управлением Windows, Linux или UNIX. СУБД Oracle содержит интегрированный менеджер блокировок, выступающий посредником между различными серверами (узлами), пытающимися обновить данные в одном и том же блоке.

RAC обеспечивает полную поддержку технологии Cache Fusion, позволяющей хранить блокировки в памяти без частой записи на диск. Эта

технология отличается от стандартных механизмов блокировки тем, что ставит блокировки на блоки данных, а не на строки. Посредник необходим, потому что два узла могут попытаться обратиться к разным строкам в одном и том же физическом блоке - минимальной единице памяти, с которой умеет работать Oracle.

Технология Cache Fusion с самого начала заметно повысила производительность операций чтения/записи по сравнению с OPS, а с выходом Oracle9i RAC появились и дополнительные улучшения. Сегодня Oracle поддерживает протокол Sockets Direct Protocol (SDP) и протоколы асинхронного ввода/вывода, требующие меньше ресурсов, чем в первых реализациях RAC, основанных на традиционном стеке TCP/IP. В последних версиях производительность еще возросла за счет использования более быстрых каналов связи, например сетей Infiniband спротоколом Reliable Datagram Sockets (RDS). Так, задержка в сети Infiniband при передаче данных между узлами примерно в десять раз меньше, чем в сети Gigabit Ethernet (как правило, 70-80 микросекунд).

До появления Real Application Clusters при конфигурировании кластера приходилось выбирать между высокой пропускной способностью и высокой доступностью. Если предпочтение отдавалось высокой доступности, то при отказе одного узла второй узел, подключенный к общему диску, мог получить доступ к тем же данным. Обработка запросов доходила до конца без какого-либо вмешательства за счет прозрачной передачи клиента. RAC обеспечивает одновременно доступность и масштабируемость, поскольку каждый узел кластера может перехватить управление при отказе любого другого узла.

Технология Real Application Clusters все шире применяется на платформах Windows и Linux, не допускающих адекватного масштабирования, или в качестве замены более дорогим решениям на базе UNIX. Кластерные решения имеет смысл развертывать и для обеспечения высокой доступности. На кластерной платформе Windows в качестве альтернативы RAC можно использовать опцию Oracle Fail Safe, хотя в этом случае данные не разделяются обеими системами, и вторую можно использовать только для резервного доступа к данным. Поскольку конкурентный доступ не поддерживается, Fail Safe не может предложить тот же уровень масштабируемости, что Real Application Clusters.

В предыдущих изданиях этой книги мы описывали очень дорогой вариант кластеров - массивно-параллельные системы (MPP). Такая система, по сути, представляет собой кластер в одном корпусе, узлы которого соединены сверхвысокоскоростными линиями связи на базе патентованных технологий (рис. 12.4). В настоящее время поставщики открытых систем очень редко продают такие платформы, поскольку с массового рынка их вытеснили кластеры из более дешевых компонентов (узлов).

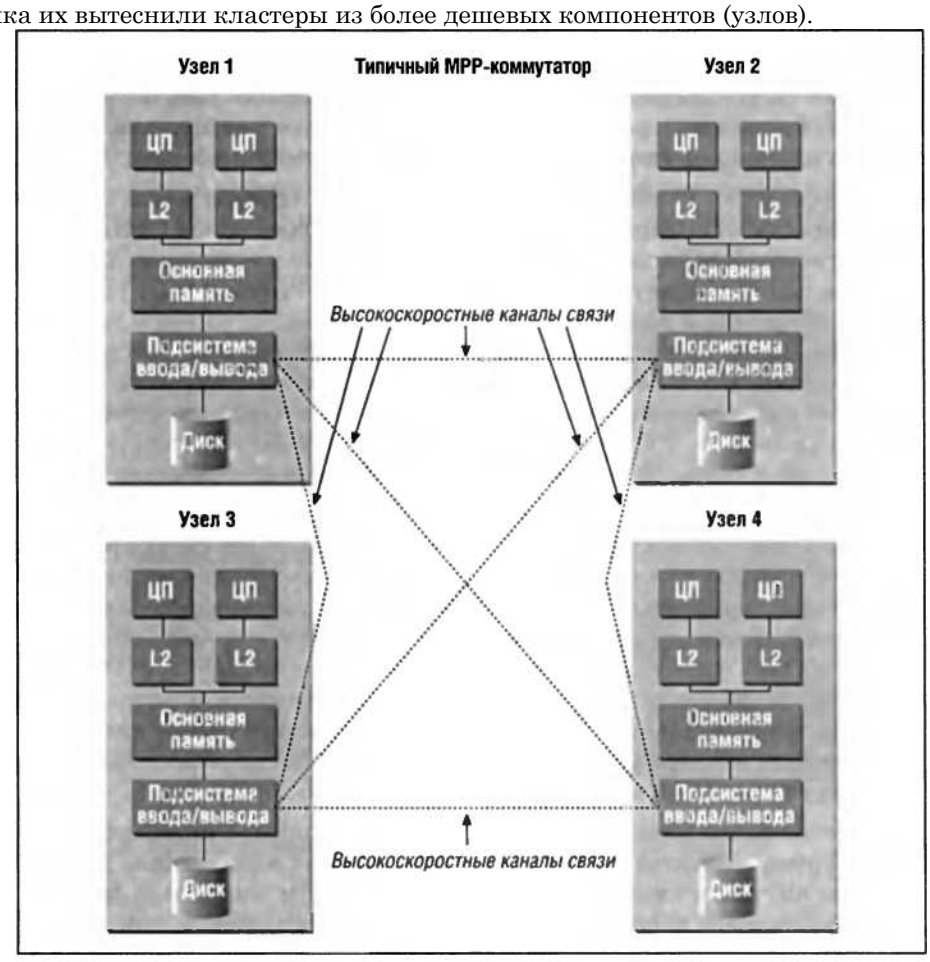



***Рис.12.4.** Массивно-параллельная система*

## Системы с неоднородной архитектурой памяти

Компьютеры с неоднородной архитектурой памяти (Non-Uniform Memory Access, NUMA), появившиеся в середине 1990-х, обеспечивают еще более высокую пропускную способность, чем SMP-системы, за счет того, что SMP-компоненты в них объединены распределенной памятью (рис. 12.5). Как и кластеры, такие системы позволяют масштабировать не только процессоры, но также память и подсистемы ввода/ вывода. Основное различие в том, что всей платформой управляет единственный экземпляр операционной системы, а для синхронизации данных применяется схема когерентности кэшей на основе каталогов. Время доступа к памяти другого узла составляет порядка сотен микросекунд, то есть гораздо быстрее, чем к диску в кластерных конфигурациях, и лишь немногим медленнее, чем к локальной шине памяти в одной SMP-системе. Объем памяти может достигать нескольких терабайтов.

Это дает NUMA-системам несколько важных преимуществ по сравнению с кластерными решениями:

• Не нужно разрабатывать и сертифицировать для работы на таких машинах параллельные версии приложений (хотя оптимизация приложений специально для архитектуры NUMA может дать дополнительный выигрыш в производительности).

• Администрировать NUMA-системы значительно проще, чем кластеры, поскольку в типичном случае имеется лишь один экземпляр операционной системы и один экземпляр СУБД.

Сегодня примером NUMA-системы, для которой продемонстрирована масштабируемость промышленных баз данных объемом десятки терабайтов, может служить Hewlett Packard Superdome. Поскольку эта платформа администрируется и ведет себя точно так же, как SMP-сис- темы, аналогичны и компромиссы, на которые приходится идти (хотя NUMA-системы обычно стоят дороже).

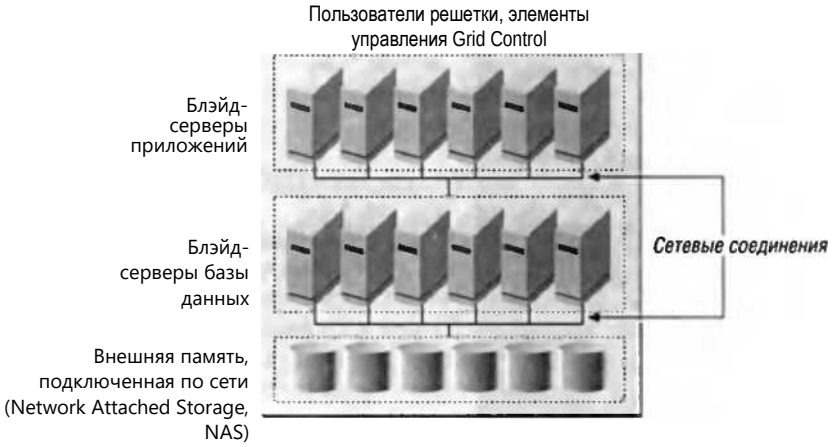
## Grid-вычисления

Буква *«g»* в названиях продуктов Oracle начиная с Oracle Database 10g означает заинтересованность компании в реализации grid-вычислений. *Решеткой* (grid) называется пул компьютеров, предоставляющих приложениям ресурсы по мере необходимости. При этом ставится цель обеспечить прозрачное масштабирование вычислительных ресурсов на большое сообщество пользователей. Это можно сравнить с электроэнергетической компанией, которая в периоды пикового потребления заимствует мощности других предприятий, входящих в состав единой энергетической системы. Вычислительные решетки сходным образом динамически предоставляют процессорные ресурсы и данные (рис. 12.6). Эта технология основана на применении СУБД Oracle с опцией RAC.

В версии Oracle Database 10g появилось несколько важных механизмов, обеспечивающих предоставление решеткой ресурсов по запросу.

Механизм автоматического выделения и освобождения ресурсов сервера на основе конфигурационных файлов и правил перехвата

*Рис.12.6. Пример конфигурации решетки, состоящей из блэйд-серверов и кластера*



управления при отказе. Запросы к службе автоматически переправляются наименее загруженному серверу. Если какой-то сервер выходит из строя, то запросы автоматически перераспределяются между оставшимися.

*Веб-службы*

Веб-службы - неотъемлемая часть решеточной архитектуры, так как работающим в решетке приложениям необходим такой же прозрачный доступ к компонентам (или службам), какой пользователи имеют к самим приложениям. Веб-службы базы данных поддерживают запросы, обмен сообщениями и манипуляцию данными, а также обеспечивают доступ к Java и PL/SQL и полную поддержку XML.

*Пошаговое обновление*

Процедура пошагового обновления (rolling upgrade) позволяет остановить некоторые узлы решетки, обновить на них программное обеспечение, а затем вернуть в рабочий режим. Затем те же действия повторяются на других узлах. В конечном итоге ПО Oracle оказывается обновленным на всех узлах решетки при непрерывном доступе к базе данных.

*Автоматическое управление хранением (Automatic Storage Management, ASM )*

Подсистема ASM позволяет администрировать множество узлов путем автоматического перераспределения данных между дисками, а также добавлять новые диски в общий пул внешней памяти.

*Элемент Enterprise Manager Grid Control*

Элемент Enterprise Manager Grid Control позволяет централизованно управлять всей инфраструктурой решетки, включающей базы данных на RAC-кластере, систему хранения, ПО промежуточного уровня Oracle Application Servers/Fusion Middleware и сетевые службы.

В версию Oracle Database llg включено управление многоуровневыми службами и усовершенствована поддержка со стороны ASM пошагового обновления, автоматического обнаружения и восстановления сбойных блоков и быстрой ресинхронизации зеркал. Автоматический диагностический монитор Automatic Database Diagnostics Monitor (ADDM) для RAC выявляет большинство проблем, влияющих на производительность решетки в целом, в том числе - связи между глобальными кэшами, перегрузку менеджера блокировок, конфликты при доступе к глобальным ресурсам (например, к подсистеме ввода/вывода) и неодинаковое время реакции различных экземпляров. Теперь на любую СУБД Oracle, в том числе развернутую в RAC-кластере, можно оперативно накладывать заплаты.

## Технологии дисков и систем хранения

До сих пор в мы обсуждали аппаратные архитектуры с точки зрения возможностей повышения производительности за счет наращивания системных ресурсов - процессоров, памяти, пропускной способности ввода/вывода - и распараллеливания работы. Но имеется еще один важный способ аппаратного увеличения производительности - оптимизация ввода/вывода, заключающаяся в правильном распределении данных по дискам и обеспечении достаточного количества путей доступа к данным. Есть эвристическое правило: подсистема ввода/вывода должна обеспечить передачу 1 Гбайт в секунду на каждые 4 процессора, но не меньше 2 Гбайт в секунду в целом. Поскольку для дисков характерна наибольшая задержка, еще один аспект оптимизации ввода/вывода - сохранение считанных с дисков данных в оперативной памяти.

В документации Oracle хорошо подобранные конфигурации - с удачным сочетанием характеристик ввода/вывода (особенно накопителей, обеспечивающих пути доступа к внешней памяти), памяти и процессоров - называются *сбалансированными*. Выше уже отмечалось, что начиная с версии Oracle Database 10g в состав Oracle входит подсистема ASM, заметно упрощающая рутинное администрирование внешней памяти. Однако зачастую у поставщиков оборудования трудно получить систему хранения в заданной конфигурации, особенно если речь идет о хранилищах данных. Кроме того, хотя емкость дисков постоянно растет, сравнимого прогресса в части времени доступа не наблюдается. Все это подвигло корпорацию Oracle совместно с основными поставщиками разработать аппаратные платформы и системы хранения ряда эталонных конфигураций, чтобы помочь пользователям точнее оценить требования к начальной конфигурации. Эта инициатива получила название Information Appliance Initiative.

### Стратегии развертывания дисков

Диски часто подключаются напрямую к системе, причем в дорогие системы включаются более быстрые дисковые контроллеры, обеспечивающие повышенную скорость ввода/вывода. С ростом пропускной способности сетей в качестве экономичных альтернатив стали предлагаться сетевые технологии подключения внешней памяти Network Attached Storage (NAS) и Storage Area Network (SAN). Кроме того, есть различные способы конфигурирования дисков с резервированием, позволяющие устранить возможность потери доступа к данным из-за единичного отказа диска.

Обычно диски собираются в массивы, причем промышленным стандартом является технология RAID. RAID-массив можно применять в любой из рассмотренных выше конфигураций для повышения производительности и надежности. Кроме того, начиная с версии Oracle Database 10g подсистема автоматического управления хранением Automatic Storage Management (ASM) предоставляет значительную часть функциональности RAID-массивов (например, расслоение и зеркалирование) для набора обычных дисков.

В версии Oracle9i было реализовано сжатие внутри базы данных для уменьшения требуемой дисковой памяти, особенно в хранилищах данных. Повторяющиеся значения в блоке данных устраняются за счет хранения дубликатов в таблице символов, расположенной в начале блока. Все вхождения повторяющихся значений заменяются короткими ссылками на элементы этой таблицы. В версии Oracle Database llg добавлена опция Advanced Compression Option для операций вставки, обновления и удаления, что существенно в OLTP-системах. Типичная степень сжатия сегодня составляет 50%. Помимо уменьшения места, занимаемого на диске, сжатие может способствовать и повышению производительности, если сжатые данные целиком помещаются в кэш (не требуя доступа к диску).

Поскольку емкость дисков постоянно растет, а цена снижается, многие организации предпочитают хранить все данные на дисках, обеспечивая оперативный доступ к ним в приложениях для работы с хранилищами данных и бизнес-анализа. Учитывая тот факт, что диски, обеспечивающие максимальную производительность, обычно стоят дороже и имеют меньшую емкость, наблюдается тенденция сочетать их с относительно медленными, но более емкими (и дешевыми) дисками для хранения данных, к которым обращаются редко. Подсистема управления жизненным циклом информации Information Lifecycle Management (ILM) и в особенности программа ILM Assistant, появившаяся в 2007 году, упрощают администрирование в таких условиях.

## Какую платформу выбрать?

С неограниченным бюджетом на оборудование решение было бы простым: определяем, какая пропускная способность и надежность нам нужна, - и просто покупаем отвечающую этим требованиям аппаратуру! К сожалению, такой бюджет трудно представить, поэтому решение о выборе оборудования часто является компромиссным. Но с момента выхода первого издания этой книги цены продолжали снижаться, так что теперь принять его стало проще.

### Сравнение платформ

Обычно сервер Oracle развертывается на SMP-системе, дающей подходящее соотношение цены и мощности. Выбор именно архитектуры SMP диктуется следующими соображениями:

- SMP-система масштабируется проще и в более широких пределах, чем однопроцессорная;
- наличие 64-разрядных процессоров и операционных систем с поддержкой больших объемов памяти позволяет SMP-системам работать с очень большими базами данных (до десятков терабайтов);
- в отличие от кластеров, в SMP-системах имеется единственный экземпляр ОС и Oracle, поэтому упрощается администрирование и сопровождение;
- для работы в SMP-системах сертифицировано больше приложений, чем для кластеров;
- SMP-система может оказаться дешевле, чем сравнивая по количеству процессоров NUMA-система, кластер или решетка, поскольку не приходится дублировать память и подсистемы ввода/вывода.

Мы вовсе не хотим сказать, что другие конфигурации не стоит даже рассматривать. Если требования к масштабируемости превышают возможности SMP-машин, то альтернативы кластеру или решетке может и не оказаться. Стоимость кластера можно снизить за счет использования узлов из компонентов «массового потребления» в RAC-конфигурации. При тщательном планировании и надлежащем управлении вычислительными ресурсами предприятия такие конфигурации вполне способны обеспечить необходимую мощность и степень доступности.

Табл. 12.2 позволяет сравнить достоинства различных платформ с точки зрения масштабируемости, управляемости и доступности.

*Таблица 12.2. Сравнение достоинств различных платформ развертывания*

| Оценка | Масштабируемость | У правляемость | Доступность |
|---|---|---|---|
| Наилучшая | Решетка | Однопроцессорная | Решетка |
| | Кластер<br>SMP | SMP<br>Решетка | Кластер<br>SMP |
| Наихудшая | Однопроцессорная | Кластер | Однопроцессорная |

Выбирая технологию хранения, следует учитывать требования к производительности и возможности восстановления, а также бюджет. В общем случае - чем выше цена, тем больше производительность и гибкость. Не забывайте учитывать и требования, предъявляемые к пропускной способности.

## Различные подходы к выбору платформы

Выбирая платформу для развертывания СУБД, многие организации отдают предпочтение системам, способным обеспечить производительность и масштабируемость на ближайшее будущее, принимая также во внимание требования к удобству администрирования и доступности. Но следует учитывать и еще два соображения.

Во-первых, азбучная истина- подольше подождешь, подешевле купишь (компьютер и все его компоненты). Согласно закону Мура, сформулированному в 1965 году сотрудником Intel Гордоном Муром (и с тех пор неоднократно доказанному на практике), вычислительная мощность процессоров удваивается каждые 1,5-2 года. Сегодня такое удвоение обеспечивается повышением тактовой частоты и реализацией нескольких ядер.

Постоянное снижение цен и увеличение производительности - это непреложный факт компьютерной индустрии. Но как им воспользоваться при планировании стратегии развертывания системы в вашей организации?

Покупайте то, что вам нужно, тогда, когда это нужно, и планируйте передачу устаревшего оборудования для других нужд организации, когда оно перестает отвечать требованиям конкретного приложения. Например, сервер, который сегодня обслуживает отдел, завтра может стать веб-сервером. Если развернута решетка, то в составе имеющейся системы можно использовать и старое оборудование.

И не забывайте о последствиях модернизации оборудования, особенно процессоров, на платформах, где решетка не реализована. В SMP-системах требуется, чтобы на всех узлах работали одинаковые процессоры, поэтому при модернизации одного узла придется модернизировать и все остальные. В какой-то момент производитель порекомендует заменить систему целиком, поскольку и другие аспекты (например, память и технология построения шины ввода/вывода) были усовершенствованы с учетом возросших возможностей новых процессоров.

Grid-вычисления выглядят очень соблазнительно, поскольку в решетку можно включать новые машины по мере их приобретения. Появление в версии Oracle Database 10g средств автоматической настройки и улучшенного администрирования и их дальнейшее развитие в Oracle Databasellg сделали grid-вычисления более практичными, так как отпала необходимость вручную настраивать различные параметры.

# Лекция 13. Распределенные данные и распределенная база данных Oracle.

## Доступ к нескольким базам данных как к единой сущности

Иногда пользователям требуется запрашивать или манипулировать данными, находящимися в нескольких базах под управлением СУБД Oracle или не только Oracle. В этом разделе мы опишем ряд приемов и архитектур, позволяющих работать с данными в таком распределенном окружении.

## Доступ к распределенным данным в нескольких базах Oracle

Уже много лет корпорация Oracle предлагает доступ к распределенным данным, хранящимся в базах Oracle на разных серверах, или *узлах*. Пользователям необязательно знать, где именно находятся данные в распределенной базе. Доступ к таблице производится по уникальному идентификатору. Администратор может создать простые идентификаторы, так что данные, находящиеся в таблице Oracle на отдельной машине, будут казаться пользователям частями единой логической базы.

Разработчик может описать на языке SQL соединения между разными базами с помощью связей (links) баз данных. За счет этих связей и образуется распределенная база данных. Команда

```
CREATE PUBLIC DATABASE LINK employees.northpole.bigtoyco.com
```

создает путь к удаленной базе данных, содержащей таблицу с данными о работниках компании Bigtoyco's North Pole. Любое приложение или пользователь, подключившийся в локальной базе данных, может обратиться и к удаленной базе North Pole, указав в SQL-команде глобальное имя доступа (`employees.northpole.bigtoyco.com`). Подсистема Oracle Net (прежние названия Net8 или SQL*Net) прозрачно реализует взаимодействие с удаленной базой данных по сетевому протоколу.

Хотя связь баз данных делает доступ к данным прозрачным для пользователей, сервер Oracle все равно должен обрабатывать взаимодействия с распределенными базами особым образом. Мы вкратце рассмотрим, чем запросы и обновления в распределенной базе отличаются от обычной. Если в запросе участвуют распределенные данные, то ваша главная забота - оптимизировать их извлечение. За производительность запросов к одной базе чаще всего отвечает оптимизатор по стоимости. В версии Oracle7 в него была добавлена возможность глобальной оптимизации с учетом распределенных баз данных. Например, оптимизатор по стоимости учитывает индексы, имеющиеся в распределенных базах, чего оптимизатор по синтаксису делать неумеет. Кроме того, оптимизатор по стоимости принимает во внимание статистику, хранящуюся в удаленных базах. В версии Oracle8i появилась возможность выполнять операции над множествами и соединения на тех узлах, которые обеспечивают максимальную производительность, минимизируя при этом объем данных, пересылаемых между системами. Начиная с версии Oracle Database 10g оптимизатор по стоимости стал единственным рекомендованным оптимизатором как для локальной, так и для распределенной базы данных.

Если пользователь хочет записать данные в распределенную базу, задача немного усложняется. Мы уже говорили, что транзакция - это атомарная логическая единица работы, как правило, состоящая из одной или нескольких SQL-команд. Эти команды записывают данные в базу и должны быть зафиксированы или откачены все сразу. В распределенных транзакциях участвуют несколько серверов базы данных. Когда такая транзакция фиксируется командой COMMIT, Oracle применяет протокол двухфазной фиксации, гарантирующий целостность инепротиворечивость транзакции во всех системах.

## Доступ к базам данных других производителей

Семейство продуктов Oracle Transparent Gateways (рис. 13.1) предоставляет пользователям доступ к базам данных сторонних производителей с помощью диалекта Oracle SQL. Команды этого диалекта автоматически транслируются на диалект SQL целевой СУБД, поэтому приложения, написанные для Oracle, могут работать и с другими базами данных. Можно также записывать команды в «родном» синтаксисе целевой СУБД, отправляя их без какой-либо трансляции. Типы данных Oracle, например NUMBER, CHAR или DATE, преобразуются в типы данных целевой СУБД. Для объектов, хранимых в целевой базе, в словаре данных Oracle имеются специальные представления. Как и базы данных Oracle, гетерогенные базы можно связывать для организации распределенной базы. Шлюзы можно развернуть в двухуровневой архитектуре в составе сервера базы данных или на промежуточном уровне (в составе Oracle Application Server).

Есть четыре основных способа организации связи между базами данных.

*Интерфейс Open Database Connectivity*
: Обобщенные интерфейсы ODBC и OLE DB бесплатно поставляются вместе с СУБД Oracle. В состав продукта Open Systems Gateways входят драйверы для Informix, Microsoft SQL Server, Sybase и других СУБД на платформах UNIX и Windows. Эти интерфейсы и шлюзы

пользуются входящими в состав СУБД Oracle службами Heterogeneous Services, определяющими оптимальную стратегию доступа к удаленной базе. Кроме того, начиная с версии Oracle Database 10g

***Рис. 13.1. Типичная конфигурация и использование Transparent Gateways***

опция OLAP Option предлагает OLE DB для OLAP (ODBO), предоставляя доступ к данным из различных инструментов бизнес-анализа.

*Прозрачные шлюзы (Transparent Gateways)*

Прозрачные шлюзы есть для десятков сторонних источников данных. Шлюз Mainframe Integration Gateways дает доступ к СУБД DB2 на больших ЭВМ. Шлюз Enterprise Integration Gateways предназначен для доступа к системе IBM AS/400 и по протоколам, применяемым в архитектуре IBM Distributed Relational Database Architecture (DRDA). Наконец, для ряда других источников Oracle предлагает семейство шлюзов EDA/SQL Gateways. Производительность Transparent Gateway была улучшена в версии Oracle8 путем переноса служб Heterogeneous Services из этого слоя в ядро СУБД. Дальнейшие улучшения были связаны с реализацией многопоточности в этих службах в версии Oracle8i с поддержкой многопоточного агента в версии Oracle9i и распараллеливанием извлечения данных из баз сторонних производителей в Oracle Database llg. В Oracle Database 10g появилась поддержка вызова удаленных функций сторонних баз данных в командах SELECT. В Oracle Database llg были добавлены шлюзы для соединения с источниками Adabas, IMS и VSAM.

*Процедурные шлюзы (Procedural Gateways)*

Процедурные шлюзы реализуют удаленные вызовы процедур в приложениях, построенных над другими источниками данных. Шлюз для APPC (стандартного протокола удаленного вызова процедур в системах IBM) используется, когда приложению Oracle необходим процедурный доступ к приложениям, работающим с базами CICS, DB2, IMS, VSAM и другими источниками данными на больших ЭВМ, или для обращения к большой ЭВМ по протоколу SNA LU6.2. Процедурный шлюз для IBM MQSeries позволяет обмениваться сообщениями с приложениями на основе технологии очередей сообщений MQSeries. Оба шлюза включены в продукт Oracle Enterprise Integration Gateways.

*Менеджер доступа (Access Manager)*

Менеджер доступа предоставляет доступ к базе данных Oracle из приложений, работающих на другой платформе. Oracle Access Manager для AS/400 размещается в системе AS/400 и позволяет написанным для нее приложениям на языке RPG, C или COBOL обращаться к базе данных Oracle на любой платформе. Для обращений можно использовать стандартный ANSI SQL или диалекты языков DML и DDL, принятые в Oracle. Поскольку язык PL/SQL также поддерживается, приложения для AS/400 могут вызывать хранимые процедуры Oracle. Для сетевого соединения используются протоколы TCP/IP и LU6.2 (через Oracle Net). Менеджер Oracle Access Manager для AS/400 включен в состав продукта Oracle Enterprise Integration Gateways.

## Двухфазная фиксация

Одна из самых сложных проблем распределенных баз данных - гарантия одинакового уровня целостности данных при обновлении разных баз. Транзакция, которая записывает данные в несколько баз, по необходимости передает информацию по сети, поэтому она более подвержена потере информации, чем в случае исполнения экземпляром Oracle на одной машине. А поскольку транзакция обязана гарантировать, что все операции записи выполнены, такая повышенная нестабильность может негативно сказаться на целостности данных.

Стандартное решение этой проблемы - разбиение процедуры фиксации транзакции на две фазы, в которых стороны обмениваются сообщениями; потому и протокол называется *двухфазной фиксацией* (two- phase commit). Главная СУБД сначала опрашивает всех участников транзакции, выясняя, готовы ли они к работе; если да, им посылаются транзакционные изменения - пока предварительно. Во второй фазе, если все участники подтверждают получение сообщений, то изменения фиксируются. Если какой-то узел, участвующий в транзакции, не подтверждает получение изменений, то транзакции на всех узлах откатываются в первоначальное состояние.

Например, если транзакция охватывает базы данных на машинахА, *В* и С, то на первой фазе каждой из них посылается транзакционное обновление. Если все машины подтвердят получение, то во второй фазе обновления будет выполнена команда COMMIT. Отделяя передачу данных для обновления от самой операции COMMIT, протокол двухфазной фиксации уменьшает вероятность нарушения целостности распределенных данных. Для сравнения представьте, что произошло бы, если, как и при однофазной фиксации, вместе с информацией о транзакционном обновлении посылалась бы команда COMMIT. В этом случае невозможно было бы узнать, дошло ли обновление до всех машин, поэтому любая ошибка при доставке привела

бы к рассогласованности данных. Поскольку вероятность утраты обновления на какой-то машине, участвующей в распределенной транзакции, резко возрастает, необходимо прибегать к протоколу двухфазной фиксации. Конечно, из-за обмена дополнительными сообщениями двухфазная фиксация занимает больше времени, чем обычная, однако только так можно сберечь самое дорогое - целостность данных.

## Мониторы обработки транзакций

В 1991 году группа разработки стандартов X/Open определила открытый интерфейс, с помощью которого мониторы обработки транзакций (transaction processing, TP) могли взаимодействовать с XA-совмес- тимыми менеджерами ресурсов, например, с СУБД Oracle и другими.

На рынке имеется несколько популярных TP-мониторов, включая BEA Tuxedo, а также CICS и Encina, поставляемые IBM.

В версию Oracle8/ для Windows NT был включен компонент Oracle Manager для Microsoft Transaction Server (MTS). С тех пор корпорация Microsoft перешла с архитектуры COM на .NET. В версии Oracle9i Release 2 добавилась также поддержка .NET, что позволило транзакционным приложениям для .NET использовать Oracle в качестве менеджера ресурсов.

Мы уже касались роли TP-мониторов в оперативной обработке транзакций. Среди прочего, TP-монитор призван гарантировать надлежащее завершение транзакций, охватывающих несколько приложений и ресурсов. Как отмечалось выше, СУБД Oracle самостоятельно поддерживает протокол двухфазной фиксации распределенных транзакций; раньше эта задача целиком возлагалась на TP-монитор. В настоящее время автономные мониторы транзакций редко используются для управления рабочей нагрузкой (рис. 13.2), поскольку эти механизмы встраиваются в приложения, работающие на промежуточном уровне.

Применение мониторов обработки транзакций оправданно, прежде всего, в следующих двух случаях:

- Перенос унаследованных приложений (обычно написанных на языке COBOL для подсистемы CICS на большой ЭВМ) на CICS на платформе UNIX или Windows NT.
- Необходимость двухфазной фиксации транзакций, охватывающих Oracle и другие XA-совместимые СУБД.

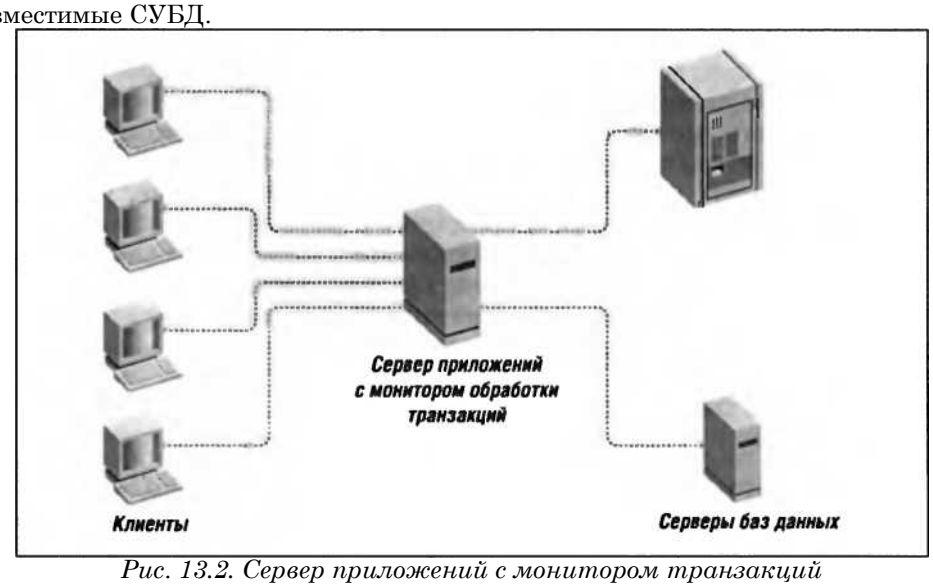



*Рис. 13.2. Сервер приложений с монитором транзакций*

## Перенос данных между распределенными системами

В предыдущем разделе обсуждалась организация нескольких серверов баз данных в единую (с точки зрения пользователей) логическую базу данных. Но иногда содержимое базы данных необходимо дублировать и перемещать на другие системы:

- если наличие локальных данных позволяет решить проблему недостаточной пропускной способности сети или устранить конкуренцию за системные ресурсы;
- если мобильные пользователи могут забирать данные с собой и работать без доступа к сети;
- если дополнительные копии базы данных позволяют достичь более высокого уровня надежности, поскольку копию можно использовать при выходе из строя основной базы.

Во многих реализациях grid-вычислений для разделения ресурсов между узлами решетки также требуется реплицировать данные между различными серверами.

Самая серьезная проблема, с которой сталкиваются пользователи идентичных или сходных баз данных, - как синхронизировать изменяющиеся со временем данные на всех серверах. Когда некий пользователь вставляет, обновляет или удаляет данные в одной базе, нужно каким-то способом передать новые данные во все остальные базы. Кроме того, следует учитывать, что изменения, внесенные в одну копию базы, могут конфликтовать с изменениями, внесенными в

другую, и тогда возникают проблемы с целостностью данных.
Для разрешения этой ситуации Oracle предлагает несколько стратегий. В версии Oracle9i Release 2 все они объединены в компонент Oracle Streams. Каждая стратегия имеет свои особенности, которые мы рассмотрим в следующих разделах.

## Технология Advanced Replication

Копирование таблиц из одной базы данных Oracle в другую в распределенной системе и сопутствующие административные действия называются *репликацией*. Изменения, произведенные в любой базе, автоматически распространяются по всем остальным. Речь идет об изменении как самих данных, так и схемы базы. Репликацию часто организуют для ускорения доступа к удаленным данным или для восстановления данных в случае полного выхода из строя основного центра обработки. Технология Oracle Advanced Replication поддерживает синхронную и асинхронную репликацию. Кроме того, Oracle поддерживает гетерогенную репликацию в СУБД DB2 с помощью служб Replication Services, включенных в продукт Mainframe Integration Gateways.
Службы репликации появились в СУБД Oracle уже давно, но продолжают совершенствоваться. В Oracle8 выполнение триггеров репликации перенесено в ядро СУБД, а также реализовано автоматическое распараллеливание репликации данных для повышения производительности. В Oracle8i добавился механизм запуска репликации при изменениях в отдельных строках или столбцах таблицы. В Oracle9i появилась возможность репликации объектных типов данных и многоуровневых обновляемых материализованных представлений. В версии Oracle9i Release 2 стала возможна репликация на основе журналов посредством технологии Oracle Streams. Хотя в современных версиях СУБД все еще поддерживаются средства репликации предыдущего поколения (Advanced Replication), в новых разработках мы рекомендуем применять для репликации технологию Streams. Ради полноты, прежде чем перейти к Streams, мы все же рассмотрим простые методы репликации и опишем некоторые возможности Advanced Replication.

*Асинхронная репликация* подразумевает локальное сохранение изменений с их последующей передачей в удаленную систему. Иногда реплицируются мгновенные снимки главной базы данных, допускающие только чтение, а иногда - обновляемые снимки, которые можно модифицировать даже при отсутствии связи с главной базой.

В редакции Oracle Standard Edition допускается только одна главная база, изменения в которой реплицируются в подчиненные базы. В редакции Enterprise Edition главных баз может быть несколько и обновлять разрешается любую из них. Обновления должны быть *синхронизированы*, то есть обновление не считается завершенным, пока не обновятся все участвующие базы; в противном случае могли бы возникнуть неразрешенные конфликты.

Конфликт происходит, когда в течение одного интервала репликации несколько систем обновляют один и тот же элемент данных. Изменения распространяются с помощью отложенных удаленных вызовов процедур (RPC), выполняемых при возникновении определенных событий или в те моменты времени, когда имеется связь или стоимость передачи данных минимальна.

Для разрешения конфликтов при репликации можно использовать одну из нескольких автоматических процедур, включенных в редакцию Enterprise Edition. Администратор просто выбирает стратегию, которую хочет использовать в конкретном плане репликации. Для обновлений, затрагивающих один столбец или группу столбцов, имеются следующие стандартные варианты разрешения конфликтов.

*Перезаписать и отбросить*
: Применяется, когда есть единственная главная (исходная) база, и хранящиеся в ней данные используются для обновления текущих значений в репликах.

*Минимальное и максимальное значение*
: Новое значение в исходной базе сравнивается с текущим значением в реплике и замещает его, если оказывается соответственно меньше или больше.

*Значение с меньшей или большей временной меткой (требует назначения столбца типа DATE)*
: Если есть несколько новых значений, то выбирается то из них, для которого временная метка в указанном столбце типа DATE той же строки соответственно минимальна или максимальна.

*Сумма или среднее для группы столбцов, содержащих числовые данные*
: В случае суммы вычисляется разность между новым и старым значением в исходной базе и результат прибавляется к текущему значению в реплике.

  В случае среднего вычисляется сумма текущего значения в исходной базе и нового значения в реплике, результат делится на 2 и становится новым значением.

*Приоритетные группы и приоритет узла*
: Если столбцам назначены приоритеты и обнаруживается несколько новых значений, то значения с узла с более высоким приоритетом служат источником обновлений для столбцов узла с меньшим приоритетом.

Процедуры разрешения конфликтов уникальности применяются, когда возникают конфликты в значениях первичного или уникального ключа в разных репликах базы данных. В СУБД встроены следующие процедуры такого рода:

*Дописать имя базы данных к повторяющемуся значению*
: Дописывает глобальное имя исходной базы данных в конец значения в реплицируемом столбце.

*Дописать порядковый номер к повторяющемуся значению*
: Дописывает сгенерированный порядковый номер к значению в столбце.

*Отбросить повторяющееся значение*
: Удаляет из исходной базы строку, послужившую причиной ошибки.

Если ни одна из стандартных процедур разрешения конфликтов вас не удовлетворяет, можете написать и зарегистрировать собственную.

### Администрирование Advanced Replication

Управлять репликацией можно с помощью Oracle Enterprise Manager. Этот централизованный интерфейс позволяет администратору указать объекты базы данных, которые следует реплицировать, задать расписание репликации, установить и устранить причины неполадок и просмотреть очередь отложенных транзакций на каждом сервере. Вочереди отложенных транзакций находятся транзакции, которые должны быть реплицированы в подчиненные базы и выполнены там.

Например, для организации типичной симметричной репликации нужно сначала определить главные группы (master group), а также таблицы и объекты, подлежащие репликации.

Далее описано соединение с системой, где хранятся главные определения (master definition site) и создается одна или несколько главных групп для репликации таблиц и объектов в несколько главных баз. Затем с каждой главной группой ассоциируются процедуры разрешения конфликтов при репликации таблиц и объектов. Наконец, назначаются необходимые привилегии пользователям приложений, обращающихся к данным на различных серверах.

## Технология Advanced Queuing

В 1980-х получило рапространение *ПО промежуточного уровня*, *ориентированное на обработку сообщений* (message-oriented middleware, MOM). В нем для передачи информации между системами используются *сообщения*. В этом случае не возникают накладные расходы на двухфазную фиксацию, поскольку само MOM гарантирует доставку всех сообщений. Такие продукты, как IBM MQSeries, хранят управляющую информацию (пункт назначения, срок хранения, приоритет и получателей), а также само содержимое сообщения в очереди, реализованной в виде файла. Гарантируется, что сообщение будет оставаться в очереди, пока пункт назначения не окажется доступным и сообщение не будет туда доставлено.

Подсистема Advanced Queuing (AQ), впервые появившаяся в редакции Oracle8 Enterprise Edition, а теперь являющаяся частью Oracle Streams, позволяет хранить очереди сообщений внутри реляционной базы Oracle. Очереди представлены специальными таблицами, поддерживающими операции *постановки в очередь* создателем сообщения и *извлечения из очереди* потребителем. Сами сообщения, которые могут быть неструктурированными или структурированными (в виде объектов Oracle), представлены строками таблицы. Сообщения могут храниться в обычных очередях для нормальной обработки или в очередях исключений, куда попадают, если по какой-то причине не могут быть извлечены.

### Создание очередей и управление ими

Очереди создаются с помощью команд на языке PL/SQL или с применением Java API. Для создания очереди администратор должен выполнить следующие действия:

1. Создать таблицу для очереди.
2. Создать и назвать очередь.
3. Указать, предназначена ли очередь для хранения обычных сообщений или исключений.
4. Указать, как долго сообщение может оставаться в очереди: неопределенное время, фиксированное время, пока интервал между повторными попытками доставки не превысит заданный порог или пока не будет исчерпано заданное количество попыток.

Администратор может запускать и останавливать очередь, он же назначает пользователям привилегии для работы с очередью или отзывает их.

Производитель сообщений указывает имя очереди, параметры постановки в очередь, свойства сообщения и его полезную нагрузку, а затем передает все это агенту создателя. Агенты потребителей следят за появлением сообщений в одной или нескольких очередях, извлекают их оттуда и передают потребителям. Уведомление о наличии в очереди сообщений можно получить, зарегистрировав процедуру обратного вызова с помощью интерфейса Oracle Call Interface или вызвав функцию прослушивания, которая может использоваться для мониторинга нескольких очередей.

Так как сообщения хранятся в базе данных, в распоряжение администратора предоставляются различные средства управления сообщениями. Можно отслеживать весь путь прохождения, поскольку вместе с сообщением хранится его история: местонахождение и состояние сообщения, перечни посещенных узлов и предыдущих получателей. Сообщения, не дошедшие до адресата в течение заданного времени, перемещаются в очередь исключений и могут быть протрассированы. Успешно доставленные сообщения после получения можно сохранить для дополнительного анализа, например, чтобы узнать время постановки и извлечения из очереди. Поскольку сообщения могут быть взаимосвязаны (например, некоторое сообщение может быть отправлено в ответ на успешную обработку двух других сообщений), их сохранение бывает полезно для анализа цепочки событий.

В версии Oracle Database 10g Release 2 появился механизм организации очередей без постоянного хранения. Он позволяет повысить производительность в случаях, когда возможности, обеспечиваемые хранением сообщений в базе данных, не нужны.

Oracle Enterprise Manager предоставляет следующие средства управления очередями:

- создание, удаление, запуск и остановка очередей;
- добавление и удаление подписчиков;
- составление расписания передачи сообщений из локальной очереди в удаленную;
- отображение статистики очереди: средняя длина, количество сообщений в состоянии ожидания, количество сообщений в состоянии готовности и количество сообщений с истекшим сроком хранения.

В версии Oracle9i технология AQ пополнилась несколькими новыми возможностями:

- Обмен сообщениями в формате XML по протоколу HTTP для преодоления межсетевых экранов; запросы можно посылать по протоколу Internet Document Access Protocol (IIAP), основанному на XML.
- С помощью служб Dynamic Services можно определять политики и службы AQ.
- Можно определять агенты AQ и управлять ими с помощью каталога Oracle Internet Directory (OID).

Начиная с версии Oracle9i AQ включает встроенный механизм трансформации сообщений для PL/SQL и XSLT. Есть также шлюзы для обмена сообщениями с другими системами, например MQSeries и TIBCO.

## Средства публикации/подписки

В редакции Oracle8i Enterprise Edition в Advanced Queuing были добавлены средства организации публикации/подписки. Как показано на рис. 13.3, *публикатор* помещает сообщения в очередь, а *подписчик* извлекает сообщения. Публикатор и подписчик работают с очередью независимо, один ничего не знает о существовании другого. Публикатор решает, когда, как и что публиковать, а подписчики изъявляют интерес в тех или иных сообщениях. Подписаться можно на определенную тему или содержимое (с помощью правил фильтрации). Зарегистрировав функции обратного вызова, подписчик может получать уведомления асинхронно.

Технологию Advanced Queuing и ее встроенные средства публикации иподписки можно использовать для дополнительного уведомления

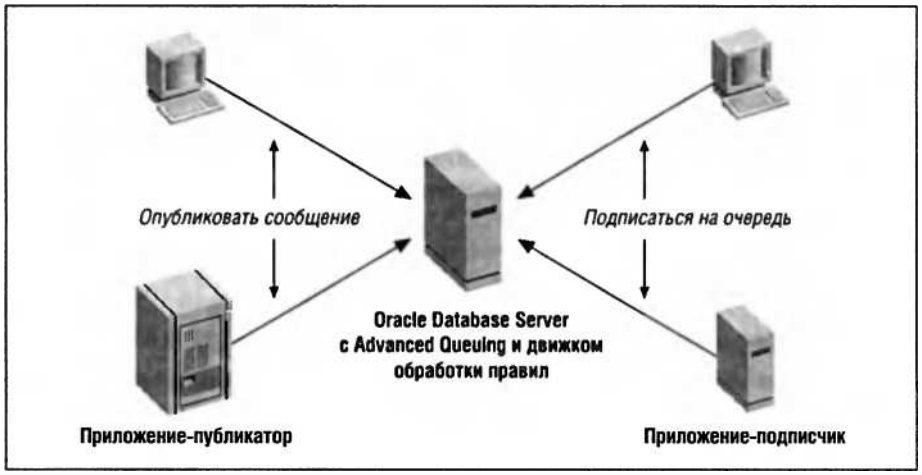



*Рис. 13.3. КонфигурацияАйоапсей Queuing для приложений публикации/подписки*

о событиях базы данных, упростив тем самым управление базой данных или бизнес-приложениями. Можно публиковать и подписываться на события манипулирования данными (вставка, обновление, удаление) и на системные события (запуск, останов и так далее). Например, можно написать приложение, которое будет автоматически информировать подписчика об отгрузке продукции определенным особо значимым заказчикам; тогда подписчик будет знать, что нужно начать отслеживать процесс доставки, уведомив заказчика о том, что товары находятся в пути.

В версии Oracle Database 1lg появилось несколько улучшений в работе сервера сообщений, повышающих его производительность и надежность.

## Подсистема Oracle Streams

В версию Oracle9i Release 2 была включена подсистема Oracle Streams, объединившая функции Advanced Replication и Advanced Queuing в одном семействе продуктов. Также был добавлен механизм разделения данных и событий в пределах одной базы данных или между разными базами. Streams позволяет распространять изменения с помощью процедуры сбора и применения (capture-and-apply), включающей и механизм сбора данных об изменениях в Oracle. Изменения можно передавать между экземплярами Oracle, из Oracle в другую БД (через Transparent Gateways), а также из сторонних СУБД в Oracle (с помощью шлюзов сообщений в сочетании со специальным кодом для стороннего источника данных, позволяющего собирать сведения об изменениях). Streams может извлекать информацию об изменениях данных (команды DML) и схемы (команды DDL) из журналов или синхронно перехватывать изменения данных, после чего помещает изменения в очередь. Порядок обработки сообщений определяется заданными пользователем «правилами применения».

После того как информация об изменениях получена путем анализа журнала или синхронного перехвата операций изменения строк, фоновый процесс создает логическую запись об изменении (logical change record, LCR). LCR и сообщения о пользовательских событиях помещаются в очередь Streams. Затем эти сообщения передаются из исходной очереди в целевую, после чего фоновый процесс извлекает их и применяет к целевой базе данных. Начиная с версии Oracle Database 10g поддерживается пакетный сбор изменений, их помещение и извлечение из очереди.

Кроме того, начиная с версии Oracle Database 10g можно сконфигурировать Steams так, чтобы уведомление об изменении в базе данных (Database Change Notification) посылалось по электронной почте, по протоколу HTTP или в PL/SQL-процедуру. Тем самым клиенту можно отправлять уведомления о любых изменениях в результирующем наборе, полученном в ответ на запрос. В Oracle Database llg этот механизм усовершенствован - теперь можно уведомлять об изменениях в отдельных строках, а не посылать общее уведомление, когда изменилась какая-нибудь строка в результирующем наборе.

В версии Oracle Database 10g Release 2 подсистемой Streams можно управлять из Oracle Enterprise Manager. Появился инструмент, упрощающий миграцию с Advanced Replication на Streams.

## Streams и grid-вычисления

Подсистема Oracle Streams - ключ к реализации grid-вычислений. По своей природе решетка может включать широко распределенные данные, пользователей и платформы. Streams обеспечивает перемещение данных (когда и куда требуется), а также общий доступ к сообщениям, уведомление или вызов пользовательских процедур по событиям, подписку на

сообщения и изменения в базе данных и интероперабельность с другими платформами. Такой подход может разгрузить систему, передав часть обработки репликам баз данных за счет создания оперативных складов данных. Аможно вместо этого создать реплики и применять произведенные в них изменения в рабочих базах, быть может, после тех или иных трансформаций данных.

В версии Oracle Databasellg Streams может искать команды DML и DDL в оперативных журналах, что уменьшает задержку при распространении изменений между экземплярами RAC-кластеров. При этом Streams работает на каком-то одном экземпляре RAC, назначенном главным для очередей и процессов. Можно также определить вспомогательный экземпляр для повышения уровня доступности.

Streams можно также использовать для миграции базы данных в решетку или на более свежую версию Oracle. После того как новая база данных установлена, а старая еще работает в промышленном режиме, с помощью Streams можно собирать изменения в старой базе и накатывать их на новую до тех пор, пока процедура миграции не завершится полностью.

## Переносимые табличные пространства

В предыдущих разделах мы говорили, в основном, о разделении «живых» данных между распределенными базами, когда ставится задача распространять изменения в реальном масштабе времени. Механизм *переносимых табличных пространств* позволяет ускорить перенос целых табличных пространств при условии, что они в это время не обновляются.

Переносимые табличные пространства появились в редакции Oracle8i Enterprise Edition с целью ускорить перемещение табличных пространств между экземплярами базы данных. Раньше приходилось экспортировать табличное пространство из исходной базы и импортировать в конечную (или выгрузить и загрузить). Для перемещения переносимого табличного пространства достаточно воспользоваться простыми командами передачи файлов, например *ftp*. Перед тем как копировать и перемещать табличное пространство, его нужно сделать доступным только для чтения во избежание случайных изменений. До переноса нужно экспортировать информацию из словаря данных в исходной базе и импортировать ее в конечную.
Вот несколько ситуаций, когда бывает удобно воспользоваться переносимыми табличными пространствами:

- быстрое копирование табличных пространств из корпоративного хранилища данных на витрину данных;
- копирование табличных пространств из оперативных систем на оперативный склад данных для последующего использования при генерации консолидированных отчетов;
- публикация табличных пространств для распространения на компакт-дисках;
- использование резервных копий для быстрого восстановления табличного пространства на конкретный момент времени.

Раньше требовалось, чтобы размер блока в исходной и конечной базах был одинаковым, но в версии Oracle9i это ограничение снято. В версии Oracle Database 10g уже не требуется, чтобы обе базы данных были развернуты в одной и той же операционной системе.

# Лекция 14. Расширенные типы данных в Oracle.

## Объектно-ориентированная разработка

Объектно-ориентированный подход к разработке программного обеспечения означает переход от привычной методики конструирования вычислительных процедур для работы с наборами данных к моделированию бизнес-процессов. Построение программных компонентов, которые моделируют бизнес-процессы и предоставляют документированные интерфейсы, повышает эффективность работы программиста и открывает путь к реализации более гибких стратегий развертывания. К тому же написанные таким образом приложения проще модифицировать, когда бизнес выдвигает новые требования. Кроме того, поскольку моделирование отражает реальное функционирование бизнеса, производительность приложения может возрасти за счет того, что построенные объекты не нуждаются в сложных манипуляциях для подстраивания к поведению представляемых ими бизнес-процессов.
В Oracle принят эволюционный подход к объектной технологии, основанный на *абстрагировании данных*, то есть создании определяемых пользователем типов данных в виде объектов и наборов, расширяющих реляционную СУБД. Включение в версию Oracle8i объектов и средств расширяемости позиционирует Oracle как объектно-реляционную СУБД.
Этот подход дополняется поддержкой языка Java. Виртуальная Java- машина JVM (прежнее название JServer) интегрирована с СУБД. Она поддерживает создание и выполнение Java-компонентов, а также написание на Java хранимых процедур и триггеров.

# Объектно-реляционные возможности

В этом разделе мы опишем основные объектно-реляционные возможности Oracle.

## Объекты в Oracle

В Oracle объекты создаются как повторно используемые компоненты, представляющие реальные бизнес-процессы. Объекты, созданные с помощью средств, собирательно названных Objects and Extensibility, играют ту же роль, что таблица в стандартной реляционной модели: объект - это шаблон для создания отдельных экземпляров объекта, которые являются аналогами строк в таблице. Объект создается с помощью конструкторов, которые можно вызывать из SQL or PL/SQL.

Описание *объекта* включает имя, один или несколько атрибутов и методы. *Атрибуты* моделируют структуру и состояние некоей реальной сущности, а *методы* моделируют операции, выполняемые этой сущностью. Методы - это функции или процедуры, обычно написанные на языке PL/SQL или Java, а иногда (если они внешние) - на других языках, например C. Методы образуют интерфейс между объектом и его программным окружением. Метод полностью идентифицируется именем содержащего его объекта и именем метода этого объекта. У метода может быть один или несколько *параметров* для передачи ему данных из вызывающего приложения.

Например, в виде объекта можно представить заказ на покупку. В качестве атрибутов можно взять номер заказа, название и адрес поставщика, адрес отгрузки и набор товаров (для каждого из которых указываются количество и цена). У такого объекта могут быть методы для добавления товара в заказ, удаления товара из заказа и получения полной стоимости заказа.

Хранить объекты можно как строки таблицы или как значения в столбце. У каждого объекта-строки имеется уникальный идентификатор объекта (object identifier, OШ), генерируемый Oracle. На объекты-строки можно ссылаться из других объектов или из реляционных таблиц. Такие ссылки представляются типом данных REF. Для объектов- столбцов Oracle добавляет скрытые столбцы, где хранятся атрибуты объекта.

*Объектные представления* (object views) - это способ создания виртуальных объектных таблиц из данных, хранящихся в столбцах реляционных таблиц. Такие представления могут также включать атрибуты других объектов. Для создания объектного представления нужно определить тип объекта, написать запрос, определяющий отображение между данными и таблицами, в которых хранятся атрибуты этого типа, и указать уникальный идентификатор объекта. При сохранении данных в реляционной таблице этот уникальный идентификатор становится первичным ключом. Такая реализация означает, что приемы объектно-ориентированного программирования можно применять, не преобразуя реляционные таблицы в объектно-реляционные. Но производительность при этом окажется не оптимальной, потому что данные, представляющие атрибуты объекта, могут находиться в нескольких разных таблицах. Поэтому в перспективе имеет смысл все же преобразовать реляционные таблицы в объектные.

Об объектах, имеющих одинаковые атрибуты и методы, говорят, что они принадлежат одному и тому же типу данных, или *классу*. Например, внутренние и внешние заказы на покупку могут принадлежать тому же классу, что и просто заказы на закупку. *Типы коллекций* моделируют множества объектов одного и того же типа в виде массивов переменной длины (VARRAY), если коллекция ограничена и упорядочена, или в виде вложенных таблиц, если коллекция не ограничена и не упорядочена. Если длина коллекции меньше 4000 байт, то она хранится непосредственно в таблице базы данных; в противном случае - как большой двоичный объект (BLOB) в отдельном сегменте, считающемся «вынесенной» (out-of-line) памятью. Строки вложенных таблиц хранятся в отдельной таблице, идентифицируемой скрытым полем NESTED_TABLE_ID. Обычно тип VARRAY используется, когда коллекция извлекается целиком, а вложенные таблицы - когда коллекция может извлекаться по запросу, особенно если коллекция велика, а нужна только ее часть.

Вызывать методы объекта из приложения можно с помощью SQL, PL/ SQL, Pro*C/C++, Java, OCI и транслятора типов (Oracle Type Translator, OTT). OTT отображает типы объектов для использования на стороне клиента, генерируя заголовочные файлы с объявлениями структур на C и индикаторами. Для повышения производительности разработчик приложения может использовать клиентский кэш объектов.

*Наследованием* (inheritance) называется использование одного класса объектов как основы для другого, более специфичного класса. Это одно из самых мощных средств объектно-ориентированного проектирования. Дочерний класс наследует все атрибуты и методы своего родителя, добавляя собственные атрибуты и методы для расширения возможностей последнего. Своей мощью механизм наследования обязан тому факту, что любое изменение в родительском

классе автоматически распространяется на все его потомки. В объектно-ориентированном проекте может быть произвольное количество уровней наследования.

*Полиморфизмом* называется возможность переопределить (override), или заместить (перегрузить), некую операцию родительского класса вдочернем классе, который предоставляет собственную реализацию метода. Если метод был замещен в дочернем классе, то изменения этого метода в его родителе не сказываются на дочернем классе и его потомках. В примере с заказом на покупку (рис. 14.1) заказы от поставщиков, работающих по контракту и без контракта, наследуют атрибуты и методы внешних заказов на покупку. Однако процедура размещения заказа может быть полиморфной, так как для закупки у поставщиков, с которыми нет контракта, может потребоваться дополнительное разрешение.


***Рис. 14.1. Иерархия классов, представляющих заказы на покупку***

В версии Oracle8i для объектов не поддерживались наследование и полиморфизм, хотя база данных могла использоваться для хранения объектов, а приложение на объектно-ориентированном языке, например C++ или Java, могло наделить объекты на стороне клиента недостающей функциональностью. В версии Oracle9i наследование SQL-типов было реализовано на уровне СУБД наряду с иерархиями объектных представлений, эволюцией типов, обобщенными и временными типами, индексированием по значениям функций, являющихся методами типов, и многоуровневыми коллекциями. В Oracle Database 10g добавилась поддержка удаленного доступа к данным объектных типов. BOracle Databasellg* был реализован описанный в стандарте ANSI SQL оператор разрешения области видимости при вызове метода.

### Другие средства расширяемости

Objects and Extensibility включает еще несколько средств расширяемости:

- возможность создавать новые типы индексов путем определения структуры индекса;
- возможность хранить данные индекса внутри или вне базы данных Oracle;
- возможность создавать определяемые пользователем операторы и применять их в стандартных командах SQL;
- интерфейс к оптимизатору по стоимости для расширения поддержки определяемых пользователем объектных типов и индексов.

Чаще всего объектно-реляционные средства используют разработчики, создающие расширения СУБД. В самой корпорации Oracle они применялись неоднократно, например для создания расширений Spatial и Multimedia.

## Роль Java и веб-службы

Java получил широкое признание как язык программирования, особенно веб-приложений, из-за своей переносимости и доступности на самых разных платформах.

Для разработчиков на Java, желающих использовать базу данных Oracle как серверную часть для своих приложений, корпорация Oracle поначалу включила поддержку интерфейса JDBC 3.0 в версию Oracle Database 10g, затем поддержала два наиболее распространенных способа доступа к базе данных из Java-программ - JDBC и SQLJ. Оба подхода основаны на стандартных промышленных API.

*JDBC*

Обычно применяется для динамического конструирования SQL-команд и в тех случаях, когда разработчику нужен явный контроль над взаимодействиями с базой данных.

*SQLJ*

Промышленный стандарт, обычно применяемый, когда в Java-программу встроены статические SQL-команды. SQLJ аналогичен другим прекомпиляторам Oracle в том смысле, что создает исходные файлы на языке Java, в которые включены обращения к среде выполнения SQLJ (а также дополнительные файлы профилей). Затем исходный Java-код компилируется и приложение компонуется с библиотекой времени выполнения SQLJ.

SQLJ и JDBC можно совместно применять в одной и той же программе, где встречаются как статические, так и динамические SQL-команды.

В версии Oracle9i и последующих в виртуальную Java-машину JVM (в Oracle8i она называлась JServer) были включены дополнительные средства создания компонентов и объектно-ориентированной разработки. Так, виртуальная Java-машина тесно интегрирована в ядро СУБД и поддерживает написание хранимых процедур на Java; в результате появилась возможность разрабатывать компоненты на основе технологии Enterprise JavaBeans (EJB-компоненты). Oracle Streams предоставляет Java API для доступа к службам сообщений (Java Messaging Support, JMS).

В версию Oracle Database 10g были добавлены веб-службы для активизации в базе данных операций клиентами без соединения. К средствам, поддерживаемым самой СУБД для построения веб-служб, относятся SQL, PL/SQL, встроенный Java, JDBC, HTTP-клиент и SOAP-клиент и соответствующие им средства на уровне Oracle Application Server (Java, J2EE, JDBC, HTTP, SOAP-сервер и XML). База данных может выступать в роли поставщика или потребителя веб-служб, а ее интерфейсы можно раскрыть с помощью утилиты JPublisher, которая служит для генерации классов, представляющих определяемые пользователем сущности в базе данных. Начиная с версии Oracle Database П^СУБД может служить поставщиком служб в сервисно-ориентированной архитектуре (SOA) за счет использования HTTP-сервера XDB для SOA. В виде веб-служб можно раскрывать PL/SQL-пакеты, процедуры и функции. При развертывании базы данных таким способом можно выполнять динамические запросы на языках SQL и XQuery.

## Технология EnterpriseJavaBeans

Серверные Java-компоненты называются Enterprise JavaBeans (EJB) в противоположность клиентским повторно используемым компонентам, которые называются просто JavaBeans. Развернуть EJB-компо- ненты можно на сервере базы данных или на сервере приложений Oracle Application Server. Поскольку виртуальная Java-машина тесно интегрирована с СУБД, можно применять механизмы управления памятью в системной глобальной области (SGA), тем самым обеспечивая гораздо более высокий уровень масштабируемости EJB-сервера, чем принято ожидать от большинства реализаций JVM. Например, на хранение состояния сеанса каждого клиента в JVM задействуется всего около 50-150 Кбайт памяти.

В самом первом выпуске Oracle8i поддерживался *компонент-сеанс* (session bean) - EJB-компонент, создаваемый специальным вызовом со стороны клиента и обычно существующий только на протяжении одного сеанса работы клиента ссервером. Компоненты-сеансы могут быть *без сохранения состояния* (stateless), что позволяет EJB-серверу повторно использовать их экземпляры для обслуживания различных клиентов, или *с сохранением состояния* (stateful) - такой компонент привязан к конкретному клиенту. Информация, которую компоненты-сеансы с сохранением состояния размещают в кэше базы данных, синхронизируется с базой данных в момент выполнения транзакций из JDBC или SQLJ. *Компоненты-сущности* (entity Java beans), называемые также *постоянными компонентами* (persistent beans), поскольку они не перестают существовать с завершением сеанса, в версии Oracle8i не поддерживались, но стали поддерживаться начиная с версии Oracle9i. Третий тип EJB-компонентов - *компонент, управляемый сообщениями* (message-driven bean), - предназначен для получения асинхронных сообщений от службы Java Message Services (JMS) и поддерживается в последних версиях сервера Applications Server, в которых реализована спецификация EJB 3.0.

# Встроенные и дополнительные средства расширяемости

Встроенные и дополнительные средства расширяемости Oracle позволяют решать с помощью языка SQL задачи, с которыми иначе было бы нелегко справиться в реляционной СУБД. Речь идет о манипулировании текстом, мультимедийным контентом и пространственными данными. Как правило, эти средства бывают нужны разработчикам приложений, но иногда поставляются в комплекте с приложениями, которые продают партнеры Oracle.

## Oracle Multimedia и Oracle Text

Подсистема Oracle Multimedia (прежнее название *interMedisi)* вошла вOrac1e начиная с выпуска8.1.6 версии Oracle8i. В версии Oracle9i средства этого продукта, предназначенные для работы с текстом, получили название Oracle Text. Ранее они были доступны как опции:

- средство Text Management прежде называлось ConText Option;
- служба Location Services стала развитием опции Spatial Option и поддерживает запросы о географическом местоположении и геокодирование;
- средства хранения и манипулирования изображениями ранее поставлялись как опция Image Option.

Кроме того, имеются расширения, позволяющие хранить аудио- и видеоклипы и манипулировать ими, в частности, извлекать содержимое и организовывать метаданные в виде CLOB-объекта в формате XML. Корпорация Oracle позиционирует Oracle Multimedia и Oracle Text как полезные средства для приложений, работающих с различными видами мультимедиа, поскольку в них интегрированы все основные типы данных и ассоциированные с ними функции. Oracle Multimedia и Oracle Text пользуются многочисленными базовыми типами хранения, перечисленнымивтабл. 14.1.

В версии Oracle Database Ю^разрешено хранить в LOB-объектах документы объемом до 128 терабайт. В Oracle Database llg максимальный размер объекта типа Multimedia увеличен и стал таким же, как для BLOB-объектов (от 8 до 128 терабайт). Кроме того, в этой версии Multimedia появилась новая высокопроизводительная реализация BLOB с помощью механизма Oracle SecureFiles.

*Таблица 14.L Типы хранения, используемые в Oracle Multimedia и Oracle Text*

| Вид мультимедиа | Тип хранения |
|---|---|
| Текст/ изображения | VARCHAR2 |
| | BLOB |
| | CLOB |
| | VARCHAR |
| | CHAR |
| | LONG |
| | LONG RAW |
| | Атрибут объекта |
| | Главные/подчиненные таблицы (в главной таблице хранится идентификатор текста или изображения, а в подчиненной - содержимое) |
| | BFILE |
| | URL, указывающий на содержимое DICOM |
| Аудио- и видеоклипы | BLOB |
| | BFILE |
| | URL, указывающий на содержимое |
| Координаты | VARRAY |

Подсистемы Oracle Multimedia и Oracle Text поддерживают следующие широко распространенные форматы файлов:

- Разрешается индексировать документы в форматах ASCII, Microsoft Word, Excel и PowerPoint, WordPerfect, HTML, XML и Adobe Acrobat (PDF).
- Поддерживаются аудиоформаты AU, AIFF, AIFF-C, WAV, MPEG1, MPEG2 и MPEG4.
- Поддерживаются видеоформаты Apple QuickTime 3.0, AVI, MPEG (MPEG и MP4) и видеоформат Real Networks Real (RMFF).
- Поддерживаются графические форматы BMPF, CALS, FPIX, GIFF (gif), JFIF (jpeg), PBMF, PGMF, PPMF, PPNF, PCXF (pcx), PICT, PNGF, RPIX, RASF, TGAF, TIFF и WBMP. Поддерживаются также следующие алгоритмы сжатия изображения: ASCII-кодирование, BMPRLE, DEFLATE, DEFLATE-ADAM7, FAX3, FAX4, GIFLZW, GIFLZW-INTERLACED, HUFFMAN3, JPEG, JPEG-PROGRESSIVE, LZW, LZWHDIFF, NONE, PACKBITS, PCXRLE, RAW, SUNRLE и TARGARLE.
- Начиная с версии Oracle Database llg поддерживается стандарт кодирования медицинской графической информации Digital Imaging and Communications in Medicine (DICOM) версии 3. В базе данных можно хранить однокадровые и многокадровые изображения, формы сигналов, трехмерные объемные срезы, видеосегменты и структурированные данные. Имеются методы и функции для конвертирования DICOM в JPEG, GIF, PNG, TIFF и другие форматы. Метаданные можно извлекать в виде XML-документов или определить специализированное преобразование.

Средства управления текстом в Oracle позволяют определить главную тему (сущность)

документа и сгенерировать реферат документа исходя из этой темы. В Oracle Database 10g добавлена возможность поиска по близкой тематике (NEAR) и реализована техника определения набора символов и языка документа с неизвестным содержимым. Возможен полнотекстовый поиск по словам и фразам, поиск по теме и смешанный поиск по текстовым и нетекстовым данным. В Oracle Database 10g подсистема Oracle Text позволяет индексировать столбцы типа XML- Type.

Поскольку средства управления текстом в Oracle нужны прежде всего новостным службам, публикующим в Интернете новости, соответствующие интересам пользователей, в последние версии СУБД включен алгоритм вычисления ранга популярности веб-страниц и контента. Кроме того, начиная с версии Oracle Database 10g JDeveloper предоставляет простой интерфейс, позволяющий создавать специализированные приложения для работы с текстом, в который входит генератор приложений обработки текста, генератор приложений для поиска в каталоге и мастер обучения классификаторов на примерах.

Поддержка изображений в Oracle позволяет конвертировать форматы и алгоритмы сжатия, работать с изображением на уровне пикселов и выполнять простейшие операции, например масштабирование и вырезание.

Клиент может воспроизводить аудио- и видеофайлы с помощью плееров Java Media Framework (JMF) (в версии Oracle9i подсистема Java Advanced Imaging поддерживает также просмотр изображений в JMF- плеерах). Потоковые серверы типа Real Networks Server также могут доставлять аудио- и видеоконтент по запросу.

Доступ к изображениям, аудио- и видеоматериалам, хранящимся в базе данных Oracle и в подсистеме Multimedia, возможен из программ на языках C++, Java, OCI или PL/SQL. Начиная с версии Oracle Database 10g графические объектные типы поддерживают стандарт SQL/ MM Still Image, ISO/IEC 13249-5 SQL и пакет Java Advanced Imaging (JAI) компании Sun Microsystems, предназначенный для хранения и обработки графического контента. Для доступа к данным в формате DICOM начиная с версии Oracle Database llg имеются API для Java и PL/SQL.

Изображения, аудио- и видеоматериалы, сохраненные с применением подсистемы Multimedia, можно также выкладывать на веб-сайт с помощью различных инструментов веб-публикации. Службы по управлению портальным контентом включены в Oracle Application Server, Oracle JDeveloper, а также предлагаются различными партнерами Oracle.

## Управление контентом в Oracle

В комплект продуктов Oracle Content Database Suite включены базовые службы для работы с документами, хранящимися в базе данных, и инфраструктура, необходимая для создания приложений по управлению документами. Продукт Content DB предоставляет репозиторий, а Content Server управляет документами. Эти продукты можно использовать для консолидации документов на файловом сервере, для управления политиками и процедурами работы с документами, для организации совместного доступа и работы над документами, а также в качестве репозитория контента для приложений.

В 2007 году корпорация Oracle завершила приобретение компании Stellent и стала предлагать более полную инфраструктуру и набор приложений для управления контентом под названием Universal Content Management (UCM). UCM - это система управления всем корпоративным контентом, в том числе документами, веб-контентом, цифровыми активами и учетно-отчетными материалами.

И, наконец, для управления контентом у Oracle есть еще одно приложение - система Imaging and Process Management (IP/M) для обработки графики в приложениях, ориентированных на организацию бизнес-процессов в контексте продуктов E-Business Suite, PeopleSoft и JD Edwards. Имеются модули для автоматизации обработки кредиторской и дебиторской задолженности, транспортных и командировочных расходов, а также кадровой информации.

При развертывании такой инфраструктуры часто выдвигаются жесткие требования к учету исторической информации и к безопасности. Продукт Universal Records Management (URM) предлагает унифицированный подход к организации отчетно-учетной информации и централизованное задание политик сохранения для управления корпоративным контентом. Доступ к репозиториям контента осуществляется с помощью адаптеров. Например, URM в сочетании с адаптером Content DB может заменить прежний продукт Oracle Records DB.

Можно развернуть систему управления правами на информацию Information Rights Management (IRM), которая будет выпускать ключи Secure Keys на сервере IRM Server, управлять доступом и обеспечивать защиту секретного контента. IRM позволяет централизованно задавать политики, производить аудит, мониторинг, шифрование и отзыв прав.

## Подсистема Oracle Ultra Search

Подсистема Ultra Search позволяет искать информацию в текстах, хранящихся в базах данных Oracle, других СУБД, доступных по ODBC, репозиториях Oracle Portal, почтовых IMAP-серверах, HTML-доку- ментах на веб-серверах и в других файлах. В версии Oracle 8.1.7 подсистема Ultra Search стала использовать Oracle Text. В настоящее время Ultra Search включается в состав СУБД Oracle и Oracle Application Server.

Ultra Search собирает информацию с помощью *паука* (crawler) - Java- процесса, запускаемого Oracle по расписанию. Паук индексирует документы, находящиеся на различных серверах, с помощью Oracle Text исохраняет полученную информацию в базе данных Oracle. Для управления Ultra Search служит веб-приложение, совместимое с J2EE. Разработчики приложений могут обращаться к Ultra Search из процедур на PL/SQL или Java, а также применять API для поиска в собранных пауком результатах.

Подсистема Ultra Search из Oracle Application Server располагается врепозитории метаданных. Пользователи Application Server могут выполнять поиск и получать список результатов с помощью портлета, доступного через Oracle Portal.

Защищенный поиск при извлечении документов организован с учетом прав пользователей. Для этого просматривается список управления доступом (ACL). Эти списки хранятся в XML DB.

## Опция Oracle Spatial Option

Под пространственными данными (spatial data) понимаются данные о географическом местоположении. В опцию Oracle Spatial Option входят функции и процедуры, позволяющие хранить пространственные данные в базе данных Oracle и обращаться к ним с запросами, включающими сравнение координат.

Пример сочетания стандартных условий и пространственных функций - запрос «найти все здания, расположенные в радиусе двух квадратных миль от пересечения Main Street и First Avenue, в которых проживают лица с годовым доходом больше 100 000 долларов, и показать их адреса». Этот запрос может вывести список адресов зданий, а при совместном использовании с геоинформационной системой (ГИС) - нанести соответствующие точки на карту (рис. 14.2). Системы геокодирования определяют координаты по таким реквизитам, как адреса, номера телефонов (суказанием кода региона) и почтовые индексы (включающие широту и долготу), и сохраняют их в базе данных.

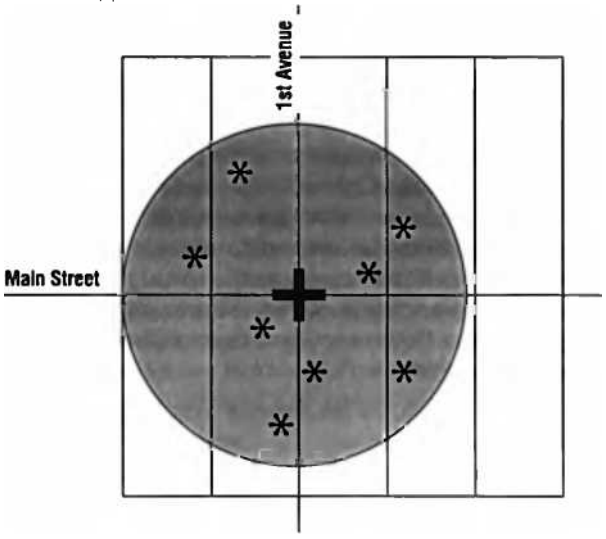



*Рис. 14.2. Отображение результатов пространственного запроса в геоинформационной системе*

Oracle Spatial Option поддерживает различные геометрические фигуры для представления пространственных данных, в том числе: точки, совокупности точек, отрезки и ломаные линии, многоугольники, в том числе с «дырами», окружности и дуги окружностей. Можно определять взаимное расположение фигур с помощью таких операторов, как touch (касается), overlap (перекрывает), inside (находится внутри) и disjoint (не пересекается).

Данные об объектах в одном и том же координатном пространстве, но описывающих разные характеристики (например, физические и экономические), часто моделируются с помощью слоев. Каждый слой разбивается на «квадраты», представляющие подобласти большой области. Представления квадратов хранятся в пространственном индексе, позволяющем быстро находить

различные атрибуты одного и того же квадрата. В Spatial Option эти представления используются для быстрого поиска по пространственным характеристикам. Например, можно спросить, какие пустые породы, минералы и водные источники имеются в заданной физической области. Скорее всего, каждая из этих характеристик хранится в отдельном слое, но их можно быстро сопоставить с соответствующими квадратами. При проектировании пространственных баз данных разрешающую способность карты можно увеличить, разбив ее на большее количество квадратов.

В опции Spatial Option широко применяются объектно-ориентированные средства Oracle - в виде *пространственного типа данных*, который представляет геометрические конфигурации, составленные из одного или нескольких элементов. Пространственные координаты хранятся в массивах переменной длины VARRAY.

В версии Oracle Database 10g появилось средство GeoRaster для хранения, индексирования, опроса, анализа и распространения растровых изображений, ассоциированных с ними векторных геометрических данных и метаданных. Это средство позволяет хранить многослойные решетки и цифровые изображения в объектно-реляционной схеме с привязкой к системе координат. В Oracle Database llg были добавлены трехмерные геометрические объекты и улучшена поддержка вебслужб за счет включения бизнес-каталога, Web Feature Service (WFS), Catalog Services для Web (CSW) и поддержки стандарта OpenLS.

В реальной практике приложения, работающие с пространственными данными, обычно не строятся самостоятельно. Вместо этого организация покупает готовую геоинформационную систему, созданную поверх СУБД. Многие поставщики таких ГИС включают технологию Oracle Spatial в свои продукты.

## Использование инфраструктуры расширяемости в Oracle

СУБД Oracle позволяет пользователям расширять базовую функциональность. В инфраструктуре расширяемости Oracle имеются точки входа, куда разработчик может подключить собственный код для расширения имеющейся функциональности. Примеры:

*Добавление новых операторов для применения в SQL-командах*
: Такие операторы могут оказаться полезными при работе с расширенными типами данных, например мультимедийными или пространственными. Можно создавать операторы, ориентированные на заданный тип данных, например оператор сравнения CLOSER TO (ближе), применимый к пространственным данным.

*Организация кооперативного индексирования*
: Кооперативным индексированием называется методика, при которой за построение и использование индекса отвечает внешнее приложение. Затем к этому индексу можно обращаться при обработке сложных типов данных. Такие индексы называются *предметными*.

*Расширение оптимизатора*
: Если вы применяете расширенные индексы, определяемые пользователем типы данных или другие средства, то можете уточнить про- цедуру сбора статистики, а также определить меру избирательности и функции стоимости. Оптимизатор по стоимости воспользуется вашими определениями для выработки плана выполнения запроса.

*Добавление картриджных служб*
: Так называют службы, используемые в расширениях Oracle (к примеру, для работы с пространственными данными) для управления памятью, контекстом и параметрами, манипуляций со строками и числами, файлового ввода/вывода, интернационализации, извещения об ошибках и управления потоками. Разработчики ПО могут обращаться к этим службам для единообразной интеграции расширений с СУБД Oracle.

Все это позволяет сторонним разработчикам включать дополнительную функциональность, не вмешиваясь в работу основных механизмов СУБД, таких как управление безопасностью, резервное копирование и восстановление или интерфейс на основе языка SQL.