Министерство образования и науки, молодежи и спорта Украины

Харьковский национальный университет

Широкопетлева М.С., Черепанова Ю.Ю., Мазурова О.А.

ПИ

Физическая организация БД на примере СУБД Oracle

ПИ

Харьков

2012

Содержание

Теория

Основные концепции СУБД Oracle

Программное обеспечение баз данных - это ключ к решению проблем управления информацией. Вообще говоря, система управления базой данных (СУБД) должна быть способна надежно управлять большими объемами данных в многопользовательской среде, так, чтобы все пользователи могли одновременно обращаться к одним и тем же данным. Все это должно достигаться при обеспечении высокой производительности пользователей базы данных. СУБД также должна быть защищена от несанкционированного доступа, и должна предоставлять эффективные решения для восстановления от сбоев.

Сервер ORACLE обеспечивает эффективные и действенные решения для основных средств баз данных:

- ORACLE поддерживает самые большие базы данных, потенциального размера до сотен гигабайт. Чтобы обеспечить действенный контроль за использованием дорогостоящих дисковых устройств, он предоставляет полный контроль распределения пространства.

- ORACLE поддерживает большое число пользователей, одновременно выполняющих разнообразные приложения, которые оперируют одними и теми же данными. Он минимизирует соперничество за данные и гарантирует согласованность данных.

- ORACLE поддерживает все описанные выше средства, сохраняя высокую степень суммарной производительности системы. Пользователи базы данных не страдают от низкой производительности обработки.

- На некоторых установках ORACLE работает 24 часа в сутки, не имея периодов разгрузки, ограничивающих пропускную способность базы данных. Нормальные системные операции, такие как откат базы данных, а также частичные сбои компьютерной системы, не прерывают работу с базой данных.

- ORACLE может выборочно управлять доступностью данных, как на уровне базы данных, так и на более низких уровнях. Например, администратор может отключить доступ к конкретному приложению (с тем, чтобы можно было осуществить перезагрузку данных этого приложения), не затрагивая других приложений.

- ORACLE удовлетворяет промышленно принятым стандартам по языку доступа к данным, операционным системам, интерфейсам с пользователем и сетевым протоколам. Это "открытая" система, которая защищает инвестиции заказчика. Сервер ORACLE7 был сертифицирован Национальным институтом стандартов и технологий США как 100%-совместимый со стандартом ANSI/ISO SQL89. ORACLE7 полностью удовлетворяет требованиям правительственного стандарта США FIPS127-1 и имеет "маркировщик" для подчеркивания нестандартных применений SQL. Кроме того, ORACLE7 был оценен Правительственным национальным

центром компьютерной безопасности (NCSC) как совместимый с критериями защиты Оранжевой книги; сервер ORACLE7 и Trusted ORACLE7 отвечают соответственно как уровням C2 и B1 Оранжевой книги, так и сравнимым с ними европейским критериям защиты ITSEC.

- Для защиты от несанкционированного доступа к базе данных ORACLE предоставляет защищенные от сбоев средства безопасности, лимитирующие и отслеживающие доступ к данным. Эти средства позволяют легко управлять даже наиболее сложными схемами доступа.

- ORACLE автоматически поддерживает целостность данных, соблюдая "организационные правила", которые диктуют стандарты приемлемости данных. Как следствие, устраняются затраты на кодирование и сопровождение проверок в многочисленных приложениях базы данных.

- Чтобы извлечь максимум преимуществ из данной компьютерной системы или сети, ORACLE позволяет разделять работу между сервером базы данных и прикладными программами клиентов. Вся тяжесть управления совместно используемыми данными может быть сосредоточена в компьютере, выполняющем СУБД, в то время как рабочие станции, на которых работают приложения, могут сконцентрироваться на интерпретации и отображении данных.

- В компьютерных окружениях, соединенных сетями, ORACLE комбинирует данные, физически находящиеся на разных компьютерах, в одну логическую базу данных, к которой имеют доступ все пользователи сети. Распределенные системы обладают такой же степенью прозрачности для пользователей и согласованности данных, что и нераспределенные системы, предоставляя в то же время преимущества управления локальной базой данных.

- Программное обеспечение ORACLE переносимо между различными операционными системами и одинаково во всех системах. Приложения, разрабатываемые для ORACLE, могут быть перенесены в любую операционную систему с минимумом модификаций или вообще без таковых.

- Программное обеспечение ORACLE совместимо с промышленными стандартами, включая большинство стандартных операционных систем. Приложения, разрабатываемые для ORACLE, могут использоваться в любой операционной системе с минимумом модификаций или вообще без таковых. Программное обеспечение ORACLE позволяет различным типам компьютеров и операционных систем совместно использовать информацию посредством сетей.

Физическая и логическая структура ORACLE

База данных ORACLE имеет как физическую, так и логическую структуру. За счет разделения физической и логической структуры базы данных достигается возможность управления физической структурой данных, не затрагивая доступа к логическим структурам данных.

Физическая структура базы данных ORACLE определяется файлами операционной системы, из которых состоит база данных. Каждая база данных ORACLE составляется из файлов трех типов: одного или нескольких файлов данных, двух или более файлов журнала повторения

работы и одного или нескольких управляющих файлов.

Файлы базы данных предоставляют действительную физическую память для информации базы данных.

Логическая структура базы данных ORACLE определяется:

-одним или несколькими табличными пространствами;

-объектами схем базы данных (таблицами, обзорами, индексами, кластерами, последовательностями, хранимыми процедурами).

Логические структуры хранения, включая табличные пространства, сегменты и экстенты, определяют, как используется физическое пространство базы данных. Объекты схем и отношения между ними формируют реляционную структуру базы данных.

Блоки данных, экстенты и сегменты ORACLE предоставляет возможность тонкого контроля за использованием дисковой памяти через структуры логического хранения, включая блоки данных, экстенты и сегменты.

Механизмы ORACLE работают через использование структур памяти и процессов. Все структуры памяти располагаются в основной памяти (иногда называемой виртуальной памятью или памятью произвольного доступа) компьютеров, составляющих систему базы данных.

СУБД Oracle имеет собственный язык PL/SQL

PL/SQL - это принадлежащее фирме Oracle процедурное языковое расширение языка SQL. PL/SQL сочетает легкость и гибкость SQL с процедурными возможностями языка структурного программирования, такими как IF...THEN, WHILE и LOOP.

При написании приложения базы данных разработчик должен рассмотреть преимущества использования хранимых подпрограмм PL/SQL:

- Поскольку код PL/SQL может сохраняться централизованно в базе данных, сетевой трафик между приложениями и базой данных сокращается, что увеличивает производительность как приложений, так и системы.

- Благодаря хранимому коду PL/SQL можно контролировать доступ к данным. При этом методе пользователи PL/SQL могут обращаться к данным лишь так, как это предусмотрено разработчиком приложения (если не предоставлен иной маршрут доступа).

- Блоки PL/SQL могут пересылаться приложениями к базе данных, что позволяет выполнять комплексные операции без избыточной нагрузки на сеть.

Даже если PL/SQL не хранится в базе данных, приложения могут пересылать к базе данных

не отдельные предложения SQL, а целые блоки, опять-таки сокращая сетевой трафик. Следующие секции описывают различные программные единицы, которые могут быть определены и централизованно сохранены в базе данных.

Процедуры и функции представляют собой совокупности предложений SQL и PL/SQL, сгруппированных в единицу для решения специфической проблемы или выполнения множества взаимосвязанных задач. Процедура создается и сохраняется в базе данных в откомпилированной форме, и может выполняться (вызываться) любым пользователем или приложением. Процедуры и функции похожи друг на друга, с той разницей, что функция всегда возвращает вызывающей программе единственное значение, тогда как процедура не возвращает значения.

Пакеты дают метод инкапсулирования и хранения взаимосвязанных процедур, функций, переменных и других конструктов пакета как единицы в базе данных. Предоставляя администратору базы данных или разработчику приложений организационные преимущества, пакеты в то же время расширяют функциональные возможности (например, глобальные переменные пакета могут объявляться и использоваться любой процедурой в пакете) и увеличивают производительность базы данных (так, все объекты пакета синтаксически разбираются, компилируются и загружаются в память один раз).

Триггеры базы данных ORACLE позволяет вам писать процедуры, которые выполняются автоматически в результате обновления, вставки или удаления из таблицы. Такие процедуры называются триггерами базы данных. Триггеры базы данных могут использоваться самыми разнообразными способами для информационного управления вашей базой данных. Например, их можно использовать для автоматизации генерации данных, аудита модификаций данных, введения в действие комплексных ограничений целостности или организации сложных процедур обеспечения защиты.

Многопользовательские системы баз данных, такие как ORACLE, включают средства защиты, которые контролируют обращения к базе данных и использование данных.

▼ **Подробнее**

Например, механизмы защиты выполняют следующее:

- предотвращают несанкционированный доступ к базе данных

- предотвращают несанкционированный доступ к объектам схем

- контролируют использование дисков

- контролируют использование системных ресурсов (таких как время процессора)

- осуществляют аудит действий пользователя

Защита базы данных может быть классифицирована по двум различным категориям: защита

системы и защита данных. **Защита системы** включает механизмы, контролирующие доступ к базе данных и ее использование на уровне системы.

▼ **Подробнее**

Например, защита системы включает:

- действительные комбинации имен пользователей и паролей;

- уполномочен ли пользователь присоединяться к базе данных ;

- объем дисковой памяти, доступный объектам пользователя ;

- лимиты ресурсов для пользователя ;

- активность или неактивность аудита базы данных ;

-какие системные операции разрешено выполнять пользователю .

Защита данных включает механизмы, контролирующие доступ к базе данных и ее использование на уровне объектов.

▼ **Подробнее**

Например, защита данных включает:

- какие пользователи имеют доступ к конкретному объекту схемы и какие типы действий разрешены каждому пользователю на этом объекте (например, пользователь SCOTT может выдавать для таблицы EMP предложения SELECT и INSERT, но не предложения DELETE) ;

- какие действия подлежат аудиторскому отслеживанию для каждого объекта схемы.

ORACLE управляет защитой базы данных, используя несколько различных средств, среди которых:

- пользователи базы данных и схемы;

- привилегии;

- роли;

- назначения пространства и квоты;

- лимиты на ресурсы;

- аудитинг.

Логическая структура

Структуры логического распределения в базе данных Oracle

Oracle распределяет пространство БД для всех ее логических единиц (таблиц, индексов и т.п.). Единицами логического распределения в Oracle являются блоки данных, экстенты и сегменты. Следующий рисунок иллюстрирует отношения между этими структурами данных.

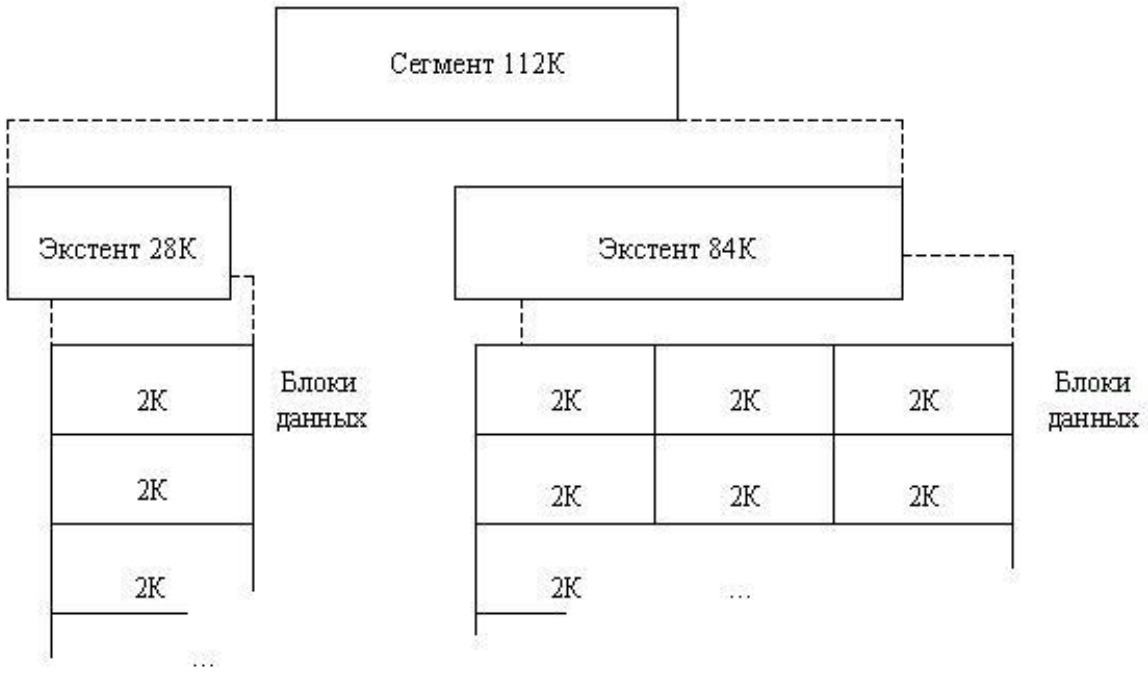


Рисунок 1 - Отношения между сегментами, экстентами и блоками данных

Блоки данных

На самом низком уровне рассмотрения, данные базы данных ORACLE хранятся в ***блоках данных*** (называемых также логическими блоками, блоками ORACLE или страницами). Один блок данных соответствует фиксированному числу байт физического пространства базы данных на диске. Размер блока данных специфически устанавливается для каждой базы данных ORACLE при ее создании. Этот размер кратен размеру блока операционной системы, но не превышает определенный максимум. Важно помнить, что база данных, на ее самом низком уровне, использует и распределяет свободное пространство в базе данных блоками данных ORACLE.

Все данные на физическом уровне, т.е. на уровне операционной системы, распределяются в байтах. Каждая операционная система имеет то, что называется размером блока, который определяется как специфическое число байт на диске.

Экстенты

Следующий уровень логического пространства базы данных называется экстентом.

Экстент - это специфическое число смежных блоков данных, распределяемых для хранения специфического типа информации.

Сегменты

Уровень логического пространства базы данных, следующий за экстентом, называется ***сегментом***.

Сегмент - это совокупность экстентов, распределенных для специфического типа структуры данных, и находящихся в одном и том же табличном пространстве. Например, данные каждой таблицы хранятся в ее собственном сегменте данных, а данные каждого индекса хранятся в его собственном сегменте индекса.

ORACLE распределяет пространство для сегментов экстентами. Поэтому, когда существующие экстенты сегмента заполнены, ORACLE распределяет очередной экстент для этого сегмента. Поскольку экстенты распределяются при необходимости, экстенты сегмента не обязательно смежны на диске, и могут быть распределены между различными файлами. Каждый экстент, однако, не может находиться в нескольких файлах.

Управление пространством

Итак, ORACLE управляет пространством в файлах данных базы данных в единицах, называемых блоками данных (наименьшими единицами ввода-вывода в базе данных).

Формат блока данных ORACLE один и тот же, независимо от того, содержит ли блок данные таблицы, индекса или кластера. Рис.2 иллюстрирует формат блока данных.

Таблица 1 - Формат блока данных

Блок базы данных
Общий и переменный заголовок
Оглавление таблиц
Оглавление строк
Свободное пространство
Данные строк

Заголовок содержит общую информацию блока, такую как адрес блока и тип сегмента (сегмент данных, сегмент индекса или сегмент отката). Заголовок составляет накладные

расходы блока, которые имеют переменный размер. В среднем, суммарные накладные расходы фиксированной и переменной частей блока составляют от 84 до 107 байт.

Часть блока, составляющая *оглавление таблиц*, содержит информацию о том, какие таблицы имеют строки в этом блоке.

Оглавление строк - эта часть блока содержит информацию о действительных строках в блоке (включая адреса каждой порции строки в области данных строк). После того, как в оглавлении строк распределено пространство, это пространство не освобождается при удалении строки. Поэтому блок, который сейчас пуст, но когда-то содержал до 50 строк, по-прежнему имеет 100 байт, распределенных в заголовке для оглавления строк. Это пространство используется повторно лишь тогда, когда в блок вставляются новые строки.

Данные строк - эта порция блока содержит данные таблицы или индекса. Строки могут переходить из блока в блок.

Свободное пространство в блоке используется для вставки новых строк и для обновлений строк, требующих дополнительного пространства (например, при замене пустых хвостовых значений на непустые значения). Будут ли конкретные вставки действительно осуществляться в данном блоке - зависит от значения параметра управления пространством PCTFREE и от текущей величины свободного пространства в блоке.

Введение в PCTFREE, PCTUSED и цепочки строк

Два параметра управления пространством, PCTFREE и PCTUSED, позволяют разработчику управлять использованием свободного пространства для вставок и обновлений строк в блоках данных. Оба этих параметра могут быть специфицированы лишь при создании (CREATE) или изменении (ALTER) таблиц и кластеров (сегментов данных). Кроме того, параметр PCTFREE можно также специфицировать при создании или изменении индексов (сегментов индекса).

Параметр PCTFREE

Параметр PCTFREE устанавливает процент памяти блока, резервируемой (оставляемой свободной) для возможных обновлений строк, уже содержащихся в блоке.

Например, предположим, что вы специфицировали следующий параметр в предложении CREATE TABLE: PCTFREE 20

Это требует, чтобы 20% места в каждом блоке данных в сегменте данных этой таблицы оставлялись свободными и доступными для возможных обновлений строк, уже существующих в каждом блоке (см. рис. 2).

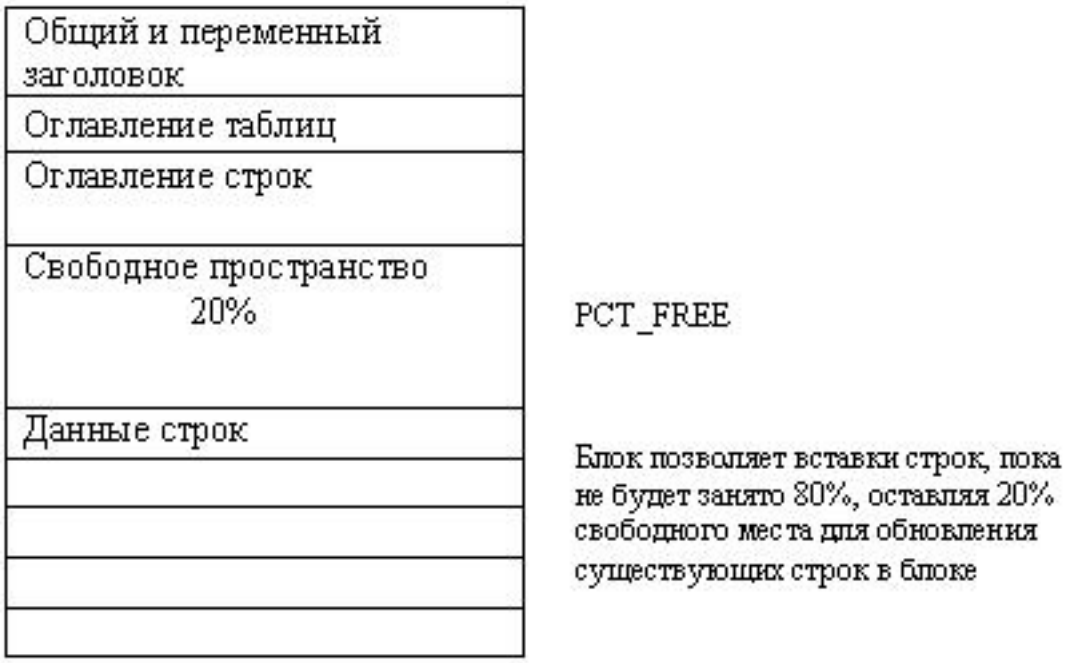


Рисунок 2 - Структура блока с параметром PCTFREE

До того, как будет достигнут процент PCTFREE, свободное пространство в блоке данных заполняется как вставками новых строк, так и ростом заголовка блока данных.

Параметр PCTUSED

После того, как блок данных будет заполнен до процента PCTFREE, этот блок не рассматривается для вставки новых строк до тех пор, пока процент используемой памяти в блоке не упадет ниже параметра PCTUSED (рис.3). До этого момента свободная память в блоке может использоваться лишь для обновления строк, уже содержащихся в блоке данных.

Например, предположим, что вы специфицировали следующий параметр в предложении CREATE TABLE: PCTUSED = 40

В этом случае, блок данных в сегменте данных, после заполнения его до отметки PCTFREE, не будет рассматриваться для вставки новых строк до тех пор, пока процент используемой памяти в блоке не упадет до 39% или ниже.

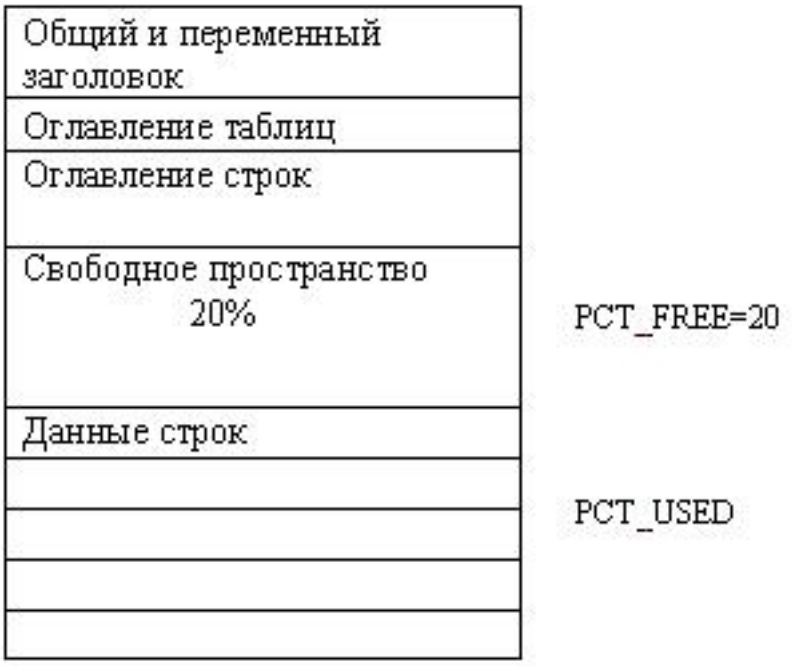


Рисунок 3 - Структура блока с параметром PCTUSED

PCTFREE и PCTUSED работают вместе, чтобы оптимизировать использование пространства в блоках данных экстентов внутри сегмента данных.

Во вновь распределенном блоке данных, место, доступное для вставок, равно размеру блока минус сумма накладных расходов (заголовок блока) и PCTFREE. Обновление существующих данных может использовать все свободное пространство в блоке; поэтому обновления могут сделать свободное место в блоке меньшим чем PCTFREE, - пространство, резервируемое для обновлений, но недоступное для вставок.

Для каждого сегмента данных и сегмента индекса ORACLE поддерживает один или несколько СВОБОДНЫХ СПИСКОВ; свободный список - это список блоков данных, которые были распределены для экстентов этого сегмента и имеют процент свободной памяти, превышающий PCTFREE; эти блоки доступны для вставок.

Когда выдается предложение INSERT, ORACLE ищет в свободном списке таблицы первый доступный блок и использует его, если можно; если свободного пространства в этом блоке слишком мало, чтобы удовлетворить требованиям этого INSERT, и по меньшей мере равно PCTUSED, то этот блок выбрасывается из свободного списка. Несколько свободных списков на сегмент могут уменьшить соперничество за свободные списки, когда имеют место одновременные вставки.

Когда выдаются предложения DELETE и UPDATE, ORACLE проверяет, не стало ли место, используемое в блоке, меньше чем PCTUSED; если это так, то блок передвигается в начало

свободного списка, и будет первым используемым из доступных блоков.

Два типа предложений освобождают пространство в одном или нескольких блоках данных: предложения DELETE и те предложения UPDATE, которые заменяют существующие значения на меньшие. Память, освобожденная в результате этих типов предложений, доступна для последующих предложений INSERT.

Освобождаемое пространство не обязательно будет смежным с основной областью свободного пространства в блоке. Свободное пространство в блоке объединяется ("сжимается") только тогда, когда предложение UPDATE или INSERT пытается использовать блок, содержащий достаточно свободного места, для размещения нового куска строки, но фрагментация свободной памяти не позволяет вставить этот кусок строки в непрерывный участок блока. Таким образом, производительность базы данных не страдает от непрерывных и излишних сжатий свободной памяти блоков по каждой операции DELETE или UPDATE.

В некоторых обстоятельствах все данные строки таблицы могут не умещаться в один блок данных. В таком случае данные этой строки сохраняются в ЦЕПОЧКЕ блоков данных, резервируемых в этом сегменте. Цепочки строк возникают чаще всего при больших строках (т.е. строках, содержащих столбец с типом данных LONG или LONG RAW). Такого типа цепочек блоков избежать невозможно. Если строка в блоке данных обновляется так, что общая длина строки увеличивается, а свободное пространство в блоке заполнено, то данные всей строки МИГРИРУЮТ, т.е. переносятся в новый блок данных, при условии, что в новом блоке поместится вся строка. На месте первоначального куска мигрировавшей строки записывается указатель на новый блок, содержащий мигрировавшую строку; ROWID (указатель на строку) мигрировавшей строки не изменяется. Когда строка занимает цепочку блоков или мигрирует, производительность операций ввода-вывода, связанных с этой строкой, падает, потому что ORACLE вынужден просматривать больше одного блока данных, чтобы извлечь информацию строки.

Экстенты и сегменты

Экстенты

Экстент - это логическая единица распределения пространства базы данных, состоящая из определенного числа непрерывных блоков данных. Каждый тип сегмента состоит из одного или нескольких экстентов. Когда существующее пространство в сегменте полностью использовано, ORACLE распределяет для сегмента новый экстент.

Независимо от типа, каждый сегмент в базе данных создается по меньшей мере с одним экстентом для хранения его данных. Этот экстент называется начальным экстентом данного сегмента. (Сегменты отката представляют исключение из этого правила; они всегда имеют как минимум два экстента.)

Например, при создании таблицы ее сегмент данных содержит ***начальный экстент*** из

заданного числа блоков данных. Хотя ни одной строки еще не вставлено, блоки данных ORACLE, занимаемые начальным экстентом, уже зарезервированы для строк этой таблицы.

Когда блоки данных в начальном экстенте сегмента заполнятся и потребуется дополнительное пространство для новых данных, ORACLE автоматически распределяет инкрементальный экстент для этого сегмента.

Инкрементальный экстент - это очередной экстент, размер которого совпадает с размером предыдущего экстента для данного сегмента или превышает его на определенную величину (инкремент). Для целей сопровождения, каждый сегмент в базе данных содержит заголовок сегмента, который описывает характеристики этого сегмента и оглавление (список) экстентов в этом сегменте.

ORACLE управляет действительным распределением экстентов для данного сегмента. При распределении нового экстента для сегмента выполняются следующие шаги:

Шаг 1. ORACLE отыскивает (в табличном пространстве, содержащем сегмент) первый свободный, непрерывный участок блоков данных, составляющих размер инкрементального сегмента (или больше).

Свободное пространство для нового экстента отыскивается с помощью следующего алгоритма:

а) ORACLE ищет непрерывный набор блоков данных, совпадающий с размером нового экстента, предварительно округленным вверх для уменьшения внутренней фрагментации.

Например, если новый экстент требует 19 блоков данных, ORACLE ищет ровно 20 непрерывных блоков данных.

б) Если точное совпадение не найдено, ORACLE ищет множество непрерывных блоков данных, размером не меньшее требуемого размера. Если ORACLE находит группу непрерывных блоков, минимум на пять блоков большую, чем требуемый размер экстента, то он расщепляет эту группу блоков на отдельные экстенты, один из которых имеет требуемый размер; если размер такой группы превышает требуемый размер меньше, чем на пять блоков, то новому экстенту распределяется вся эта группа блоков.

Продолжая наш пример, если ORACLE не находит группу ровно из 20 непрерывных блоков данных, он начинает искать множество непрерывных блоков данных числом больше 20. Если первая найденная группа содержит 25 или более блоков, ORACLE разбивает эти блоки, выделяя из них новый экстент; если первая найденная группа содержит от 21 до 24 блоков, новому экстенту распределяются все эти блоки.

в) Если больший набор непрерывных блоков не найден, ORACLE соединяет все свободные смежные блоки данных в соответствующем табличном пространстве, так что формируются группы непрерывных блоков большего размера. После соединения блоков данных в табличном пространстве этапы поиска (а) и (б) повторяются второй раз.

Если экстент не может быть распределен и после вторичного поиска, возвращается ошибка.

Шаг 2. После того, как ORACLE находит необходимое свободное место в табличном пространстве, часть этого свободного места, соответствующая размеру инкрементального экстента, распределяется этому экстенту. Если ORACLE нашел участок свободного пространства большего размера, чем требуется для экстента, то остаток оставляется как свободное пространство (но не меньше, чем пять непрерывных блоков).

Шаг 3. Заголовок сегмента и словарь данных обновляются, чтобы показать, что новый сегмент распределен, и что распределенное пространство больше не свободно. Обычно ORACLE обнуляет блоки вновь распределенного экстента при первом использовании этого экстента; в некоторых случаях ORACLE обнуляет блоки экстента в момент распределения этого экстента.

Когда освобождаются экстенты

В общем случае, экстенты сегмента не возвращаются в табличное пространство, пока сегмент не освобождается как единица, как, например, при удалении таблицы предложением DROP TABLE.

Все экстенты, распределенные сегменту индекса, остаются распределенными, пока существует индекс. Когда индекс или ассоциированная таблица (или кластер) удаляется, его экстенты освобождаются для табличного пространства.

Также, ORACLE периодически проверяет, не выросли ли сегменты отката в базе данных больше их оптимального размера. Если сегмент отката имеет размер больше оптимального (т.е. имеет слишком много экстентов), то ORACLE автоматически освобождает один или несколько экстентов из этого сегмента отката.

Кроме того, когда заканчивается выполнение предложения, требовавшего временного сегмента, этот временный сегмент автоматически удаляется, а его экстенты возвращаются в табличное пространство, в котором был создан сегмент.

Определение размеров и лимитов для экстентов сегментов

Каждый сегмент определяется несколькими ПАРАМЕТРАМИ ПРОСТРАНСТВА, выражаемыми в терминах экстентов. Параметры пространства применяются ко всем типам сегментов. Они контролируют, как СУБД распределяет свободное пространство базы данных для данного сегмента.

Например, специфицируя параметры пространства в фразе STORAGE предложения CREATE TABLE при создании таблицы, вы можете определить, сколько памяти будет первоначально зарезервировано для сегмента данных таблицы, или лимитировать максимальный объем пространства (число экстентов), которое может быть распределено для таблицы.

Сегменты

Сегмент - это набор экстентов, содержащих все данные для конкретного типа структуры логического пространства внутри табличного пространства.

Например, для каждой таблицы ORACLE распределяет один или несколько экстентов, чтобы сформировать сегмент данных этой таблицы; для каждого индекса ORACLE распределяет один или несколько экстентов, чтобы сформировать сегмент индекса для этого индекса.

База данных ORACLE может содержать четыре различных типа сегментов:

- сегмент данных;

- сегмент индекса;

- сегмент отката;

- временный сегмент.

Сегмент данных

Каждая некластеризованная таблица (в том числе снимок и журнал снимков) в базе данных ORACLE имеет единственный сегмент данных, содержащий все данные этой таблицы. Этот сегмент данных косвенно создается командой CREATE TABLE. Параметры пространства для таблицы, снимка, журнала снимков или кластера определяют способ распределения экстентов для сегмента данных. Непосредственное указание этих параметров через команды CREATE TABLE /SNAPSHOT /SNAPSHOT LOG /CLUSTER или ALTER TABLE /SNAPSHOT /SNAPSHOT LOG /CLUSTER влияет на эффективность хранения и извлечения данных в этом сегменте данных

Сегмент индекс

Каждый индекс в базе данных ORACLE имеет единственный сегмент индекса, содержащий все данные этого индекса. Этот сегмент косвенно создается командой CREATE INDEX. Эта команда позволяет вам специфицировать параметры пространства для экстентов сегмента индекса, а также табличное пространство, в котором должен быть создан этот сегмент индекса. (Сегмент данных для таблицы и сегмент индекса для ассоциированного индекса не обязаны размещаться в одном и том же табличном пространстве.) Создатель индекса может управлять способом распределения экстентов для сегмента индекса; непосредственное указание параметров пространства влияет на эффективность хранения и извлечения данных в этом сегменте индекса.

Сегменты отката

Каждая база данных содержит один или несколько сегментов отката.

Сегмент отката - это часть базы данных, в которой записываются действия транзакций, которые, возможно, придется отменить в некоторых обстоятельствах. Сегменты отката используются для обеспечения согласованности по чтению, для отката транзакций и для восстановления базы данных.

Информация в сегменте отката состоит из нескольких записей отката. Среди прочего, запись отката включает информацию блока (имя файла и ID блока, соответствующего измененным данным) и данные в том виде, какими они были перед действием транзакции. Записи отката для одной и той же транзакции связаны друг с другом, что позволяет легко найти эти записи при необходимости осуществить откат транзакции. К сегментам отката не могут обращаться ни пользователи, ни администратор базы данных; их пишет и читает только ORACLE.

Сегмент отката SYSTEM

Начальный сегмент отката SYSTEM создается при создании базы данных. Этот начальный сегмент создается в табличном пространстве SYSTEM и использует умалчиваемые параметры, ассоциированные с этим табличным пространством. Сегмент отката SYSTEM не может быть удален. Oracle всегда получает сегмент отката SYSTEM в дополнение ко всем остальным необходимым ей сегментам отката. Если в наличии есть несколько сегментов отката, ORACLE пытается использовать сегмент отката SYSTEM лишь для специальных системных транзакций, и распределяет пользовательские транзакции среди других сегментов отката; если таких транзакций слишком много, то при необходимости используется и сегмент SYSTEM. В общем случае, после создания базы данных, в табличном пространстве SYSTEM должен быть создан по меньшей мере один дополнительный сегмент отката.

Состояния сегмента отката

Сегмент отката всегда находится в одном из нескольких состояний: офлайн, получен инстанцией, участвует в неподтвержденной транзакции, нуждается в восстановлении либо удален. Состояние сегмента отката определяет, может ли он использоваться в транзакциях, и какие административные процедуры администратор может выполнять на этом сегменте. Сегмент отката может иметь следующие состояния:

- **OFFLINE** не был получен ни одной инстанцией;

- **ONLINE** был получен (переведен в онлайн) инстанцией; может содержать данные от активных транзакций;

- **NEEDS RECOVERY** - содержит данные от неподтвержденных транзакций, которые не могут быть отменены (из-за того, что соответствующие файлы данных недоступны);

- запорчен **PARTLY AVAILABLE** - содержит данные от сомнительной распределенной транзакции;

- **INVALID** - был удален (пространство, которое было распределено этому сегменту отката, будет позднее занято при создании нового сегмента отката).

Состояния PARTLY AVAILABLE (частично доступен) и NEEDS RECOVERY (требует восстановления) очень похожи: и в том, и в другом состоянии сегмент отката обычно содержит данные от незаконченной транзакции.

Различия между этими двумя состояниями следующие:

- Сегмент отката PARTLY AVAILABLE используется сомнительной распределенной транзакцией, которая не может быть разрешена из-за сетевого сбоя. Сегмент отката NEEDS RECOVERY используется транзакцией (локальной или распределенной), которая не может быть разрешена из-за локального сбоя носителя, например, отсутствующего или запорченного файла данных, или запорчен сам этот сегмент.

- ORACLE или АБД может перевести сегмент PARTLY AVAILABLE в состояние ONLINE. Напротив, сегмент NEEDS RECOVERY должен быть переведен сначала в OFFLINE, прежде чем может быть переведен в ONLINE. (Когда база данных восстанавливается и тем самым разрешает транзакцию, ORACLE автоматически изменяет состояние сегмента отката NEEDS RECOVERY на OFFLINE.)

- АБД может удалить (DROP) сегмент NEEDS RECOVERY. (Это позволяет администратору удалять запорченные сегменты.) Сегмент PARTLY AVAILABLE не может быть удален; сначала должна быть разрешена сомнительная транзакция, - либо автоматически, процессом RECO, либо вручную администратором.

После перевода сегмента отката PARTLY AVAILABLE в состояние ONLINE (администратором или во время запуска инстанции) ORACLE может использовать его для новых транзакций.

Кроме того, пока сомнительная транзакция не разрешена, она продолжает удерживать в сегменте отката занятые ей экстенты, не позволяя другим транзакциям использовать их; поэтому сегменту отката может потребоваться получение новых экстентов для активных транзакций, и он будет расти. Чтобы предотвратить рост сегмента отката, АБД может предпочесть создать новый сегмент отката для транзакций, пока не будет разрешена сомнительная транзакция, вместо того чтобы переводить сегмент PARTLY AVAILABLE в состояние ONLINE. Просмотр состояний сегментов отката Таблица словаря данных DBA_ROLLBACK_SEGS перечисляет состояния всех сегментов отката, наряду с другой информацией об этих сегментах.

Когда табличное пространство переводится в офлайн, так что невожможно осуществить немедленный откат транзакций, ORACLE создает ОТСРОЧЕННЫЙ СЕГМЕНТ ОТКАТА. Этот сегмент содержит записи отката, которые не могут быть применены к табличному пространству и должны ждать, пока табличное пространство не будет снова переведено в онлайн. Такие сегменты отката исчезают после того, как табличные пространства будут переведены в онлайн и восстановлены. Отсроченные сегменты отката автоматически создаются в табличном пространстве SYSTEM.

Временные сегменты

При обработке запросов ORACLE часто требует временного рабочего пространства для промежуточных этапов обработки предложения SQL. Это дисковое пространство, называемое ***временным сегментом***, распределяется автоматически. Обычно временный сегмент нужен как рабочая область для сортировки. Сегмент не создается, если операция сортировки может быть выполнена в памяти, или если ORACLE находит какой-нибудь иной способ выполнения операции с использованием индексов.

Следующие предложения SQL могут потребовать использования временного сегмента:

- CREATE INDEX;

- SELECT ... ORDER BY ;

- SELECT DISTINCT ... ;

- SELECT ... GROUP BY ;

- SELECT ... UNION ;

- SELECT ... INTERSECT ;

- SELECT ... MINUS ;

- неиндексированные соединения ;

-некоторые коррелированные подзапросы ;

Например, если запрос содержит фразу DISTINCT, фразу GROUP BY и фразу ORDER BY, то может потребоваться целых два временных сегмента. Если приложения часто выдают предложения, приведенные в показанном списке, то администратору базы данных следует настроить параметр инициализации SORT_AREA_SIZE.

Временные сегменты распределяются при необходимости во время сессии пользователя. Например, пользователь может выдать запрос, требующий распределения трех временных сегментов. Временные сегменты удаляются, когда предложение завершает выполнение. Характеристики пространства для экстентов временного сегмента определяются умолчаниями того табличного пространства, в котором создается этот временный сегмент. Временные сегменты создаются во временном табличном пространстве пользователя, выдавшего предложение, как специфицрованого для этого пользователя опцией TEMPORARY TABLESPACE в командах CREATE USER или ALTER USER. Если для пользователя не было явно специфицировано временное табличное пространство, то для него используется пространство SYSTEM.

Поскольку распределение и освобождение временных сегментов происходит часто, есть резон в том, чтобы создать специальное табличное пространство для временных сегментов.

Таким путем вы можете распределить дисковые операции по различным устройствам и избежать фрагментации SYSTEM и других табличных пространств, которые в противном случае использовались бы для временных сегментов. Журнал повторения не содержит никаких записей для изменений, осуществляемых во временных сегментах.

Табличные пространства и файлы данных

Кроме рассмотренных выше способов физического распределения памяти под логические объекты БД, стоит отметить, что на самом верхнем уровни физическое представление БД следующее:

- файлы данных - хранят собственно данные БД (его структура - коммерческая тайна);

- файл контрольных точек - хранит управляющую информацию, необходимую для работы процессов Oracle (см. тему "Физическая и логическая структура Oracle");

- файлы журнала повторения работы (или журналы повтора, журналы транзакций) - хранят незавершенный транзакции;

- файл настроек (INIT . ORA) - управляющий файл, который хранит все параметры запуска БД Oracle .

База данных ORACLE составлена из одного или и табличные нескольких единиц логической памяти, называемых ***табличными пространствами.***

Таким образом, используемые данные базы данных ORACLE логически хранятся в табличных пространствах, а физически располагаются в файлах данных, ассоциированных с соответствующим табличным пространством. Рисунок 4 иллюстрирует эту взаимосвязь.

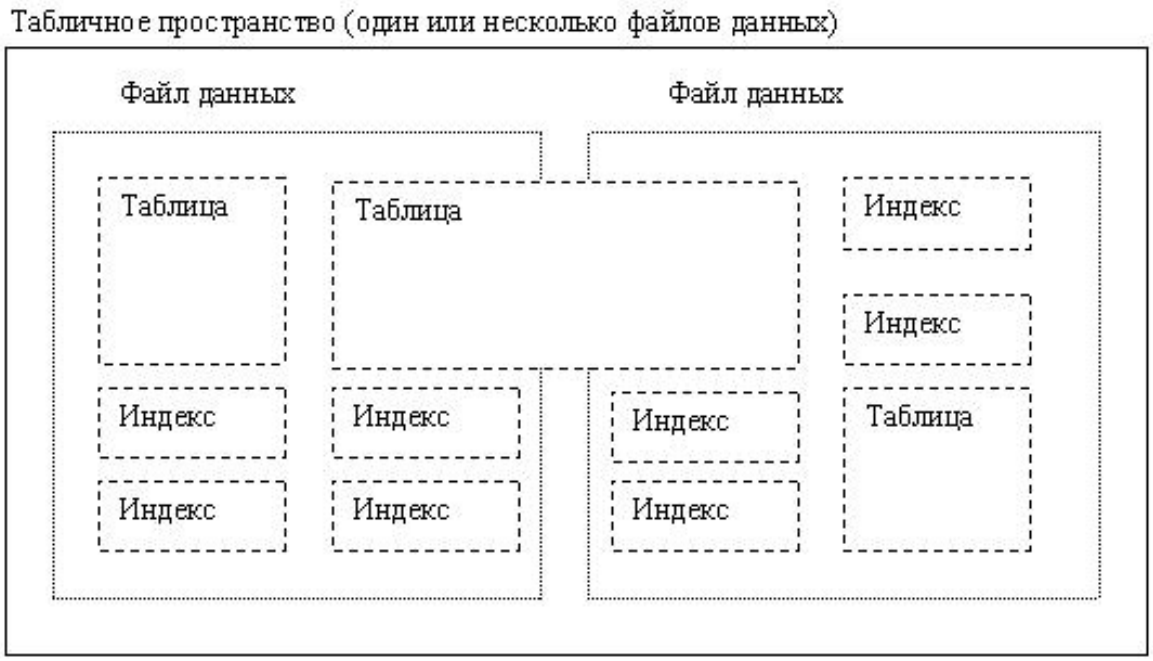


Рисунок 4 - Файлы данных и табличные пространства

Таким образом, ***файлы данных*** - физические структуры, каждая из которых связана с одним табличным пространством. Объекты БД (таблицы, индексы и т.п.) хранятся в табличных пространствах, и могут располагаться в нескольких файлах данных.

Все данные базы данных хранятся в табличных пространствах этой базы данных.

Каждое табличное пространство базы данных ORACLE пространства составлено из одного или нескольких файлов данных.

Файлы данных табличного пространства физически хранят соответствующие данные базы данных на диске.

Данные базы данных в совокупности хранятся в файлах данных, из которых состоит каждое табличное пространство этой базы данных (см. рис. 5).

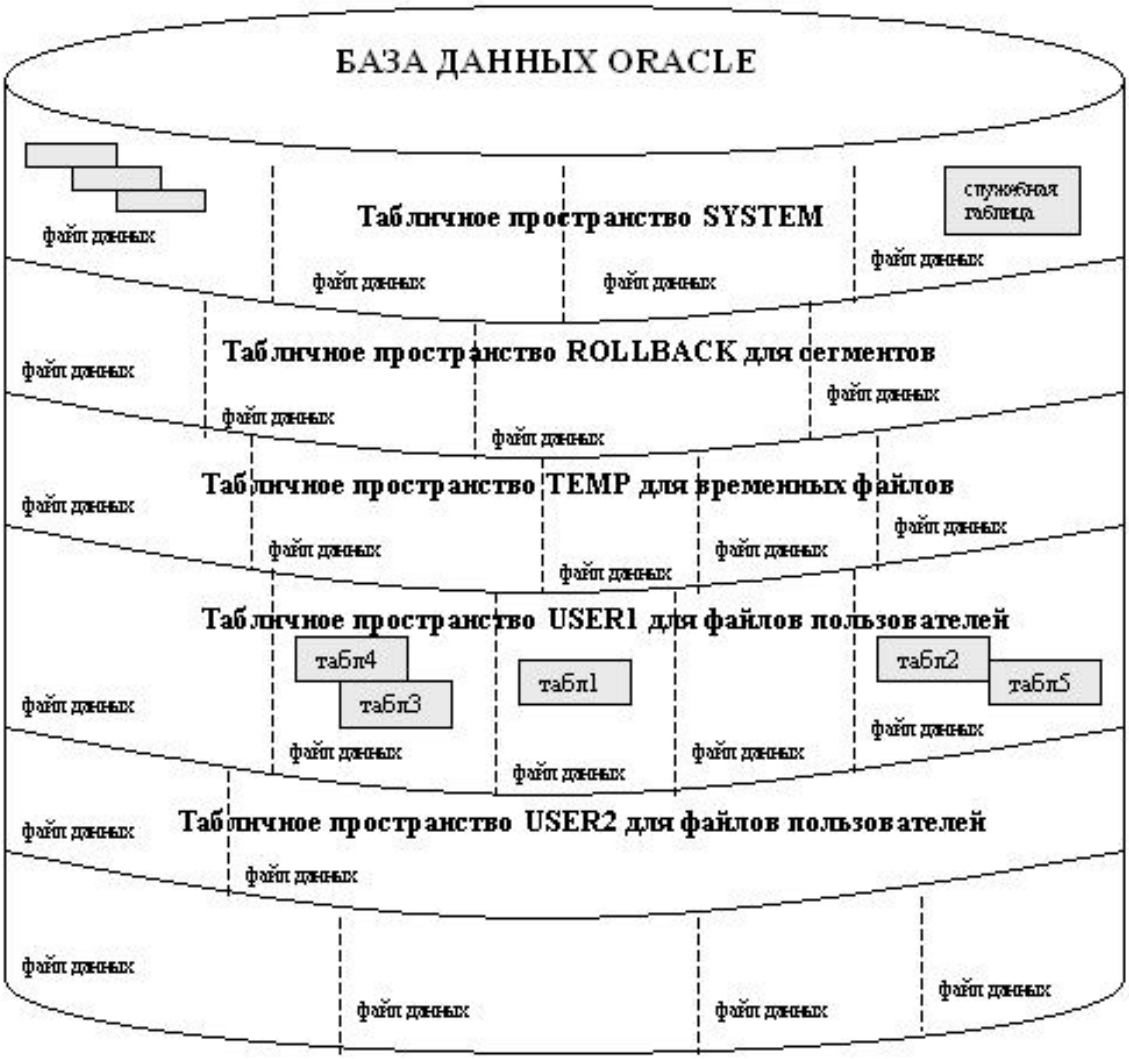


Рисунок 5 - Физическая структура базы данных Oracle

Например, простейшая база данных ORACLE могла бы иметь одно табличное пространство с одним файлом данных. Более сложная база данных могла бы иметь три табличных пространства, каждое из которых состоит из двух файлов данных (что в совокупности дает шесть файлов данных).

Следующая команда позволяет создать табличное пространство:

CREATE TABLESPACE

Для выполнения оператора необходимы полномочия CREATE TABLESPACE .

Табличное пространство SYSTEM должно содержать не менее двух сегментов отката, включая сегмент отката SYSTEM .

Синтаксис команды созданиия табличного пространства

Синтаксис команды созданиия табличного пространства показан на рисунке 6.

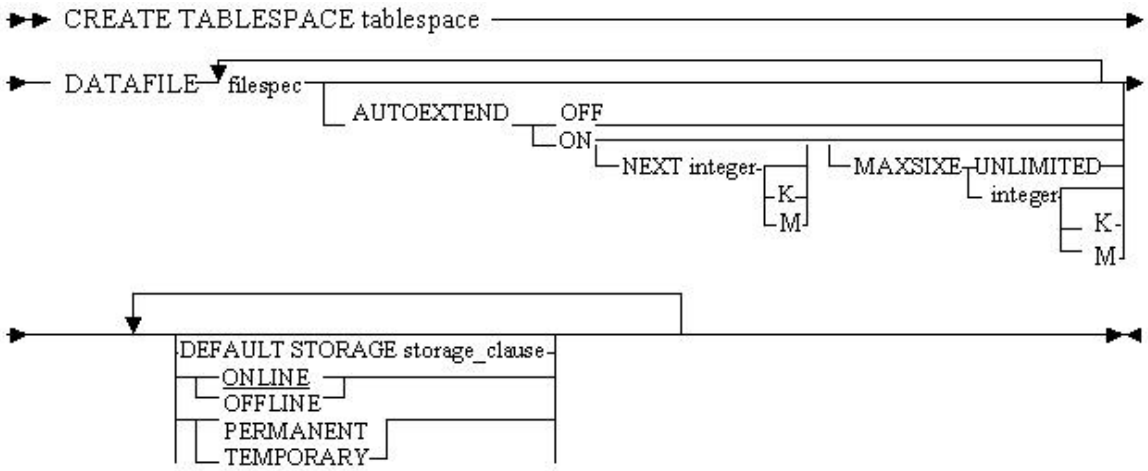


Рисунок 6 - Синтаксис команды созданиия табличного пространства

Ключевые слова и параметры:

- DATAFILE - указывает один или несколько файлов данных, образующих создаваемое табличное пространство.

- AUTOEXTEND - разрешает или запрещает автоматическое увеличение файлов данных.

- OFF - режим запрета автоматического расширения файлов данных. При этом значения параметров NEXT и MAXSIZE обнуляются. Если в дальнейшем потребуется разрешить автоматическое расширение файлов данных, то это можно установить оператором ALTER TABLESPACE AUTOEXTEND , установив новые значения параметров NEXT и MAXSIZE .

- NEXT - размер дискового пространства, добавляемого при необходимости каждый раз в режиме автоматического увеличения дискового пространства. Этот параметр иногда называется экстентом (extent).

- MAXSIZE - максимальный размер дискового пространства, которое можно выделить под файл данных.

- UNLIMITED - устанавливает неограниченный размер дискового пространства, выделяемого под файл данных.

- DEFAULT STORAGE - определяет для всех объектов, создаваемых в табличном

пространстве, значения параметром хранения по умолчанию.

- ONLINE - устанавливает для данного табличного пространства оперативный режим, который позволяет использовать его сразу после создания.

- OFFLINE - устанавливает для данного табличного пространства автономный режим, делающий его недоступным для использования.

- Отметим, что режим функционирования табличного пространства можно изменить оператором ALTER TABLESPACE . Установленные режимы хранятся в словаре данных в виде DBA _ TABLESPACES .

- PERMANENT - определяет, что табличное пространство будет использоваться для хранения постоянных объектов.

При создании объекта схемы, такого как таблица или индекс, создается сегмент этого объекта в табличном пространстве базы данных. Например, предположим, что таблица создается в указанном табличном пространстве с помощью команды CREATE TABLE с опцией TABLESPACE. Пространство для сегмента данных этой таблицы распределяется в одном или нескольких файлах данных, составляющих это табличное пространство. Сегмент объекта может размещаться лишь в одном табличном пространстве базы данных.

Администратор базы данных может использовать табличные пространства для:

- управления распределением памяти для объектов базы данных;

- установления квот памяти для пользователей базы данных;

- управления доступностью данных путем перевода отдельных табличных пространств в состояния online или offline;

- копирования и восстановления данных;

- распределения данных по устройствам для повышения производительности.

АБД может создавать новые табличные пространства, добавлять и удалять файлы данных из табличных пространств, устанавливать и изменять умалчиваемые характеристики пространства для сегментов, создаваемых в табличном пространстве, а также удалять табличные пространства.

Табличное пространство SYSTEM

Каждая база данных ORACLE содержит табличное пространство SYSTEM, которое создается автоматически при создании базы данных. Табличное пространство SYSTEM всегда содержит таблицы словаря данных для всей базы данных. Небольшой базе данных может оказаться достаточным одного табличного пространства SYSTEM; однако рекомендуется создать по

меньшей мере одно дополнительное пространство, чтобы хранить данные пользователей отдельно от информации словаря данных. Это позволит вам более гибко осуществлять разнообразные операции администрирования, а также уменьшит соперничество за одни и те же файлы данных между объектами словаря и объектами схем.

Табличное пространство SYSTEM должно всегда находиться в состоянии онлайн и не может быть переведено в офлайн. Все данные, хранящиеся от имени хранимых программных единиц PL/SQL (процедуры, функции, пакеты и триггеры), размещаются в табличном пространстве SYSTEM. Если для базы данных создается много таких объектов, то АБД должен спланировать память, которая будет занята в табличном пространстве SYSTEM под эти объекты.

Распределение дополнительного пространства для базы данных

Чтобы расширить базу данных, к одному из существующих в ней табличных пространств можно добавить второй файл данных, тем самым, увеличив величину дисковой памяти, распределенной для соответствующего табличного пространства.

Размер базы данных и размеры табличных пространств увеличиваются с добавлением новых файлов данных. Альтернативно, АБД может создать новое табличное пространство (определяемое дополнительным файлом данных), чтобы увеличить размер базы данных.

Размер табличного пространства - это размер его файла данных или суммарный размер всех файлов данных, составляющих это табличное пространство. Размер базы данных - это общий размер всех табличных пространств, составляющих базу данных.

Онлайновые и офлайновые табличные пространства

АБД может перевести любое табличное пространство в базе данных ORACLE в состояние ***онлайн*** (т.е. доступно) или ***офлайн*** (недоступно), если база данных открыта. Единственным исключением является то, что табличное пространство SYSTEM всегда находится в онлайне, ибо словарь данных должен быть всегда доступен ORACLE. Обычное состояние табличного пространства - онлайн, так что данные, содержащиеся в нем, доступны пользователям базы данных.

Однако администратору может понадобиться перевод табличного пространства в офлайн по одной из следующих причин:

- чтобы сделать часть базы данных недоступной, сохраняя в то же время нормальный доступ к остальной части;

- чтобы выполнить резервное копирование офлайнового табличного пространства (хотя такое копирование можно осуществлять и в онлайне, одновременно с использованием табличного пространства);

- чтобы сделать приложение вместе с его группой таблиц временно недоступным на время обновления или сопровождения этого приложения.

Когда табличное пространство переводится в офлайн, ORACLE не позволяет последующим предложениям SQL обращаться к объектам этого табличного пространства. Активные транзакции с уже выполненными предложениями, обращавшимися к данным в переведенном в офлайн табличном пространстве, не затрагиваются на уровне транзакции; однако данные отката, соответствующие таким предложениям, сохраняются в отсроченном сегменте отката (в табличном пространстве SYSTEM) и будут применены к табличному пространству, если потребуется, когда оно будет переведено в онлайн. Табличное пространство не может быть переведено в офлайн, если оно содержит активные сегменты отката. Табличное пространство может быть переведено в офлайн лишь в том случае, если все сегменты отката, содержащиеся в нем, не используются.

В словаре данных (в табличном пространстве SYSTEM) отмечается, переведено ли пространство в онлайн или в офлайн. Если табличное пространство было в состоянии офлайн в момент закрытия базы данных, оно будет в том же состоянии и при последующем монтировании и открытии базы данных. Табличное пространство может быть переведено в онлайн только той базой данных, которой оно переводилось в офлайн, поскольку необходимая информация словаря данных находится в табличном пространстве SYSTEM этой базы данных. Табличное пространство в состоянии офлайн не может читаться и редактироваться никакой другой утилитой, кроме ORACLE. Таким образом, табличные пространства не могут передаваться из одной базы данных в другую. При возникновении некоторых ошибок табличное пространство может автоматически переводиться в состояние офлайн (например, если системный процесс DBWR не сумел записать данные за несколько попыток). Пользователи, пытающиеся работать с таблицами этого пространства, получат сообщение об ошибке. Если эти проблемы вызываются сбоем носителя, то после исправления ошибки оборудования должно быть проведено восстановление табличного пространства. При использовании нескольких табличных пространств для разграничения различных типов данных, АБД может также переводить некоторые из табличных пространств в офлайн для некоторых процедур сопровождения, в то время как остальные табличные пространства остаются в онлайне и их информация доступна для работы. Однако при переводе табличных пространств в онлайн могут возникнуть особые обстоятельства.

Например, если для отделения данных таблиц от их индексов используются два табличных пространства, то справедливо следующее:

- Если табличное пространство, содержащее индексы, переведено в офлайн, то запросы по-прежнему могут обращаться к данным, потому что наличие индексов для запросов не обязательно.

- Если в офлайн переведено табличное пространство, которое содержит таблицы, то данные этих таблиц недоступны, и, следовательно, запросы не могут быть выполнены.

В общем, если ORACLE определяет, что в онлайновых табличных пространствах достаточно информации для выполнения предложения, он выполняет его. Если необходимы данные из офлайнового табличного пространства, предложение возвратит ошибку.

Файлы данных

Табличное пространство в базе данных ORACLE состоит из одного или нескольких физических ***файлов данных***.

Файлы данных, ассоциированные с табличным пространством, хранят все данные этого табличного пространства. Любой файл данных может ассоциироваться только с одним табличным пространством и только с одной базой данных. При создании файла данных для табличного пространства ORACLE распределяет ему указанное количество дисковой памяти. Когда файл данных создается, операционная система несет ответственность за очистку старой информации и за установку должных режимов доступа к файлу, прежде чем он будет распределен ORACLE. Если файл велик, этот процесс (очистка) может потребовать значительного времени. Поскольку первым табличным пространством в любой базе данных всегда является SYSTEM, первые файлы данных любой базы данных автоматически распределяются табличному пространству SYSTEM во время создания базы данных.

Содержимое файла данных

После первоначального создания файла данных соответствующее дисковое пространство еще не содержит никаких данных; однако это пространство зарезервировано за будущими сегментами ассоциированного табличного пространства - оно не может содержать каких-либо иных (чужих) данных. Когда сегмент (например, сегмент данных таблицы) будет создан и начнет увеличиваться в размерах, ORACLE использует свободное место в соответствующих файлах данных, чтобы распределять экстенты для этого сегмента. Данные в сегментах объектов (сегментах данных, сегментах индексов, сегментах отката и т.д.) в табличном пространстве физически хранятся в одном или нескольких файлах данных, составляющих это табличное пространство. Заметьте, что объект схемы не соответствует определенному файлу данных; скорее, файл данных является хранилищем данных любого объекта в конкретном табличном пространстве. Экстенты одного сегмента могут быть распределены в нескольких файлах данных табличного пространства; таким образом, объект может "занимать" один или несколько файлов данных. Обычно АБД и пользователи не могут контролировать, в каких файлах данных размещается объект.

Офлайновые файлы данных

Табличные пространства в любой момент можно переводить в **офлайн** (т.е. делать недоступными) или в **онлайн** (делать доступными). Поэтому все файлы данных, составляющие табличное пространство, переводятся в офлайн или онлайн одновременно, всей группой.

Индивидуальные файлы данных также могут быть переведены в офлайн; однако это обычно делается лишь при некоторых процедурах восстановления базы данных.

Процессы инстанции Oracle

Инстанция ORACLE

Независимо от типа компьютера, на котором выполняется ORACLE, и от конкретных используемых опций памяти и процессов, с каждой работающей базой данных ORACLE ассоциируется инстанция ORACLE.

Каждый раз, когда на сервере запускается база данных, происходит распределение области памяти, называемой ***глобальной областью системы (SGA)***, и запускаются один или более процессов ORACLE.

Совокупность SGA и процессов ORACLE называется ***инстанцией базы данных ORACLE*** (см. рис. 7). Структуры памяти и процессы инстанции предназначены для эффективного управления базой данных и обслуживания одного или нескольких пользователей ассоциированной базы данных.

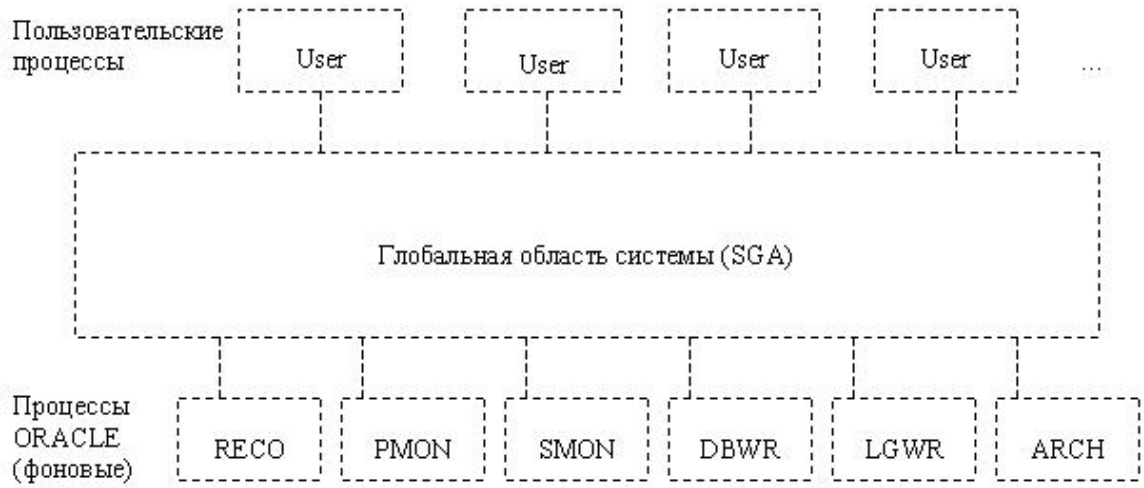


Рисунок 7 - Инстанция ORACLE

Сначала инстанция запускается, затем она монтирует базу данных. На одних и тех же машинах могут быть запущены несколько инстанций, каждая из которых имеет доступ к своей собственной физической базе данных. В слабосвязанных системах используется параллельный сервер ORACLE, в котором одна и та же база данных монтируется несколькими инстанциями; все эти инстанции разделяют единую физическую базу данных.

Процесс - это "канал управления", или механизм в операционной системе, способный

выполнять последовательность шагов. Некоторые операционные системы используют термины "задание" или "задача".

Процесс обычно имеет свою собственную личную область памяти, в которой он выполняется. Структура процесса такой системы, как ORACLE, существенна, потому что она определяет, как осуществляется параллельная деятельность и как она управляется.

Например, двумя целями структуры процесса могут быть:

- симуляция личных окружений для нескольких одновременно работающих процессов, так, как будто каждый процесс имеет свое собственное личное окружение;

- обеспечение разделения между процессами ресурсов компьютера, необходимых каждому процессу, но не на долгое время.

Архитектура процессов ORACLE спроектирована для максимизации производительности.

Однопроцессная инстанция ORACLE

Однопроцессная (также называемая однопользовательской) система ORACLE - это система базы данных, в которой весь код ORACLE исполняется одним процессом. Не используются разные процессы, чтобы разграничить выполнение компонент ORACLE и прикладной программы клиента. Вместо этого весь код ORACLE и единственное приложение базы данных выполняются как единственный процесс.

Единственный процесс исполняет весь код, ассоциированный как с приложением базы данных, так и с ORACLE.

К инстанции ORACLE в однопроцессном окружении может иметь доступ лишь один пользователь; несколько пользователей не могут обращаться к базе данных одновременно.

Например, к ORACLE, выполняющемуся под операционной системой MS-DOS на PC, может иметь доступ лишь один пользователь, потому что MS-DOS не способна выполнять несколько процессов.

Многопроцессная инстанция ORACLE

Многопроцессный ORACLE (называемый также ***многопользовательским***) использует несколько процессов для исполнения различных частей ORACLE, а также отдельный процесс для каждого присоединенного пользователя. Каждый процесс в многопроцессном ORACLE выполняет специфическую задачу. Благодаря разделению работы ORACLE и приложений базы данных на несколько процессов, несколько пользователей и приложений могут одновременно присоединяться к единственной инстанции базы данных, в то время как система поддерживает отличную производительность.

Большинство систем баз данных - многопользовательские, ибо одним из основных преимуществ СУБД является управление данными, с которыми много пользователей работают одновременно.

Каждый присоединенный пользователь имеет отдельный пользовательский процесс, а для выполнения ORACLE используются несколько фоновых процессов. В многопроцессной системе все процессы можно разделить на две группы: пользовательские процессы и процессы ORACLE.

Пользовательские процессы

Когда пользователь запускает прикладную программу, такую как программу Pro*C, или инструмент ORACLE, такой как SQL*DBA, для выполнения приложения пользователя создается пользовательский процесс.

В многопроцессных системах, ORACLE функционирует через два типа процессов ORACLE: процессы сервера и фоновые процессы.

Процессы сервера создаются для обработки запросов от пользовательских процессов, присоединяемых к инстанции. Часто, когда приложение и ORACLE работают не через сеть, а на одной и той же машине, пользовательский процесс и соответствующий ему процесс сервера комбинируются в единый процесс, чтобы уменьшить накладные расходы системы. Однако, если приложение и ORACLE работают на разных машинах, пользовательский процесс взаимодействует с ORACLE через отдельный процесс сервера.

Процессы сервера (или серверная порция в комбинированном процессе пользователь/сервер), создаваемые от имени приложения каждого пользователя, могут выполнять одну или несколько из следующих задач:

- разбор и исполнение предложений SQL, выдаваемых приложением;

- считывание необходимых блоков данных с диска (из файлов данных) в разделяемые буфера базы данных в SGA, если этих блоков еще нет в SGA;

- возврат результатов таким способом, чтобы приложение могло обрабатывать эту информацию.

Для максимизации производительности и координации работы многих пользователей многопроцессная система ORACLE использует несколько дополнительных процессов, называемых ФОНОВЫМИ ПРОЦЕССАМИ. На многих операционных системах фоновые процессы создаются автоматически при запуске инстанции. На других операционных системах процессы сервера создаются как часть инсталляции ORACLE.

Инстанция ORACLE может иметь большое число фоновых процессов; не все из них присутствуют всегда. Эти процессы имеют следующие имена в инстанции ORACLE:

- процесс записи в базу данных (DBWR);

- процесс записи в журнал (LGWR);

- контрольная точка (CKPT);

- монитор системы (SMON);

- монитор процессов (PMON);

- архиватор (ARCH);

- восстановитель (RECO);

- блокировка (LCKn);

- диспетчер (Dnnn);

- сервер (Snnn).

Процессы инстанции Oracle

Взаимодействие фоновых процессов с различными частями базы данных ORACLE проиллюстрировано на рисунке 8.

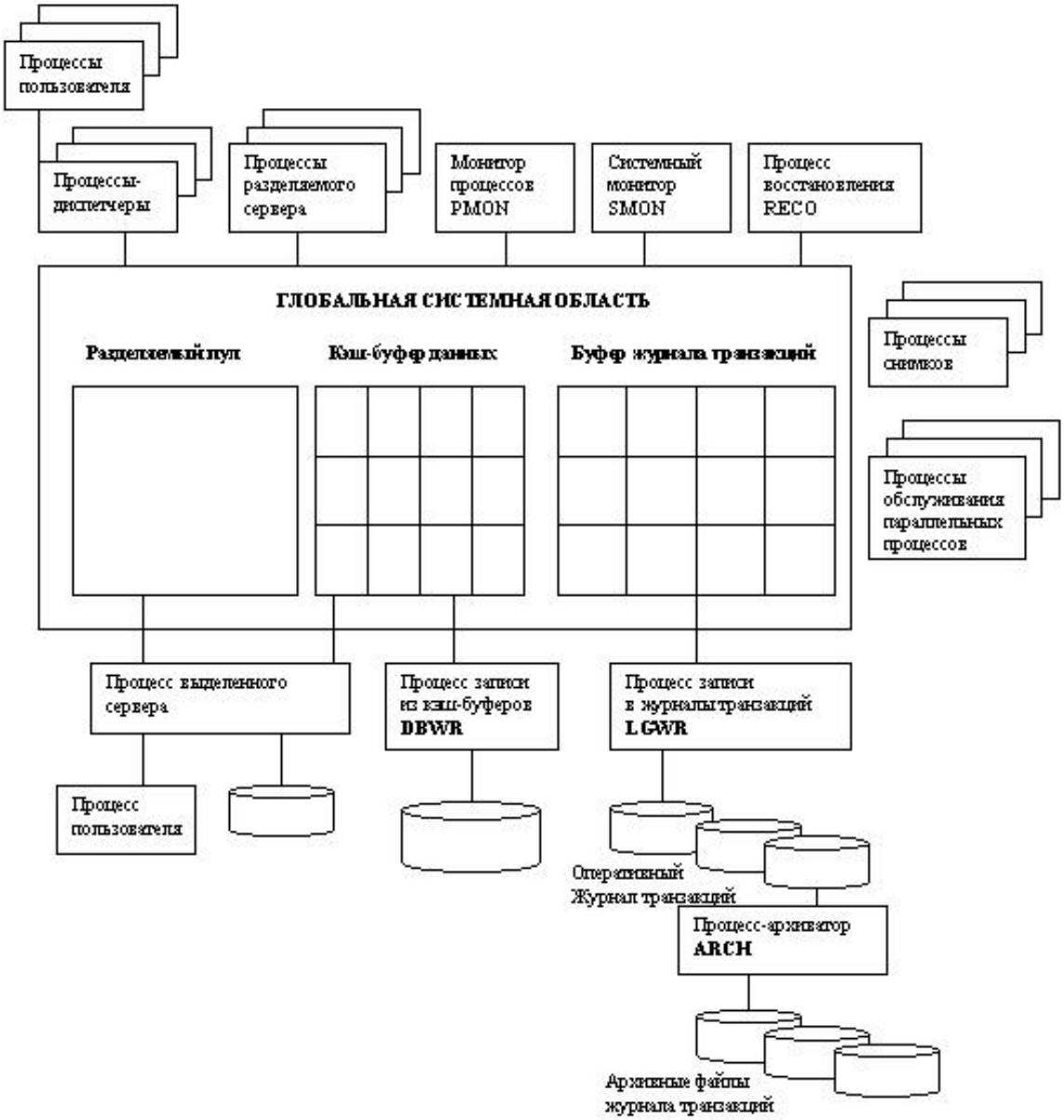


Рисунок 8 - Фоновые процессы много процессной инстанции ORACLE

Процесс записи в базу данных (DBWR)

Вся запись буферов в файлы данных выполняется процессом DBWR, фоновым процессом ORACLE, ответственным за управление буферным кэшем. Когда буфер в буферном КЭШе модифицируется, он помечается как "грязный"; основная ***задача процесса DBWR*** - поддержание "чистоты" буфера путем записи грязных буферов на диск. По мере того, как буфера в КЭШе заполняются из базы данных и загрязняются пользовательскими процессами, число свободных буферов убывает. Когда число свободных буферов становится слишком маленьким, пользовательские процессы, которые должны считывать в кэш блоки данных с диска, не могут найти свободные буфера. DBWR управляет буферным КЭШем так, чтобы пользовательские

процессы всегда могли найти свободные буфера.

Алгоритм LRU поддерживает в памяти блоки данных, которые использовались наиболее недавно, и тем самым минимизирует ввод-вывод. Часто используемые блоки, например, блоки, входящие в часто используемые небольшие таблицы или индексы, постоянно поддерживаются в кэше, так что их не приходится все время считывать с диска. Блоки, используемые не столь часто, например, части больших таблиц или блоки листьев из очень больших индексов, удаляются из SGA, если они не были модифицированы после их считывания, или записываются на диск после их модификации, чтобы освободить в буферном кэше место для других блоков. Схема LRU заставляет наиболее часто используемые блоки оставаться в буферном кэше, так что, когда блок записывается на диск, маловероятно, что он вскоре потребуется снова. Однако, если процесс DBWR становится слишком активным, он может записывать на диск и те блоки, которые могут потребоваться снова.

Процесс DBWR получает сигнал о том, что надо записывать грязные блоки, при следующих условиях:

- Когда процесс сервера перемещает буфер в "грязный список" и обнаруживает, что этот список достиг пороговой величины, он сигнализирует процессу DBWR. Пороговая величина определяется как половина значения параметра DB_BLOCK_WRITE_BATCH.

- Когда процесс сервера просмотрел в списке LRU число буферов, равное DB_BLOCK_MAX_SCAN_CNT, и не нашел свободного буфера, он прекращает поиск и сигнализирует процессу DBWR (ибо не хватает свободных буферов, и DBWR должен освободить часть буферов).

- Когда истекает таймаут (каждые три секунды), DBWR сигнализирует самому себе.

- Когда возникает контрольная точка, процесс записи журнала (LGWR) сигнализирует процессу DBWR.

В первых двух случаях DBWR записывает блоки из грязного списка на диск посредством одной МНОГОБЛОЧНОЙ ОПЕРАЦИИ ЗАПИСИ. Число блоков, записываемых в одной операции, специфицируется параметром инициализации DB_BLOCK_WRITE_BATCH. Если в грязном списке еще нет DB_BLOCK_WRITE_BATCH буферов, когда DBWR принял сигнал, то DBWR просматривает список LRU в поисках дополнительных грязных буферов. ТАЙМАУТ возникает, если DBWR неактивен в течение трех секунд. В этом случае DBWR отыскивает указанное число буферов в списке LRU и записывает все найденные грязные буфера на диск. При каждом таймауте DBWR ищет новую группу буферов. Число буферов, которое DBWR просматривает при таймауте, равно удвоенному значению параметра инициализации DB_BLOCK_WRITE_BATCH. Если база данных простаивает, DBWR в конечном счете переписывает на диск весь буферный кэш.

Когда возникает ***контрольная точка***, процесс записи журнала (LGWR) специфицирует список модифицированных буферов, которые должны быть записаны на диск. DBWR переписывает указанные буфера на диск.

Процесс записи в журнал (LGWR)

Буфер журнала повторения записывается в файл журнала повторения на диск процессом записи в журнал (LGWR), фоновым процессом ORACLE, ответственным за управление буфером журнала повторения.

Процесс LGWR записывает все записи повторения, которые были скопированы в буфер после последней операции его записи. LGWR записывает на диск один непрерывный участок буфера.

Процесс LGWR записывает:

- запись подтверждения, когда пользовательский процесс подтверждает транзакцию;

- буфера повторения каждые три секунды;

- буфера повторения, когда буфер журнала повторения заполняется на одну треть;

- буфера повторения, когда процесс DBWR записывает на диск модифицированные буфера.

LGWR осуществляет синхронную запись в активную зеркальную группу файлов журнала повторения. Если один из файлов в этой группе запорчен или недоступен, LGWR может продолжать запись в остальные файлы (ошибка также регистрируется в файле трассировки LGWR и системном файле ALERT). Если запорчены все файлы в группе, или группа недоступна, потому что она не была архивирована, то LGWR не может продолжать функционировать.

Иногда, если требуется больше места в буфере, LGWR записывает записи журнала повторения до того, как транзакция подтверждена. Эти записи становятся постоянными лишь в случае подтверждения транзакции.

Контрольная точка (CKPT)

Когда возникает контрольная точка, заголовки всех файлов данных должны быть обновлены, чтобы отразить эту контрольную точку. В обычных обстоятельствах эту работу выполняет LGWR. Однако, если контрольные точки значительно понижают производительность системы (обычно при наличии многих файлов данных), то вы можете задействовать процесс CKPT, чтобы отделить выполнение контрольных точек от другой работы, выполняемой процессом LGWR. Для большинства приложений процесс CKPT не является необходимым. Если ваша база данных имеет много файлов данных, и производительность процесса LGWR существенно снижается во время контрольных точек, вы можете включить процесс CKPT.

Процесс CKPT не записывает блоки на диск; эту работу всегда выполняет DBWR. Статистика DBWR CHECKPOINTS, которую выдает монитор статистики SQL*DBA, показывает число завершенных сообщений контрольной точки, независимо от того, задействован ли процесс CKPT. Процесс CKPT включается или отключается параметром инициализации

CHECKPOINT_PROCESS; его умалчиваемое значение FALSE.

Монитор системы (SMON)

Процесс монитора системы (SMON) выполняет восстановление инстанции при запуске инстанции. SMON также отвечает за очистку временных сегментов, когда они больше не используются; он также сжимает непрерывные свободные экстенты, чтобы сделать доступным большее количество блоков свободной памяти. В среде Параллельного сервера SMON осуществляет восстановление инстанции для сбившегося процессора или инстанции.

SMON регулярно "просыпается", чтобы проверить необходимость своей работы; он может также быть вызван, если другой процесс обнаружит необходимость в нем.

Монитор процессов (PMON)

Монитор процессов (PMON) выполняет восстановление процесса, когда пользовательский процесс сбивается. PMON отвечает за очистку кэша и освобождение ресурсов, использовавшихся процессом. Например, он сбрасывает состояние таблицы активных транзакций, освобождает блокировки и удаляет ID процесса из списка активных процессов.

PMON также периодически проверяет состояние серверных и диспетчерских процессов и перезапускает погибшие процессы (но не те, которые ORACLE уничтожил преднамеренно). Как и SMON, PMON регулярно "просыпается", чтобы проверить необходимость своей работы; он может также быть вызван, если другой процесс обнаружит необходимость в нем.

Восстановитель (RECO)

Восстановитель (RECO) - это процесс, используемый с распределенной опцией ORACLE, которая автоматически исправляет сбои, в которых участвуют распределенные транзакции. Фоновый процесс RECO на узле автоматически соединяется с базами данных, участвовавшими в сомнительной распределенной транзакции. Когда соединение между соответствующими серверами баз данных восстановлено, процессы RECO автоматически разрешают все сомнительные транзакции. Строки, соответствующие каждой разрешаемой сомнительной транзакции, автоматически удаляются из таблицы ожидающих транзакций каждой базы данных. Если фоновый процесс RECO сервера базы данных пытается установить соединение с удаленным сервером, а этот удаленный сервер оказывается недоступным в сети, или соединение не восстанавливается, то RECO автоматически повторяет попытку соединения через интервал времени. Интервалы между последовательными попытками соединения экспоненциально увеличиваются.

Архиватор (ARCH)

Архиватор (ARCH) копирует онлайновые файлы журнала повторения на указанное устройство памяти, когда они заполняются. ARCH присутствует лишь в том случае, когда журнал повторения используется в режиме ARCHIVELOG и включено автоматическое архивирование.

Блокировка (LCKn)

В среде Параллельного сервера до десяти процессов ***блокировки*** (LCK0, ..., LCK9) могут использоваться для блокировок между инстанциями; однако для большинства систем параллельного сервера достаточно одного процесса блокировки (LCK0).

Диспетчерские процессы (Dnnn)

Диспетчерские процессы позволяют пользовательским процессам разделять ограниченное число серверных процессов. Без диспетчера, каждому пользовательскому процессу требуется свой выделенный процесс сервера; однако в конфигурации многоканального сервера несколько разделяемых серверных процессов могут обслуживать такое же число пользователей. Поэтому в системе с большим количеством пользователей многоканальный сервер может поддерживать большее число пользователей, особенно в окружениях клиент-сервер, когда приложение-клиент и сервер работают на разных машинах. Для одной инстанции базы данных может быть создано несколько диспетчерских процессов; по меньшей мере один диспетчер должен быть создан для каждого сетевого протокола, через который пользователи соединяются с ORACLE. Администратор базы данных должен запустить оптимальное число диспетчерских процессов, в зависимости от ограничения операционной системы для числа соединений ни один процесс, и может добавлять и удалять диспетчерские процессы во время работы инстанции.

Многоканальный сервер требует наличия SQL*Net версии 2 или более поздней. Каждый пользовательский процесс, соединяющийся с диспетчером, должен делать это через SQL*Net, даже если оба процесса выполняются на одной и той же машине. В конфигурации многоканального сервера, процесс слушателя сети ожидает запросов на соединение от приложений-клиентов, и направляет каждый такой запрос диспетчерскому процессу. Если слушатель не может соединить запрос со свободным диспетчером, он запускает выделенный серверный процесс и соединяет клиента с этим процессом. Процесс слушателя не является частью инстанции ORACLE; скорее это часть сетевых процессов, работающих с ORACLE.

Когда запускается инстанция, слушатель открывает и устанавливает коммуникационный путь, через который клиенты соединяются с ORACLE. Затем каждый диспетчер сообщает слушателю адрес, по которому этот диспетчер слушает запросы на соединение. Когда пользовательский процесс выдает запрос на соединение, слушатель исследует этот запрос и определяет, может ли данный пользователь использовать диспетчера. Если да, то процесс слушателя возвращает адрес диспетчерского процесса с наименьшей загрузкой, и пользовательский процесс непосредственно соединяется с этим диспетчером.

Некоторые пользовательские процессы не могут общаться с диспетчером (например,

пользователи, соединяющиеся через SQL*Net более старой версии, чем 2.0), и процесс сетевого слушателя не может соединять таких пользователей с диспетчером. В этом случае слушатель создает выделенный серверный процесс и устанавливает требуемое соединение.

Структуры памяти ORACLE

ORACLE использует память для хранения следующей информации:

- исполняемого программного кода;

- информации о присоединенной сессии, даже если она неактивна в данный момент;

- данных, необходимых для выполнения программы (например, текущего состояния запроса, из которого извлекаются строки данных);

- информации, которая разделяется и циркулирует между процессами ORACLE (например, информации блокировок);

- кэшируемой информации, которая постоянно хранится в периферийной памяти (например, блоков данных).

Основные структуры памяти, ассоциированные с ORACLE (см. рис. 2), включают:

- области программных кодов;

- глобальную область системы (SGA);

- кэш буферов базы данных;

- буфер журнала повторения;

- разделяемый пул;

- глобальные области программ (PGA);

- области стеков;

- области данных;

- области сортировки.

Рассмотрим более подробно эти структуры памяти.

Виртуальная память

На многих операционных системах ORACLE использует преимущества *виртуальной памяти*. ***Виртуальная память*** - это средство операционной системы, которое предоставляет программам больше памяти, чем реально имеется в машине, и обеспечивает больше гибкости в использовании основной памяти. Виртуальная память симулирует память машины, используя комбинацию реальной (основной) памяти и вторичной памяти (обычно на диске). Операционная система осуществляет доступ к виртуальной памяти прозрачно для прикладных программ.

Обычно лучше держать всю область SGA в реальной памяти. Программы и остальные части ORACLE могут использовать виртуальную память.

Области программных кодов

Области программных кодов - это участки памяти, используемые для хранения кода, который выполняется или может быть выполнен. Код ORACLE хранится в области программного кода, которая обычно отделена от программ пользователей (например, защищена). Области программных кодов обычно статичны, изменяясь лишь при обновлении или повторной инсталляции программного обеспечения. Требуемый размер различен для разных операционных систем.

Программные области являются только-читаемыми, и могут быть инсталлированы как разделяемые или неразделяемые. Когда это возможно, код ORACLE разделяется, так что все пользователи ORACLE могут обращаться к нему, не требуя его дополнительных копий в памяти. Это экономит реальную основную память и улучшает общую производительность.

Пользовательские программы могут быть разделяемыми или неразделяемыми; некоторые инструменты и утилиты ORACLE могут быть инсталлированы как разделяемые (SQL*Forms, SQL*Plus и т.д.), тогда как другие не могут. Несколько инстанций ORACLE, работающих с разными базами данных на одном и том же компьютере, могут разделять один и тот же код ORACLE. Однако, инсталляция программного обеспечения как разделяемого возможна не во всех операционных системах. Например, это невозможно на PC под управлением MS-DOS.

Глобальная область системы (SGA)

Глобальная область системы (SGA) - это группа структур разделяемой памяти, распределяемой ORACLE, которая поддерживает данные и управляющую информацию для одной инстанции ORACLE.

Если к одной инстанции одновременно присоединяются несколько пользователей, то данные в области SGA этой инстанции "разделяются" между всеми пользователями. Поэтому SGA и называется глобальной областью системы, или разделяемой глобальной областью.

Память для SGA автоматически распределяется при запуске инстанции и освобождается при закрытии инстанции. Каждая запускаемая инстанция имеет свою собственную SGA.

SGA - это область разделяемой памяти; все пользователи, присоединенные к многопользовательской инстанции базы данных, могут использовать информацию в SGA этой инстанции. Область SGA является обновляемой; во время выполнения ORACLE в нее ведут запись несколько процессов. Информация, хранящаяся в SGA, разделяется на несколько областей памяти, включая:

- буферный кэш базы данных;

- буфер журнала повторения;

- разделяемый пул;

- очереди запросов и ответов (если используется многоканальный сервер);

- кэш словаря данных;

- прочую смешанную информацию.

Буферный кэш базы данных

Одну часть SGA составляет кэш, в котором хранится информация базы данных. Буфера в буферном кэше содержат копии блоков данных, прочитанных из файлов данных. Эти буфера разделяются всеми пользовательскими процессами ORACLE, присоединенными к инстанции.

Буфера в кэше организованы в два списка: грязный список и список LRU (Least Recently Used). Грязный список содержит грязные буфера, т.е. буфера, которые были модифицированы, но еще не записаны на диск. Список LRU содержит свободные буфера, прикрепленные буфера и грязные буфера, которые еще не перемещены в грязный список. Свободные буфера - это буфера, которые не были модифицированы и доступны для использования. Прикрепленные буфера - это буфера, к которым осуществляется обращение в текущий момент. Когда процесс ORACLE обращается к буферу, он перемещает его в конец списка LRU. По мере непрерывных перемещений буферов в конец списка LRU грязные буфера "стареют" и сдвигаются к началу этого списка. Когда пользовательскому процессу требуется блок, которого еще нет в буферном кэше, он должен прочитать этот блок из файла данных на диске в буфер в кэше. Перед этим процесс должен отыскать свободный буфер. Процесс просматривает список LRU с начала. Поиск продолжается, пока не будет найден свободный буфер, или пока не будет просмотрено число буферов, специфицированное параметром DB_BLOCK_MAX_SCAN_CNT. В процессе поиска свободного буфера процессу могут встречаться грязные буфера. Процесс перемещает их в грязный список и продолжает поиск. Когда процесс находит свободный буфер, он считывает в него блок данных и перемещает его в конец списка LRU.

Если пользовательский процесс ORACLE просмотрел в списке LRU число буферов, равное DB_BLOCK_MAX_SCAN_CNT, но не нашел свободного буфера, то он прекращает поиск и сигнализирует процессу DBWR, чтобы записать часть грязных буферов на диск.

Буфер журнала повторения

Буфер журнала повторения - это циклический буфер в SGA, который содержит информацию об изменениях, выполненных в базе данных. Эта информация сохраняется в *записях повторения*. Записи повторения содержат информацию, необходимую для повторения, реконструирования изменений, осуществляемых в базе данных операциями INSERT, UPDATE, DELETE, CREATE, ALTER или DROP; эти записи используются для восстановления базы данных, если необходимо.

Записи повторения копируются серверными процессами ORACLE из памяти пользовательских процессов в буфер журнала повторения в SGA. Записи повторения занимают в этом буфере непрерывное, последовательное пространство. Буфер журнала повторения записывается в активную группу онлайновых файлов журнала повторения на диске фоновым процессом LGWR.

Размер буфера журнала повторения (в байтах) определяется значением параметра инициализации LOG_BUFFER. В общем случае, большие значения уменьшают ввод-вывод, особенно если транзакции длинны или многочисленны. Умалчиваемое значение равно четырехкратному размеру блока данных в вашей операционной системе (не в базе данных).

Разделяемый пул

Разделяемый пул - это область в SGA, содержащая такие конструкты, как разделяемые области SQL и кэш словаря данных.

Разделяемые области SQL и личные области SQL

ORACLE представляет каждое исполняемое им предложение SQL в виде разделяемой части (разделяемой области SQL) и личной части (личная область SQL). ORACLE распознает, когда два пользователя выполняют одинаковое предложение SQL, и использует одну и ту же разделяемую область SQL для обоих. Однако каждый пользователь должен иметь отдельную копию для личной части предложения.

Разделяемая область SQL - это область памяти, которая содержит дерево разбора и план исполнения для одного предложения SQL. Размер разделяемой области SQL вычисляется во время разбора предложения и зависит от сложности этого предложения.

Разделяемая область SQL всегда находится в разделяемом пуле, и используется совместно всеми идентичными предложениями SQL.

Личная область SQL - это область памяти, содержащая такие данные, как информация привязки и буфера времени выполнения. Каждая сессия, выдавшая предложение SQL, имеет личную область SQL. Каждый пользователь из тех, кто выдал идентичное предложение SQL,

имеет свою собственную личную область SQL, использующую одну разделяемую область SQL; таким образом, с одной и той же разделяемой областью SQL может быть ассоциировано много личных областей SQL.

Личная область SQL далее разбивается на постоянную область и область времени выполнения:

а) *Постоянная область* содержит информацию привязки, область, которая остается одной и той же между выполнениями. Размер этой области постоянен, и зависит от числа привязок и столбцов, исопльзуемых в предложении. Например, постоянная область тем больше, чем больше столбцов содержит запрос.

б) *Область времени выполнения* содержит информацию, времени используемую во время исполнения предложения выполнения SQL. Размер этой области зависит от типа и сложности исполняемого предложения SQL, а также от размеров строк, обрабатываемых этим предложением. В общем случае, область времени выполнения несколько больше для предложений INSERT, UPDATE и DELETE, чем для предложения SELECT. Для предложений INSERT, UPDATE и DELETE эта область освобождается после выполнения предложения. Для запросов, эта область освобождается лишь после того, как извлечены все строки, или запрос снят.

Личная область SQL продолжает существовать до тех пор, пока не будет закрыт соответствующий курсор. Так как область времени выполнения освобождается по концу предложения, обычно остается ждать лишь постоянная область. Поэтому при разработке приложений закрывайте все открытые курсоры, которые не будут больше использоваться, чтобы минимизировать объем памяти, требуемый для приложения. Местоположение личной области SQL варьируется в зависимости от типа соединения, установленного для сессии.

Если сессия соединена через выделенный сервер, личные области SQL размещаются в пользовательской области PGA. Однако, если сессия соединена через многоканальный сервер, то ее постоянные области, а также, для предложений SELECT, области времени выполнения, хранятся в SGA.

За управление личными областями SQL отвечает пользовательский процесс. Распределение и освобождение личных областей SQL в большой степени зависит от того, какой инструмент вы используете, хотя число личных областей SQL, которые может распределить пользовательский процесс, всегда лимитируется параметром инициализации OPEN_CURSORS. Умалчиваемое значение этого параметра равно 50.

Программные единицы PL/SQL и разделяемый пул

Обработка программных единиц PL/SQL (процедур, функций, пакетов, анонимных блоков и триггеров базы данных) осуществляется ORACLE в манере, аналогичной обработке индивидуальных предложений SQL. Для хранения разобранной, откомпилированной формы программной единицы распределяется разделяемая область SQL. Личная область SQL распределяется для хранения данных, специфичных для сессии, исполняющей программную

единицу, в том числе для значений локальных переменных. Если несколько пользователей одновременно выполняют одну и ту же программную единицу, то все они используют одну и ту же разделяемую область SQL, но отдельные копии личной области SQL, в которых содержатся значения, специфичные для сессии.

Индивидуальные предложения SQL, содержащиеся в программе PL/SQL, обрабатываются как описано в предыдущих секциях. Несмотря на свое происхождение из программной единицы PL/SQL, такие предложения SQL используют разделяемые области SQL для хранения своих разобранных представлений, и личные области SQL для каждой сессии, выполняющей предложение. Единая разделяемая область SQL, используемая для программной единицы PL/SQL, часто называется "родительским курсором", чтобы отличать ее от разделяемых областей SQL, используемых предложениями SQL в этой программной единице, и называемых "порожденными курсорами".

Кэш словаря данных

Словарь данных - это коллекция таблиц и обзоров, содержащих справочную информацию о базе данных, ее структурах и ее пользователях. Среди данных, хранящихся в словаре данных, есть следующие:

- имена всех таблиц и обзоров в базе данных;

- имена и типы данных столбцов в таблицах базы данных;

- привилегии всех пользователей ORACLE.

Эта информация полезна как справочный материал не только для администраторов и разработчиков приложений, но и для конечных пользователей. Сам ORACLE часто обращается к словарю данных во время разбора предложений SQL. Доступность словаря является существенной для работоспособности ORACLE.

Так как ORACLE столь часто обращается к словарю данных, в памяти выделяется специальная область для хранения данных словаря. Эта область называется ***кэшем словаря данных***, или, иногда, ***кэшем строк***. Кэш словаря данных разделяется всеми пользовательскими процессами ORACLE. Кэш словаря данных содержится в разделяемом пуле внутри SGA. Его размер, вместе с размером библиотечного кэша, лимитируется суммарным размером разделяемого пула.

Распределение и освобождение памяти в разделяемом пуле

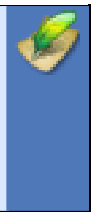
В общем, любой элемент в разделяемом пуле (разделяемая область SQL или строка словаря) сохраняется в пуле до тех пор, пока не будет вытеснен в соответствии с модифицированным алгоритмом LRU; память, занимаемая нерегулярно использовавшимися элементами, освобождается, если для новых элементов, которым требуется пространство в разделяемом пуле, не хватает места. Благодаря использованию модифицированного алгоритма

LRU элементы разделяемого пула, которые совместно используются несколькими сессиями, могут оставаться в памяти столько, сколько нужно, даже если процесс, первоначально создавший такой элемент, уже прекращен. Как следствие, накладные расходы и обработка предложений SQL, ассоциированных с многопользовательской системой ORACLE, сводятся к минимуму.

Когда предложение SQL предоставляется ORACLE для исполнения, есть несколько шагов, заслуживающих рассмотрения. От имени каждого предложения SQL, ORACLE выполняет следующие шаги по распределению памяти:

а) От имени сессии распределяется личная область SQL. Точное местоположение этой области зависит от вида соединения .

б) ORACLE проверяет разделяемый пул, чтобы отыскать в нем разделяемую область SQL для идентичного предложения SQL. Если такая область уже существует, она используется для всех последующих новых экземпляров данного предложения. Поэтому, вместо того чтобы иметь несколько разделяемых областей SQL для идентичных предложений SQL, лишь одна разделяемая область SQL используется для всех идентичных предложений манипулирования данными, что ведет к большой экономии памяти, особенно когда много пользователей выполняют одно и то же приложение. Альтернативно, если для предложения не найдена разделяемая область SQL, для него в разделяемом пуле распределяется новая разделяемая область SQL. В любом случае, с разделяемой областью SQL для предложения ассоциируется пользовательская личная область SQL.



Разделяемая область SQL может быть вытеснена из разделяемого пула, даже если она ассоциирована с открытым курсором, который некоторое время не использовался. Если этот открытый курсор будет впоследствии использован для выполнения своего предложения, это предложение будет заново разобрано, и для него будет распределена новая разделяемая область SQL в разделяемом пуле.

Разделяемая область SQL также может быть вытеснена из разделяемого пула при некоторых уникальных обстоятельствах.

Когда команда ANALYZE используется для обновления или удаления статистики по таблице, кластеру или индексу, все разделяемые области SQL, которые содержат предложения, ссылающиеся на анализируемый объект, удаляются из разделяемого пула. Когда вытесненное предложение выполняется в следующий раз, оно разбирается в новой разделяемой области SQL, чтобы отразить новую статистику для объекта. Разделяемые области SQL также зависят от объектов, к которым обращаются соответствующие предложения SQL. Если адресуемый объект модифицируется, разделяемая область SQL помечается как ***недействительная***, и должна быть разобрана заново при очередном выполнении предложения.

Если вы изменяете глобальное имя базы данных, из разделяемого пула удаляется вся информация. Если нужно, администратор может вручную очистить разделяемый пул, чтобы оценить ожидаемую производительность (по отношению к разделяемому пулу, но не к буферному кэшу), не закрывая текущую работающую инстанцию.

Размер SGA определяется при запуске инстанции. Для оптимальной производительности в большинстве систем вся SGA должна размещаться в реальной памяти. Если SGA не помещается целиком в реальной памяти, и для размещения части SGA используется виртуальная память, то общая производительность базы данных может существенно снизиться, потому что порции SGA подвергаются страничному обмену (считываются и записываются на диск) со стороны операционной системы. Основными детерминантами размера SGA являются параметры в файле параметров базы данных. Наибольшее воздействие на размер SGA оказывают следующие параметры:

- DB_BLOCK_SIZE размер в байтах блока данных и буфера базы данных ;

- DB_BLOCK_BUFFERS число буферов базы данных (каждый размером DB_BLOCK_SIZE), распределяемых для SGA (общее количество памяти, выделяемой для буферного кэша в SGA, равно произведению DB_BLOCK_SIZE на DB_BLOCK_BUFFERS);

- LOG_BUFFER размер в байтах буфера журнала повторения;

- SHARED_POOL_SIZE размер в байтах области, выделяемой для разделяемого пула.

Память, распределенная для области SGA инстанции, выдается при запуске инстанции через SQL*DBA. Вы можете также увидеть текущий размер SGA при помощи команды SQL*DBA SHOW SGA.

Глобальная область программы (PGA)

Глобальная область программы (PGA) - это область памяти, содержащая данные и управляющую информацию для одного процесса (серверного или фонового). Поэтому она и называется глобальной областью программы, или глобальной областью процесса.

Содержимое PGA

Область PGA распределяется ORACLE, когда пользовательский процесс соединяется с базой данных и создается сессия, хотя это различается для разных операционных систем и конфигураций.

Область стека

PGA всегда содержит область стека, распределяемую для хранения переменных, массивов и других данных сессии. Информация сессии PGA в инстанции без многоканального сервера требует дополнительной памяти для сессии пользователя, такой как личные области SQL и т.д. В конфигурации многоканального сервера эта дополнительная память распределяется не в PGA, а в SGA.

Разделяемые области SQL

Разделяемые области SQL всегда располагаются в участках разделяемой памяти в SGA (не в PGA), независимо от того, используется ли многоканальный сервер. PGA - это неразделяемая область памяти, в которую процесс может писать. Одна PGA распределяется для каждого процесса сервера; эта PGA монопольно принадлежит данному серверному процессу, и чтение и запись в эту PGA осуществляется лишь кодом ORACLE, действующим от имени этого процесса.

Размер PGA зависит от операционной системы и не является динамическим. Когда клиент и сервер находятся на разных машинах, PGA распределяется на сервере базы данных во время соединения; если для соединения не хватает памяти, возникает ошибка. Это ошибка ORACLE, которая находится в диапазоне номеров ошибок операционной системы. После установления соединения пользователь никогда не может переполнить область PGA; памяти может не хватать лишь в момент установления соединения.

Следующие параметры инициализации влияют на размер PGA:

- OPEN_LINKS;

- DB_FILES;

- LOG_FILES.

Некоторые дополнительные параметры оказывают воздействие на размер области стека в каждой PGA, создаваемой от имени фоновых процессов ORACLE (таких как DBWR и LGWR).

Области сортировки

Для сортировки требуется место в памяти. Участки памяти, в которых ORACLE сортирует данные, называются ОБЛАСТЯМИ СОРТИРОВКИ. Область сортировки существует в памяти пользовательского процесса ORACLE, запрашивающего сортировку. Эта область может расти соответственно объему сортируемых данных, но ограничивается значением параметра инициализации SORT_AREA_SIZE. Это значение выражается в байтах. Умалчиваемое значение зависит от операционной системы.

Создание таблиц с распределением памяти

Объекты схемы. Таблицы

Объекты схемы - это логические структуры памяти данных.

Каждому объекту схемы не соответствует физический файл на диске, в котором содержится информация этого объекта. Однако объект схемы логически хранится в табличном пространстве

базы данных. Данные каждого объекта физически размещаются в одном или нескольких файлах данных табличного пространства. Для некоторых объектов, таких как таблицы, индексы и кластеры, вы можете указать количество дисковой памяти, которое должно быть распределено объекту в файлах данных табличного пространства.

На рисунке 9 показано отношение между объектами, табличными пространствами и файлами данных.

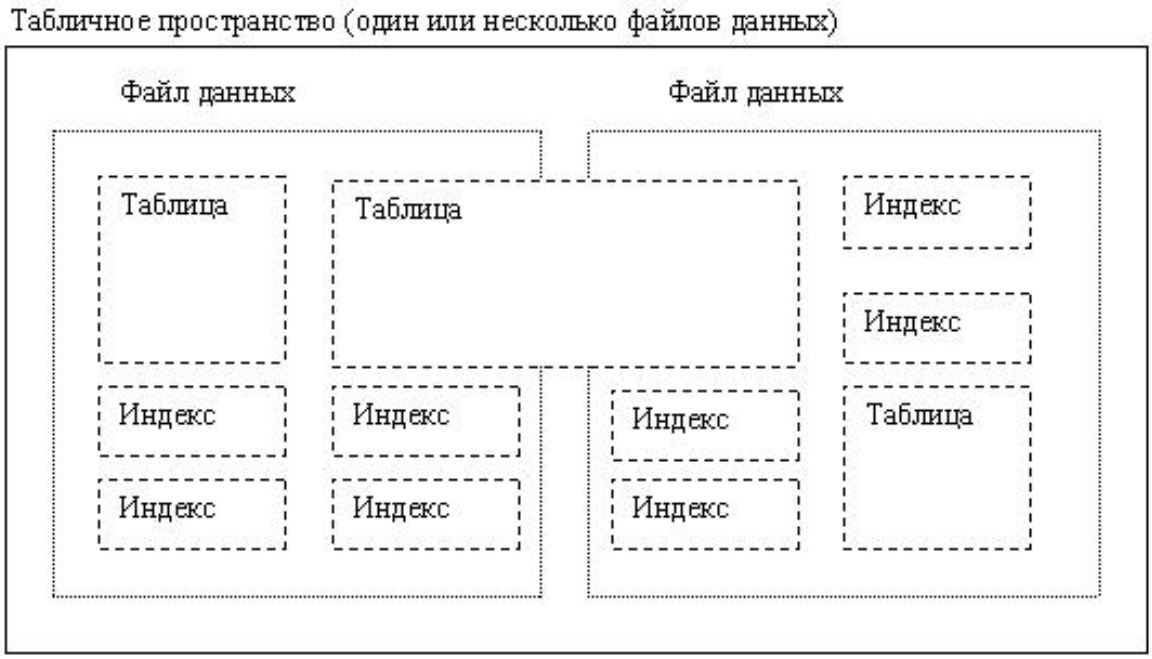


Рисунок 9 - Объекты схемы, табличные пространства и файлы данных

Не существует отношения между схемами и табличными пространствами; табличные пространства могут содержать объекты из разных схем, а объекты одной схемы могут содержаться в разных табличных пространствах.

Таблицы

Таблица - это основная единица памяти данных в базе данных ORACLE.

Данные таблицы хранятся в строках и столбцах. Каждая таблица определяется именем и набором столбцов. Каждому столбцу дается имя столбца, тип данных (такой как VARCHAR2, DATE или NUMBER) и ширина (которая может быть предопределена типом данных, как в случае DATE) или точность и масштаб (только для столбцов типа NUMBER). Строка - это коллекция информации столбцов, соответствующая одной записи.

Можно специфицировать правила для каждого столбца таблицы. Эти правила называются ограничениями целостности. Примером может служить ограничение целостности NOT NULL, которое диктует, что столбец не может содержать пустых значений.

После создания таблицы, в нее можно вставлять строки данных, используя предложения SQL. Затем информацию таблицы можно опрашивать, удалять или обновлять с помощью SQL.

Как хранятся данные таблицы

При создании таблицы в табличном пространстве автоматически создается сегмент данных, в котором будут храниться будущие данные этой таблицы. Вы можете контролировать распределение и использование пространства для сегмента данных таблицы различными способами:

- Можно управлять размерами экстентов для сегмента данных, установив параметры пространства для этого сегмента.

- Можно контролировать использование свободной памяти в блоках данных, составляющих экстенты сегмента данных, установив параметры PCTFREE и PCTUSED для этого сегмента.

Табличное пространство, в котором содержится таблица - это либо умалчиваемое табличное пространство пользователя, либо табличное пространство, явно указанное в предложении CREATE TABLE.

Синтаксис создания таблицы

Синтаксис создания таблицы показан на рисунке 10.

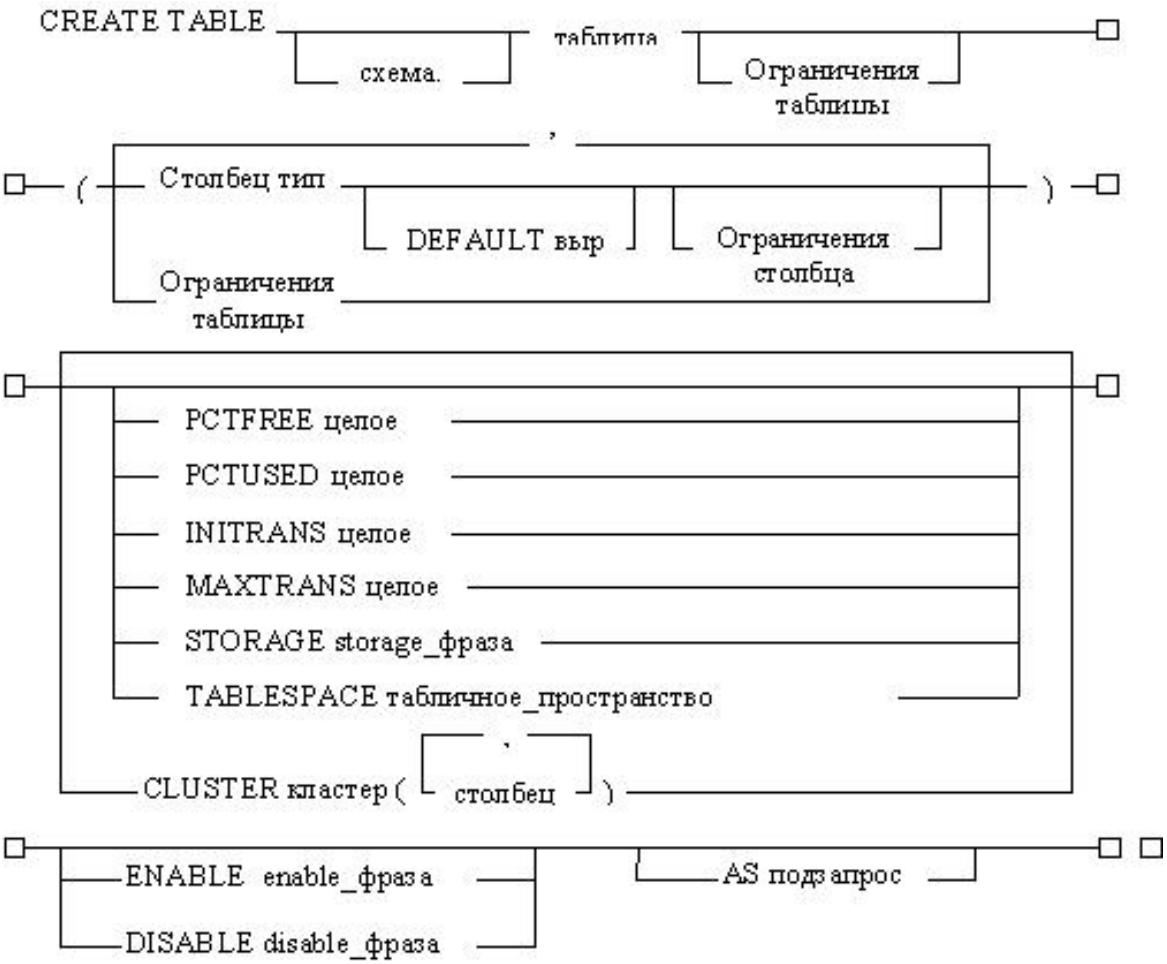


Рисунок 10 - Синтаксис создания таблицы

Некоторые ключевые слова и параметры приведены в таблице 2, об остальных подробно рассказано ниже

Таблица 2 - Ключевые слова команды CREATE TABLE

Ключевые слова	Описание
1	2
Схема	Схема, в которой создается таблица. Если СХЕМА опущена, то ORACLE создает таблицу в вашей схеме
Таблица	Имя создаваемой таблицы.
Столбец	Специфицирует имя столбца таблицы. Количество столбцов в таблице может варьироваться от 1 до 254
Тип	Тип данных столбца.

DEFAULT	Специфицирует умалчиваемое значение, которое должно назначаться столбцу, если в предложении INSERT, вставляющем данные в таблицу, будет отсутствовать значение для этого столбца. Тип данных выражения "выр" должен совпадать с типом данных столбца Выражение DEFAULT не может содержать ссылок на другие столбцы, псевдостолбцов CURRVAL, NEXTVAL, LEVEL и ROWNUM, а также не полностью специфицированных констант дат.
огр_столбца	Определяет ограничение целостности как часть определения столбца
огр_таблицы	Определяет ограничение целостности как часть определения таблицы.
Табличное пространство	Специфицирует табличное пространство, в котором ORACLE создает таблицу. Если эта опция опущена, то ORACLE создает таблицу в умалчиваемом табличном пространстве владельца схемы, в которой создается таблица.
STORAGE	Специфицирует характеристики памяти для таблицы. Эта фраза имеет некоторые варианты производительности для больших таблиц. Следует распределять память так, чтобы минимизировать динамическое распределение дополнительной памяти.
CLUSTER	Указывает, что таблица должна быть частью кластера. Столбцы, перечисленные в этой фразе, называют те столбцы таблицы, которые соответствуют столбцам ключа кластера. Обычно эти столбцы составляют первичный ключ таблицы или часть ее первичного ключа. Столбцы таблицы и кластера сопоставляются по позициям, а не по именам. Поскольку кластеризованная таблица использует распределение памяти кластера, не рекомендуется использование с опцией CLUSTER параметров PCTFREE, PCTUSED, INITRANS и MAXTRANS, а также опции TABLESPACE.
ENABLE	Включает ограничение целостности
DISABLE	Выключает ограничение целостности.
AS подзапрос	Вставляет в таблицу при ее создании строки, возвращаемые подзапросом. При включении этой фразы, определения столбцов могут специфицировать только имена столбцов, их умалчиваемые значения и ограничения целостности, но не типы данных. ORACLE выводит типы данных и длины столбцов из подзапроса. ORACLE также автоматически определяет ограничения NOT NULL по столбцам новой таблицы, если эти ограничения существовали по соответствующим столбцам опрашиваемой таблицы, и если подзапрос не модифицирует значение столбца посредством функции SQL или оператора. Предложение CREATE TABLE не может содержать одновременно фразу AS и определение ограничения ссылочной целостности. Число столбцов в таблице должно совпадать с количеством выражений в подзапросе. Если все выражения в подзапросе представляют собой столбцы, то можно полностью опустить перечень столбцов в определении таблицы. В этом случае имена столбцов таблицы будут такими же, как имена столбцов в подзапросе.

Рассмотрим возможности управления распределением памяти при создании таблиц.

Следующие секции объясняют, как использовать параметры PCTFREE и PCTUSED для выполнения следующих задач:

- повышения производительности записи и извлечения данных или индексов;

- уменьшения объема неиспользуемой памяти в блоках данных;

- уменьшения количества цепочек строк между блоками данных.

Специфицирование PCTFREE

Умолчание для PCTFREE равно 10 процентов; вы можете задавать любое целое значение от 0 до 99 включительно, пока сумма PCTFREE и PCTUSED не превышает 100. (Если вы установите PCTFREE как 99, то ORACLE будет помещать в каждый блок по меньшей мере одну строку, независимо от размера этой строки. Если строки очень малы, а блоки очень велики, может уместиться даже несколько строк.)

Низкое значение PCTFREE имеет следующие эффекты:

- резервирует меньше места для обновлений существующих строк таблицы;

- позволяет более полно заполнять блок вставками;

- может экономить память, так как все данные таблицы или индекса хранятся в меньшем количестве блоков (больше строк на один блок);

- увеличивает стоимость обработки, так как ORACLE вынужден часто реорганизовывать блоки по мере заполнения их свободной памяти новыми или обновленными данными;

- потенциально увеличивает стоимость обработки и требуемую память, если обновления строк или записей индекса приводят к росту строк и расщеплению их между блоками (ибо предложения UPDATE, DELETE и SELECT должны считывать больше блоков для данной строки, следуя по цепочке, связывающей куски строки)

Высокое значение PCTFREE имеет следующие эффекты:

- резервирует больше места для обновлений существующих строк таблицы;

- может потребовать больше памяти для того же количества вставляемых данных (вставляет меньше строк на один блок);

- уменьшает стоимость обработки, так как блоки редко требуют реорганизации своей свободной памяти;

- может улучшить производительность обновлений, потому что ORACLE не должен столь часто, как прежде, строить цепочки для кусков строк.

▼ **Подробнее**

При установке PCTFREE необходимо понимать природу данных таблицы или индекса. Обновления могут приводить к росту строк. Новые значения могут иметь размер, отличный от размера заменяемых ими значений. Если имеют место много обновлений, при которых размер данных увеличивается, то PCTFREE следует увеличить; если обновления существенно не влияют на размеры строк, PCTFREE может быть низким.

Ваша цель - найти удовлетворительный компромисс между плотной упаковкой данных (низкий PCTFREE, заполненные блоки) и хорошей производительностью обновлений (высокий PCTFREE, менее заполненные блоки).

PCTFREE также влияет на производительность запросов данного пользователя по таблицам, имеющим неподтвержденные транзакции от других пользователей (т.е. по таблицам, одновременно обновляемым другими пользователями). Обеспечение согласованности по чтению может потребовать частой реорганизации свободной памяти в блоках, если свободные участки в этих блоках малы.

PCTFREE для таблиц

Если данные в строках таблицы имеют тенденцию к увеличению размера, зарезервируйте часть места для таких обновлений. В противном случае обновления строк будут приводить к расщеплению строк между блоками и ухудшению производительности операций ввода-вывода, связанных с этими строками.

Специфицирование PCTUSED

Когда свободная память в блоке данных падает до PCTFREE, в этот блок не вставляются новые строки, пока процент занятой памяти не упадет ниже PCTUSED. ORACLE старается удержать блок данных заполненным по крайней мере на PCTUSED. Это - процент памяти в блоке, свободной для данных после вычета накладных расходов из общей памяти блока.

Умолчание для PCTUSED равно 40 процентов; вы можете задавать любое целое значение от 0 до 99 включительно, пока сумма PCTFREE и PCTUSED не превышает 100.

Низкое значение PCTUSED имеет следующие эффекты:

- в среднем, удерживает блоки менее заполненными, чем высокий процент PCTUSED;

- уменьшает стоимость обработки, требующейся при UPDATE и DELETE для перемещения блока в свободный список, когда его занятая память падает ниже PCTUSED;

- увеличивает неиспользуемую память в базе данных.

Высокое значение PCTUSED имеет следующие эффекты:

- в среднем, удерживает блоки более заполненными, чем низкий процент PCTUSED;

- улучшает эффективность использования памяти;

- увеличивает стоимость обработки, требующейся при UPDATE и DELETE.

Выбор связанных значений PCTUSED и PCTFREE

Если вы решили явно задать значения PCTUSED и PCTFREE, примите во внимание следующие соображения:

- Сумма PCTFREE и PCTUSED не должна превышать 100.

- Если эта сумма меньше 100, то идеальным компромиссом между утилизацией памяти и производительностью ввода-вывода является случай, когда сумма PCTFREE и PCTUSED отличается от 100 на величину, равную проценту памяти в свободном блоке, занимаемому средней строкой. Например, предположим, что размер блока данных равен 2048 байт; за минусом 100 байт накладных расходов это дает 1948 байт, доступных для данных. Если средняя строка требует 195 байт, или 10% от 1948, то наилучший компромисс даст сумма PCTFREE и PCTUSED, равная 90%.

-Если эта сумма равна 100, то ORACLE пытается удерживать в блоке не больше чем PCTFREE свободной памяти, и стоимость обработки будет максимальной.

- Фиксированные накладные расходы блока не включаются в вычисления PCTUSED и PCTFREE.

- Чем меньше разница между 100 и суммой PCTUSED и PCTFREE (скажем, при PCTUSED=75 и PCTFREE=20), тем выше утилизация памяти, за счет некоторого повышения стоимости обработки.

Примеры выбора значений PCTFREE и PCTUSED

Следующие примеры иллюстрируют подбор значений PCTFREE и PCTUSED при заданных сценариях.

Сценарий 1: Типичная работа включает предложения UPDATE, которые увеличивают размеры строк.

Установка:

PCTFREE = 20

PCTUSED = 40

Объяснение: PCTFREE установлен в 20, чтобы оставить достаточно места для строк, увеличивающихся в размере при обновлениях. PCTUSED установлен в 40, чтобы требовалось меньше обработки при высокой активности обновлений, т.е. для улучшения производительности.

Сценарий 2: Типичная работа включает предложения INSERT и DELETE, а предложения UPDATE в среднем не увеличивают размеры строк.

Установка:

PCTFREE = 5

PCTUSED = 60

Объяснение: PCTFREE установлен в 5, так как большинство предложений UPDATE не увеличивают размеров строк. PCTUSED установлен в 60, так что память, освобождаемая предложениями DELETE, скоро начинает повторно использоваться, так что обработка минимизируется.

Сценарий 3: Таблица очень велика; поэтому основной заботой является память. Типичная работа включает только-читающие транзакции.

Установка:

PCTFREE = 5

PCTUSED = 90

Объяснение: PCTFREE установлен в 5, так предложения UPDATE используются редко. PCTUSED установлен в 90, так что для хранения данных используется большая часть блока. Это значение PCTUSED уменьшает число блоков, требуемое для размещения всех данных таблицы, сокращает среднее число блоков, просматриваемых во время запросов, и тем самым увеличивает производительность запросов.

Установка параметров памяти

Можно устанавливать параметры памяти для следующих типов структур логической памяти:

- табличных пространств (действуют на каждый сегмент в табличном пространстве);

-таблиц, кластеров, снимков и журналов снимков (сегментов данных);

- индексов (сегментов индексов);

- сегментов отката.

Каждая база данных имеет умалчиваемые значения для параметров памяти. Вы можете

специфицировать умолчания для табличного пространства, которые перекроют системные умолчания и станут умолчаниями для объектов, создаваемых в этом конкретном табличном пространстве; кроме того, вы можете явно специфицировать характеристики памяти для каждого индивидуального объекта. Параметры памяти, которые вы можете устанавливать, перечислены ниже, вместе с их системными умолчаниями.

▼ **Подробнее**

INITIAL - размер, в байтах, первого экстента, который распределяется при создании сегмента.

Умолчание: 5 блоков данных.

Минимум: 2 блока данных.

Максимум: зависит от операционной системы.

Хотя системные умолчания приведены в блоках данных, при задании этого значения используйте байты. Можно применять сокращения К или М для обозначения кило- и мегабайтов. Заданное значение округляется ВВЕРХ до ближайшего кратного размеру блока данных, который определяется параметром DB _ BLOCK _ SIZE .

Например, если размер блока данных базы данных равен 2048 байт, то системное умолчание для параметра памяти INITIAL равно 10240 байт. Если при создании табличного пространства вы специфицируете его параметр INITIAL как 20000 (байт), то ORACLE автоматически округляет это значение вверх до 20480 (10 блоков данных).

NEXT - размер, в байтах, следующего (инкрементального) экстента, распределяемого для сегмента. Второй экстент сегмента будет равен заданному значению NEXT . После этого, каждое очередное значение NEXT будет вычисляться как предыдущее значение NEXT , умноженное на (1 + PCTINCREASE /100) .

Умолчание: 5 блоков данных.

Минимум: 1 блок данных.

Максимум: зависит от операционной системы.

Как и для INITIAL , хотя системные умолчания приведены в блоках данных, при задании этого значения используйте байты. Можно применять сокращения К или М для обозначения кило- и мегабайтов. Заданное значение округляется ВВЕРХ до ближайшего кратного размеру блока данных, который определяется параметром DB _ BLOCK _ SIZE .

MAXEXTENTS - общее число экстентов, включая начальный, которое может быть распределено для сегмента.

Умолчание: зависит от размера блока данных и от операционной системы

Минимум: 1 (экстент).

Максимум: зависит от операционной системы.

MINEXTENTS - общее число экстентов, которое должно быть распределено для сегмента при его создании. Это позволяет распределять большое количество памяти во время создания, даже если непрерывного пространства нет.

Умолчание: 1 (экстент).

Минимум: 1 (экстент).

Максимум: зависит от операционной системы.

Если MINEXTENTS больше 1, то во время создания сегмента ему распределяется необходимое число инкрементальных экстентов, с использованием значений INITIAL , NEXT и PCTINCREASE .

Умалчиваемое и минимальное значения MINEXTENTS для сегмента отката всегда равно 2. Вы можете захотеть увеличить значение MINEXTENTS , когда создаете сегмент отката, или если ваша база данных фрагментирована, а вы хотите гарантировать, что у вас хватит места для загрузки всех данных для одной таблицы.

PCTINCREASE - процент, на который увеличивается каждый следующий инкрементальный экстент по отношению к предыдущему инкрементальному экстенту, распределенному для сегмента. Если PCTINCREASE равен 0, то все инкрементальные сегменты будут одного размера. Если PCTINCREASE больше 0, то каждое очередное значение NEXT увеличивается на PCTINCREASE процентов. Значение PCTINCREASE не может быть отрицательным.

Очередное значение NEXT вычисляется как предыдущее значение NEXT , умноженное на (1 + PCTINCREASE /100) и округленное ВВЕРХ до ближайшего кратного размеру блока данных.

Умолчание: 50 (%).

Минимум: 0 (%).

Максимум: зависит от операционной системы.

Для сегментов отката PCTINCREASE всегда имеет нулевое значение. Нельзя задавать PCTINCREASE для сегментов отката.

За счет корректного применения PCTINCREASE вы можете уменьшить фрагментацию сегмента, увеличивая размеры инкрементальных экстентов и сокращая общее число экстентов, распределяемых для сегмента. Сегмент будет содержать меньшее число экстентов большего размера, нежели большее число экстентов меньшего размера.

Если вы измените значение PCTINCREASE для сегмента, то текущее значение NEXT для этого сегмента не измениться; ваше значение повлияет лишь на последующие значения NEXT .

INITRANS - резервирует предопределенный объем памяти для начального числа записей транзакций, которые будут одновременно обращаться к строкам в блоке данных. Память резервируется в заголовках всех блоков данных в ассоциированном сегменте данных для индекса. *Умалчиваемое значение* равно 1 для таблиц (1*23 байта = 23 байта), и 2 для кластеров и индексов.

MAXTRANS . Когда несколько транзакций одновременно обращаются к строкам одного и того же блока данных, для записи каждой транзакции в блоке распределяется память. После исчерпания памяти, зарезервированной параметром INITRANS , память для дополнительных записей транзакций распределяется из свободной памяти блока, если возможно. После распределения эта память фактически становиться постоянной частью заголовка блока. Параметр MAXTRANS служит для того, чтобы ограничить число записей транзакций, которые могут одновременно использовать данные в блоке данных. Этот параметр лимитирует количество свободной памяти, которая может быть распределена для записей транзакций в блоке данных. *Умалчиваемое значение* зависит от операционной системы и от размера блока, и не превышает 255.

Если значение MAXTRANS слишком мало, транзакции, блокированные этим лимитом, должны ожидать, пока другие транзакции завершаться и освободят им место для записей транзакций. Например, если MAXTRANS равно 3, и блок данных уже используется тремя активными транзакциями, то четвертая транзакция, обратившаяся к этому же блоку, будет блокирована до завершения одной из первых трех транзакций.

Привилегии, требуемые для создания таблиц

Чтобы создать новую таблицу в собственной схеме, пользователь должен иметь системную привилегию CREATE TABLE. Чтобы создать таблицу в схеме другого пользователя, пользователь должен иметь системную привилегию CREATE ANY TABLE. Кроме того, владелец таблицы должен иметь квоту для табличного пространства, в котором содержится таблица, либо системную привилегию UNLIMITED TABLESPACE.

Изменение таблиц

Существуют следующие причины, которые могут потребовать изменения таблицы в базе данных ORACLE:

- чтобы добавить один или несколько новых столбцов;

- чтобы добавить одно или несколько ограничений целостности;

- чтобы модифицировать определение существующего столбца (тип данных, длину,

умалчиваемое значение или ограничение целостности NOT NULL);

- чтобы модифицировать параметры использования памяти блока данных таблицы (PCTFREE, PCTUSED);

- чтобы модифицировать характеристики записей транзакций (INITRANS, MAXTRANS);

- чтобы модифицировать параметры памяти (NEXT, PCTINCREASE и т.п.);

- чтобы включить или выключить ограничения целостности или триггеры, ассоциированные с таблицей;

- чтобы удалить ограничения целостности, ассоциированные с таблицей.

При изменении определений столбцов в таблице, вы можете лишь увеличить длину существующего столбца; вы можете уменьшить ее лишь в том случае, если таблица пуста.

Более того, если вы увеличиваете длину столбца с типом данных CHAR, вы должны понимать, что эта операция может потребовать значительного времени и существенной дополнительной памяти, особенно если таблица содержит много строк. Причина в том, что значение CHAR в каждой строке должно быть дополнено пробелами до новой длины столбца.

Для изменения таблицы используется команда SQL ALTER TABLE. Например, следующее предложение изменяет таблицу EMP:

```
ALTER TABLE emp PCTFREE 30 PCTUSED 60;
```

Последствия изменения таблицы:

- Если к таблице добавляется новый столбец, то изначально он пуст. Вы можете добавить новый столбец с ограничением NOT NULL лишь в том случае, если в таблице нет ни одной строки.

- Если обзор или программная единица PL/SQL зависят от базовой таблицы, то изменение этой базовой таблицы может повлиять на зависимый объект, и всегда делает этот зависимый объект недействительным.

Привилегии, требуемые для изменения таблиц

Чтобы изменить таблицу, либо она должна содержаться в собственной схеме пользователя, либо пользователь должен иметь объектную привилегию ALTER для этой таблицы, или системную привилегию ALTER ANY TABLE.

Удаление таблиц

Для удаления ненужной таблицы, используется команду SQL DROP TABLE. Например, следующее предложение удаляет таблицу EMP:

```
DROP TABLE emp;
```

Если удаляемая таблица содержит первичный или уникальный ключ, на который ссылаются внешние ключи других таблиц, то вы можете одновременно с этой таблицей удалить ограничения FOREIGN KEY для порожденных таблиц, включив в команду DROP TABLE опцию CASCADE, например:

```
DROP TABLE emp CASCADE CONSTRAINTS;
```

Эффекты удаления таблицы:

- Удаление таблицы приводит к удалению ее определения из словаря данных. Все строки таблицы необратимо теряются.

- Все индексы и триггеры, ассоциированные с таблицей, также удаляются.

- Все обзоры и программные единицы PL/SQL, зависимые от удаляемой таблицы, остаются, но становятся недействительными (непригодными для использования).

- Все синонимы удаленной таблицы остаются, но возвращают ошибку при обращении к ним.

- Все экстенты, распределенные удаляемой некластеризованной таблице, возвращаются в свободную память табличного пространства и могут использоваться любым другим объектом, требующим новых экстентов.

Привилегии, требуемые для удаления таблиц

Чтобы удалить таблицу, либо она должна содержаться в собственной схеме пользователя, либо пользователь должен иметь системную привилегию DROP ANY TABLE.

Формат и размер строки

Каждая строка таблицы базы данных хранится как один или несколько кусков строки. Если вся строка может быть вставлена в один блок данных, она первоначально вставляется как единственный кусок. Однако, если все данные в строке не могут быть вставлены в один блок данных, или если в результате обновления строки ее данные переполняют ее блок данных, то строка сохраняется как несколько кусков. Блок данных обычно содержит лишь один кусок для

каждой строки данных; все данные строки, которые умещаются в блоке, хранятся в одном куске строки внутри блока. Когда требуется разбить строку на несколько кусков, образуется "цепочка" из кусков этой строки, размещенных в нескольких блоках данных. Куски строки логически сцепляются через идентификаторы строки (ROWID'ы).

Каждый кусок строки, в цепочке он или нет, содержит заголовок строки и данные для всех или некоторых столбцов строки. Индивидуальные значения столбцов также могут занимать несколько кусков строки и, следовательно, блоков данных.

Заголовок строки предшествует данным строки и содержит следующую информацию:

- информацию об этом куске строки;

- информацию о цепочке, если этот кусок в цепочке;

- информацию о столбцах в этом куске строки.

Строка, полностью умещающаяся в одном блоке, имеет не менее чем трехбайтовый заголовок строки. Вслед за информацией заголовка строки каждая строка содержит информацию о столбцах и данные столбцов. На длину столбца требуется один байт, если эта длина не превышает 250 байт, или три байта для столбцов, содержащих более 250 байт; это поле длины столбца непосредственно предшествует данным столбца. Память, занимаемая данными столбца, зависит от типа данных столбца. Если столбец имеет переменную длину, то место, занимаемое значением столбца, может уменьшаться и увеличиваться при обновлениях.

Для экономии места, строка, содержащая только пустые значения, хранится только как нулевая длина столбца, без данных.

На каждую строку используется два байта в оглавлении строк в заголовке блока данных.

Порядок столбцов

Порядок столбцов один и тот же для всех строк в данной таблице. Столбцы обычно хранятся в том порядке, в котором они были перечислены в предложении CREATE TABLE, но это не гарантируется. Например, если вы создаете таблицу со столбцом типа LONG, то ORACLE всегда размещает этот столбец последним. Кроме того, если таблица изменяется так, что к ней добавляется новый столбец, то этот новый столбец становится последним хранящимся столбцом.

В общем случае, вы должны стараться последними размещать те столбцы, которые часто содержат пустые значения, с тем, чтобы строки занимали меньше места. Заметьте, однако, что, если ваша таблица содержит столбец типа LONG, то преимущества от перемещения таких пустых столбцов теряются.

ROWID'ы кусков строк

Каждый кусок строки идентифицируется ее адресом, который называется ROWID. Однажды назначенный ROWID сохраняется за всеми кусками данной строки до тех пор, пока строка не будет удалена, или экспортирована и импортирована с помощью утилит EXPORT и IMPORT.

Поскольку ROWID является постоянным на все время жизни (куска) строки, к нему можно обращаться в предложениях SQL, и он может оказаться полезным в предложениях SELECT, UPDATE и DELETE.

Пустые значения

NULL, или пустое значение, обозначает отсутствие значения в столбце строки. Пустые значения указывают на то, что данные отсутствуют, неизвестны или неприменимы. Пустое значение не может подразумевать никакого значения, в том числе нулевого. Столбец может содержать пустые значения, если для него не было определено ограничение целостности NOT NULL; в этом случае в таблицу нельзя вставить строку, которая содержала бы пустое значение для данного столбца.

Пустые значения хранятся в базе данных, если они попадают между столбцами, имеющими значения. В таких случаях пустое значение занимает один байт. Хвостовые пустые значения не хранятся и не занимают памяти. В таблицах, содержащих много столбцов, те столбцы, которые чаще всего будут пустыми, следует определять в конце, чтобы сэкономить дисковую память.

Пустые значения распознаются в SQL через предикат IS NULL. Функция SQL с именем NVL может использоваться, чтобы преобразовывать пустые значения в непустые. Пустые значения не индексируются, исключая случай, когда столбец ключа кластера имеет пустое значение.

Умалчиваемые значения столбцов

Столбцу таблицы может быть назначено умалчиваемое значение, так что при вставке в таблицу новой строки, если значение этого столбца опущено, ему будет автоматически присваиваться значение по умолчанию. Умалчиваемые значения столбцов работают так, как если бы они были явно специфицированы в предложении INSERT.

Допустимыми умалчивамыми значениями могут быть литералы или выражения, не ссылающиеся на столбец. Умалчиваемые значения могут включать функции SYSDATE, USER, USERENV. Тип данных умалчиваемого литерала или выражения должен быть преобразуемым в тип данных столбца.

Если для столбца явно не определено умалчиваемое значение, то умолчанием для столбца неявно становится пустое значение (NULL).

Умалчиваемые значения столбцов и ограничения целостности

Проверка ограничений целостности происходит после вставки строки с умалчиваемыми значениями для столбцов.

Рекомендации по созданию таблиц:

- Используйте описательные имена для таблиц, индексов и кластеров.

- Согласовывайте сокращения, а также единственную и множественную формы имен таблиц и столбцов.

- Документируйте назначение каждой таблицы и ее столбцов с помощью команды COMMENT.

- Нормализуйте каждую таблицу.

- Выберите правильный тип данных для каждого столбца.

- Определяйте столбцы, которые допускают пустые значения, последними, чтобы экономить дисковую память.

- Кластеризуйте таблицы, когда это целесообразно, чтобы экономить дисковую память и оптимизировать производительность ваших предложений SQL.

Прежде чем создавать таблицу, вы должны также определить, будете ли вы использовать ограничения целостности. Ограничения целостности могут быть определены по столбцам вашей таблицы для того, чтобы автоматически задействовать организационные правила вашей базы данных.

Примеры создания таблиц приведены в практическом задании.

Объекты базы данных

Обзоры (представления)

Обзор, представления (VIEW) - это настроенное представление данных, содержащихся в одной или нескольких таблицах (или других обзорах). Обзор определяется с использованием запроса, и поэтому о нем можно думать как о "хранимом запросе" или о "виртуальной таблице". Обзоры можно использовать в большинстве мест, где могут использоваться таблицы.

Например, таблица имеет несколько столбцов и много строк информации. Если вы хотите, чтобы пользователи видели лишь несколько из этих столбцов, или лишь определенные строки, вы можете создать по этой таблице обзор, к которому будут обращаться пользователи.

Создание обзоров

Создавайте обзоры с помощью команды SQL CREATE VIEW. Вы можете определять обзоры посредством любого запроса, который обращается к таблицам, снимкам или другим обзорам; однако запрос, определяющий обзор, может содержать фразы ORDER BY или FOR UPDATE.

Например, вы можете создать представление, которое содержит информацию о товарах производителя «HONDA», цена которых выше средней цены по товарам в магазине:

```
CREATE VIEW HONDA_PRODUCT AS

SELECT PRODUCT_NAME, PRICE FROM PRODUCTS WHERE MFR=HONDA AND PRICE >
(SELECT MAX(PRICE) FROM PRODUCTS);
```

Пользователь сможет получать информацию из представления через достаточно простой запрос:

```
SELECT * FROM HONDA_PRODUCT;
```

Поскольку обзоры выводятся из таблиц, между обзорами и таблицами много общего. Например, как и таблицы, обзоры могут содержать до 254 столбцов. Обзоры можно опрашивать, а при некоторых ограничениях пользователи могут обновлять обзоры, вставлять в них и удалять из них строки. Все операции, выполняемые на обзоре, в действительности осуществляются над данными базовых таблиц этого обзора, и попадают под действие ограничений целостности и триггеров этих базовых таблиц.

Пространство для обзоров

В отличие от таблицы, обзору не распределяется пространство в базе данных, и обзор в действительности не содержит никаких данных; скорее, обзор определяется как запрос, соответствующий данным из таблиц, на которые он ссылается. Эти таблицы называются БАЗОВЫМИ ТАБЛИЦАМИ обзора. Базовые таблицы, в свою очередь, могут быть как действительными таблицами, так и другими обзорами (в том числе снимками). Поскольку обзор базируется на других объектах, он не требует иной памяти в словаре данных, помимо той, которая необходима для хранения определения обзора (хранимого запроса).

Как используются обзоры

Обзоры дают средство представить данные в иной перспективе, чем они хранятся в базовых таблицах. Обзоры представляют собой очень мощное средство, ибо они позволяют вам настраивать различные представления данных для различных типов пользователей. Обзоры часто используются:

- для обеспечения дополнительного уровня защиты таблицы за счет предоставления доступа

к предопределенному множеству строк и/или столбцов таблицы.

- для скрытия сложности данных. Например, обзор может быть определен как соединение, т.е. коллекция связанных столбцов или строк из нескольких таблиц. Однако обзор скрывает тот факт, что эта информация в действительности исходит из нескольких таблиц.

- для упрощения команд для пользователей. Например, обзоры позволяют пользователю выбирать информацию из нескольких таблиц, не зная, как осуществлять операцию соединения.

- для представления данных с иной точки зрения, чем в базовой таблице. Например, столбцы обзора можно переименовать, не затрагивая столбцов базовой таблицы.

- для изоляции приложений от изменений в определениях базовых таблиц. Например, если определяющий запрос обзора обращается к трем столбцам четырехстолбцовой таблицы, и в эту таблицу добавляется пятый столбец, то определение обзора не изменяется, и все приложения, использующие этот обзор, не затрагиваются.

- для формулирования запроса, который нельзя выразить иначе, чем через обзор. Например, можно определить обзор, который соединяет обзор GROUP BY с таблицей, или обзор, который соединяет обзор UNION с таблицей.

- для сохранения сложных запросов. Например, запрос может выполнять обширные вычисления с информацией таблиц. Благодаря сохранению такого запроса как обзора можно гарантировать, что эти вычисления будут выполняться при каждом обращении к обзору.

Механика обзоров

Определение обзора представлено хранящимся в словаре данных текстом соответствующего запроса. Когда предложение SQL обращается к обзору, ORACLE осуществляет слияние этого предложения с запросом, определяющим обзор, а затем выполняет синтаксический разбор результирующего предложения в разделяемой области SQL и выполняет его. Разбор предложения, обращающегося к обзору, в новой разделяемой области SQL выполняется лишь в том случае, если ни одна существующая разделяемая область SQL не содержит идентичного предложения. Таким образом, использование обзоров не мешает извлекать выгоду из использования разделяемых областей SQL.

Привилегии, требуемые для создания обзоров

Чтобы создать обзор, вы должны удовлетворять следующим требованиям:

- Чтобы создать обзор в вашей схеме, вы должны иметь привилегию CREATE VIEW; чтобы создать обзор в схеме другого пользователя, вы должны иметь системную привилегию CREATE ANY VIEW. Обе эти привилегии могут быть получены вами явно или через роль.

- ВЛАДЕЛЕЦ обзора (т.е. владелец схемы, в которой создается обзор) должен обладать

ЯВНО назначенными привилегиями для доступа к объектам, на которые ссылается определение обзора; эти привилегии НЕ МОГУТ быть получены через роль. Кроме того, работоспособность обзора зависит от привилегий владельца этого обзора. Например, если владелец обзора имеет лишь привилегию INSERT для таблицы SCOTT.EMP, то обзор можно использовать лишь для вставки новых строк в таблицу EMP, но не для запросов по этой таблице и не для обновления или удаления строк в ней.

- Если владелец обзора намеревается предоставить доступ к этому обзору другим пользователям, он должен обладать объектными привилегиями для базовых объектов обзора с опцией GRANT OPTION, или системными привилегиями с опцией ADMIN OPTION.

В противном случае владелец не может дать доступ к обзору другим пользователям.

Замена обзоров

Если вы хотите изменить определение обзора, то обзор должен быть заменен; нельзя изменить определение обзора командой ALTER. Вы можете заменять обзоры следующими способами:

- Вы можете удалить и заново создать обзор. Когда обзор удаляется, все гранты на соответствующие объектные привилегии отзываются от ролей и от пользователей. После пересоздания обзора все привилегии должны быть назначены заново.

- Вы можете переопределить обзор с помощью предложения CREATE VIEW с опцией OR REPLACE. Эта опция заменяет текущее определение обзора и не затрагивает текущих полномочий.

Прежде чем заменять обзор, примите во внимание следующие эффекты:

- Замена обзора лишь заменяет его определение в словаре данных. Никакие объекты, на которые ссылается обзор, не затрагиваются.

- Если ограничение в опции CHECK OPTION ранее было определено, но не включается в новое определение обзора, то это ограничение удаляется.

- Все обзоры и программные единицы PL/SQL, зависимые от заменяемого обзора, становятся недействительными (непригодными для использования).

Привилегии, требуемые для замены обзора

Чтобы заменить обзор, вы должны иметь все привилегии, необходимые для удаления и создания обзора.

Удаление обзоров

Удаляйте обзор с помощью команды SQL DROP VIEW. Например, следующее предложение удаляет обзор с именем SALES_STAFF:

```
DROP VIEW sales_staff;
```

Привилегии, требуемые для удаления обзора

Вы можете удалять любой обзор, содержащийся в вашей схеме. Чтобы удалить обзор в схеме другого пользователя, вы должны иметь системную привилегию DROP ANY VIEW.

Синонимы

Синоним - это алиас (псевдоним) таблицы, обзора, снимка, последовательности, процедуры, функции или пакета. Поскольку синоним - это лишь алиас, он не требует памяти, помимо своего определения в словаре данных.

Создавайте синоним с помощью команды SQL CREATE SYNONYM.

Синонимы часто используются для безопасности и для удобства. Например, они могут делать следующее:

- маскировать имя владельца объекта;

```
CREATE SYNONYM tab1 ON DIANA.MANUFACTURERS;
```

- обеспечивать прозрачность местоположения для удаленных объектов в распределенной базе данных;

```
CREATE SYNONYM tab2 ON CLARC.PRODUCTS@NEWYORK;
```

- упрощать предложения SQL для пользователей базы данных.

```
SELECT * FROM tab1;
```

Синонимы могут быть общими и личными.

Общий синоним принадлежит специальной группе пользователей с именем PUBLIC и доступен любому пользователю в базе данных.

Личный синоним содержится в схеме конкретного пользователя, и доступен только этому пользователю и тем, кому он дал это право.

Синонимы очень полезны как в распределенных, так и в нераспределенных окружениях баз данных, потому что они скрывают идентичность объекта, включая его местоположение в

распределенной системе. Это выгодно, потому что в случае необходимости переименования или перемещения объекта достаточно переопределить лишь его синоним, а приложения, опирающиеся на этот синоним, продолжают функционировать без модификации.

Синонимы также упрощают предложения SQL для пользователей в распределенной системе баз данных. Следующий пример показывает, как и почему общие синонимы часто создаются администратором базы данных, чтобы скрыть идентичность базовой таблицы и упростить предложения SQL.

Предположим следующее:

- Существует таблица с именем SALES_DATA, содержащаяся в схеме пользователя с именем JWARD.

- Привилегия SELECT для таблицы SALES_DATA предоставлена всем пользователям (PUBLIC).

Допустим, пользователь базы данных должен опросить таблицу SALES_DATA с помощью предложения SQL, аналогичного следующему:

```
SELECT * FROM jward.sales_data;
```

Заметьте, что для выполнения запроса необходимо указать как схему, содержащую таблицу, так и имя таблицы.

Предположим, что АБД создает общий синоним с помощью следующего предложения SQL:

```
CREATE PUBLIC SYNONYM sales FOR jward.sales_data;
```

После того как создан общий синоним, пользователь может опрашивать таблицу SALES_DATA простым предложением SQL:

```
SELECT * FROM sales;
```

Заметьте, что общий синоним SALES скрывает как имя таблицы SALES_DATA, так и имя

схемы, в которой содержится эта таблица.

Удаление синонимов

Чтобы удалить ненужный синоним, используйте команду SQL DROP SYNONYM. При удалении личного синонима опустите ключевое слово PUBLIC; при удалении общего синонима укажите ключевое слово PUBLIC. Например, следующее предложение удаляет личный синоним с именем EMP:

```
DROP SYNONYM emp;
```

Следующее предложение удаляет общий синоним с именем PUBLIC_EMP:

```
DROP PUBLIC SYNONYM public_emp;
```

Когда вы удаляете синоним, его определение удаляется из словаря данных. Все объекты, ссылающиеся на удаленный синоним, остаются, но становятся недействительными.

Последовательности

Последовательность (sequence) - это объект базы данных, который генерирует порядковые номера.

При создании последовательности вы можете специфицировать ее начальное значение и приращение.

CURRVAL возвращает текущее значение указанной последовательности. Прежде чем обратиться к CURRVAL в данной сессии, вы должны хотя бы раз обратиться к NEXTVAL, чтобы сгенерировать номер.

Обращение к NEXTVAL записывает текущий порядковый номер в CURRVAL. NEXTVAL выполняет приращение и возвращает очередное значение. Чтобы получить текущее или очередное значение последовательности, вы должны использовать квалифицированную ссылку:

имя_последовательности.CURRVAL;

имя_последовательности.NEXTVAL .

После создания последовательности вы можете использовать ее для генерации уникальных порядковых номеров для обработки ваших транзакций. В следующем примере последовательность используется для того, чтобы вставить один и тот же номер товара в две таблицы:

```
INSERT INTO goods VALUES (goods_seq.NEXTVAL, name1, ...);
INSERT INTO storages VALUES (storage_id1, ware_id1, goods_seq.CURRVAL, ...);
```

Чтобы обратиться к текущему или следующему значению последовательности, принадлежащей схеме другого пользователя, вы должны иметь любо объектную привилегию SELECT по этой последовательности, либо системную привилегию SELECT ANY SEQUENCE, и должны дополнительно квалифицировать эту последовательность именем содержащей ее схемы:

- schema.sequence.CURRVAL ;

- schema . sequence . NEXTVAL .

Чтобы обратиться к значению последовательности, расположенной в удаленной базе данных, вы должны квалифицировать эту последовательность полным или частичным именем связи баз данных:

- schema . sequence . CURRVAL @ dblink ;

- schema . sequence . NEXTVAL @ dblink ;

Вы можете использовать значения CURRVAL и NEXTVAL в следующих местах:

- в списке SELECT предложения SELECT;

- в фразе VALUES предложения INSERT;

- в фразе SET предложения UPDATE.

Вы не можете использовать значения CURRVAL и NEXTVAL в следующих местах:

- в подзапросе;

- в запросе обзора или снимка;

- в предложении SELECT с оператором DISTINCT;

- в предложении SELECT с фразой GROUP BY или ORDER BY;

- в предложении SELECT, объединенном с другим предложением SELECT оператором множеств UNION, INTERSECT или MINUS;

- в фразе WHERE предложения SELECT;

- в умалчиваемом (DEFAULT) значении столбца в предложении CREATE TABLE или ALTER TABLE;

- в условии ограничения CHECK .

При создании последовательности вы можете определить ее начальное значение и приращение между ее значениями. Первое обращение к NEXTVAL возвращает начальное значение последовательности. Последующие обращения к NEXTVAL изменяют значение последовательности на приращение, которое было определено, и возвращают новое значение. Любое обращение к CURRVAL всегда возвращает текущее значение последовательности, а именно, то значение, которое было возвращено последним обращением к NEXTVAL. Заметьте, что, прежде чем обращаться к CURRVAL в вашей сессии, вы должны хотя бы один раз выполнить обращение к NEXTVAL. В одном предложении SQL приращение последовательности может быть выполнено только один раз. Если предложение содержит несколько обращений к NEXTVAL для одной и той же последовательности, то ORACLE наращивает последовательность один раз, и возвращает одно и то же значение для всех вхождений NEXTVAL. Если предложение содержит обращения как к CURRVAL, так и к NEXTVAL, то ORACLE наращивает последовательность и возвращает одно и то же значение как для CURRVAL, так и для NEXTVAL, независимо от того, в каком порядке они встречаются в предложении. К одной и той же последовательности могут обращаться одновременно несколько пользователей, без какого-либо ожидания или блокировки.

Следующий пример выбирает текущее значение последовательности (такую команду можно выполнить, например, в утилите SQL * Plus):

```
SELECT goods_seq.currval FROM DUAL
```

Следующий пример наращивает значение последовательности и использует его для нового товара, вставляемого в таблицу товаров:

```
INSERT INTO goods VALUES (goods_seq.nextval, 'dresses', 0.5, dryness', 154)
```

Следующий пример добавляет новый товар с очередным номером в главную таблицу товаров, а затем добавляет информацию о хранении товара с этим же номером в таблицу с местоположениями товаров:

```
INSERT INTO goods VALUES (goods_seq.nextval, 'dresses', 0.5, dryness', 154)
INSERT INTO storages VALUES (stor_seq.nextval, ware_id1, goods_seq.nextval, quantity1,
SYSDATE, '12.09.04');
```

Команда CREATE SEQUENCE

Команда CREATE SEQUENCE создает последовательность. Чтобы создать последовательность в вашей собственной схеме, вы должны иметь привилегию CREATE SEQUENCE. Чтобы создать последовательность в схеме другого пользователя, вы должны иметь привилегию CREATE ANY SEQUENCE.

Синтаксис

CREATE SEQUENCE команда ::=
CREATE SEQUENCE последовательность [схема]
[INCREMENT BY целое]
[START WITH целое]
[MAXVALUE целое | NOMAXVALUE]
[MINVALUE целое | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE целое | NOCACHE]
[ORDER | NOORDER];

Ключевые слова и параметры

Схема - схема, в которой создается последовательность. Если СХЕМА опущена, ORACLE создает последовательность в вашей схеме.

Последовательность - имя создаваемой последовательности.

INCREMENT BY специфицирует интервал между значениями (номерами) последовательности. Это значение может быть любым положительным или отрицательным целым числом ORACLE, но не может быть нулевым. Если это значение отрицательно, то последовательность убывает. Если это значение положительно, то последовательность возрастает. Если эта фраза опущена, по умолчанию принимается интервал 1.

MINVALUE задает минимальное значение последовательности. NOMINVALUE указывает, что минимальное значение равно 1, если последовательность возрастает, или -10----26, если последовательность убывает.

MAXVALUE задает максимальное значение, которое может быть сгенерировано последовательностью. NOMAXVALUE указывает, что максимальное значение равно 10----27, если последовательность возрастает, или -1, если последовательность убывает. По умолчанию принимается NOMAXVALUE.

START WITH специфицирует первый генерируемый номер последовательности. Вы можете использовать эту опцию, чтобы начать возрастающую последовательность с значения, превышающего ее минимум, или начать убывающую последовательность с значения, которое меньше ее максимума. По умолчанию возрастающая последовательность начинается со своего

минимума, а убывающая - со своего максимума.

CYCLE указывает, что последовательность будет продолжать генерировать значения после достижения своего максимума или минимума. Возрастающая последовательность после достижения своего максимума генерирует свой минимум. Убывающая последовательность после достижения своего минимума генерирует свой максимум. NOCYCLE указывает, что последовательность не может продолжать генерировать значения после достижения своего максимума или минимума. По умолчанию принимается NOCYCLE.

CACHE указывает, сколько значений последовательности ORACLE распределяет заранее и поддерживает в памяти для быстрого доступа. Минимальное значение этого параметра равно 2. Для циклических последовательностей это значение должно быть меньше, чем количество значений в цикле. NOCACHE указывает, что значения последовательности не распределяются в памяти. Если опущены как параметр CACHE, так и опция NOCACHE, то ORACLE по умолчанию кэширует 20 номеров последовательности. Однако, если вы используете ORACLE с опцией параллельного сервера в параллельном режиме, и специфицировали опцию ORDER, то значения последовательности никогда не кэшируются, независимо от того, специфицировали ли вы параметр CACHE.

Например, для реализаций на уровне приложений типичным приемом является блокировка таблицы номеров последовательности, приращение номера и освобождение этой таблицы. При такой реализации лишь один номер может быть сгенерирован в каждый момент времени. Напротив, последовательности ORACLE позволяют одновременно генерировать множественные номера последовательностей, в то же время гарантируя уникальность каждого номера. Когда генерируется значение последовательности, выполняется ее приращение, независимо от подтверждения или отката транзакций. Если два пользователя одновременно осуществляют приращение одной и той же последовательности, то номера, которые видит каждый пользователь, могут иметь пропуски, потому что номера генерируются и для другого пользователя. Пользователь никогда не увидит номер последовательности, сгенерированный для другого пользователя. После того, как значение последовательности сгенерировано пользователем, этот пользователь может обращаться к этому значению независимо от того, осуществляет ли приращение последовательности другой пользователь.

Поскольку номера последовательностей генерируются независимо от таблиц, одну и ту же последовательность можно использовать для одной или для многих таблиц. Может оказаться, что отдельные номера последовательности будут пропущены, потому что эти номера были сгенерированы и использованы в транзакции, которая в конечном счете была подвергнута откату. Кроме того, каждый отдельный пользователь может не отдавать себе отчет в том, что другие пользователи черпают значения из той же самой последовательности.

Умолчания последовательности спроектированы таким образом, что, если вы не специфицируете ни одну из фраз в предложении CREATE SEQUENCE, то создадите возрастающую последовательность без верхнего предела, которая начинается с 1 и возрастает с шагом 1. Если вы специфицируете только INCREMENT BY -1, то создадите убывающую последовательность без нижнего предела, которая начинается с -1 и убывает с шагом 1.

При создании последовательности вы выбираете один из возможных способов ее

приращения:

- Значения последовательности наращиваются безгранично.

- Значения последовательности наращиваются до предопределенной границы, после чего последовательность останавливается.

- Значения последовательности наращиваются до предопределенной границы, после чего последовательность рестартует.

Чтобы создать последовательность, наращиваемую безгранично, опустите параметр MAXVALUE или специфицируйте опцию NOMAXVALUE (для возрастающей последовательности), либо опустите параметр MINVALUE или специфицируйте опцию NOMINVALUE (для убывающей последовательности).

Чтобы создать последовательность, которая останавливается на предопределенной границе, задайте значение для параметра MAXVALUE (для возрастающей последовательности) или MINVALUE (для убывающей последовательности). Кроме того, укажите опцию NOCYCLE. Любая попытка сгенерировать значение последовательности после того, как она достигла своего предела, приведет к ошибке. Чтобы создать последовательность, которая рестартует после предопределенной границы, задайте значение для параметра MAXVALUE или MINVALUE. Кроме того, укажите опцию CYCLE.

Значение параметра START WITH задает начальное значение, которое генерируется после создания последовательности. Заметьте, что это значение не обязательно совпадает с тем, с которого циклическая последовательность будет рестартовать после достижения своего максимума или минимума.

Индексы

Индексы - это необязательные структуры, ассоциированные с таблицами. Вы можете создавать индексы явно, чтобы ускорять выполнение предложений SQL по таблице. Как индекс в справочнике помогает быстрее найти нужную информацию, так и индекс ORACLE предоставляет более быстрый путь к данным таблицы. При правильном использовании индексы являются главным средством сокращения дискового ввода-вывода.

Наличие или отсутствие индекса не требует никаких изменений в написании предложений SQL. Индекс - это просто более быстрый путь к данным; он влияет лишь на скорость исполнения. При заданном значении данных, индексированных по этим значениям, индекс указывает непосредственно на адреса строк, содержащих это значение.

Индексы логически и физически независимы от данных в ассоциированной таблице. Индексы можно создавать и удалять в любой момент, не затрагивая ни базовых таблиц, ни других индексов. После удаления индекса все приложения по-прежнему будут работать; однако доступ к ранее индексированным данным может замедлиться. Индексы, как независимые структуры, требуют пространства для своего размещения.

Однажды созданный, индекс автоматически поддерживается и используется ORACLE. Изменения в данных, такие как добавление новых строк, обновление строк или удаление строк, автоматически отражаются во всех соответствующих индексах без каких-либо дополнительных действий пользователей.

Производительность извлечения индексированных данных остается почти постоянной, даже по мере вставки новых строк. Однако наличие многих индексов по таблице уменьшает производительность обновлений, удалений и вставок, поскольку индексы, ассоциированные с таблицей, также должны обновляться.

Уникальные и неуникальные индексы

Индексы могут быть ***уникальными*** или ***неуникальными***. Уникальные индексы гарантируют, что никакие две строки в таблице не содержат повторяющихся значений в столбце, по которому определен индекс. Неуникальные индексы не налагают такого ограничения на значения столбца.

Хотя это возможно, рекомендуется явно не определять уникальных индексов по таблицам базы данных ORACLE; уникальность - это строго логическое понятие, и она должна ассоциироваться с определением самой таблицы. Вместо этого следует определять ограничение целостности UNIQUE для соответствующих столбцов. ORACLE вводит в действие ограничения целостности UNIQUE путем автоматического определения уникального индекса по уникальному ключу.

Составные индексы

Составной индекс (называемый также сцепленным индексом) - это индекс, созданный по нескольким столбцам в таблице. Столбцы в составном индексе могут появляться в любом порядке и не обязаны быть соседними в таблице. Составные индексы могут ускорить извлечение данных предложениями SELECT, в которых фраза WHERE обращается ко всем или к ведущим столбцам в составном индексе. Поэтому выбор порядка столбцов в определении составного индекса требует некоторой мысли; обычно чаще используемые или наиболее селективные столбцы должны идти первыми.

Индексы и ключи

Хотя эти термины часто используются вместо друг друга, следует делать различие между "индексами" и "ключами".

Индексы - это структуры, действительно хранящиеся в базе данных, которые пользователи создают, изменяют и удаляют с помощью предложений SQL. Вы создаете индекс, чтобы обеспечить быстрый путь доступа к данным таблицы.

Ключи - это строго логическое понятие. Ключи соответствуют другому средству ORACLE,

называемому ограничениями целостности. Ограничения целостности вводят в действие организационные правила базы данных. Так как некоторые ограничения целостности реализуются индексами, термины "ключ" и "индекс" часто смешиваются; однако не следует путать их друг с другом.

Как хранятся индексы

Когда создается индекс, в табличном пространстве автоматически создается сегмент индекса для хранения данных этого индекса. Распределение пространства для сегмента индекса и использование этого зарезервированного пространства может контролироваться следующими способами:

- Распределением экстентов сегмента индекса можно управлять установкой параметров пространства для сегмента индекса.

- Свободную память в блоках данных, составляющих экстенты сегмента индекса, можно контролировать установкой параметра PCTFREE для сегмента индекса.

Табличное пространство для сегмента индекса - это либо умалчиваемое табличное пространство владельца, либо табличное пространство, явно специфицированное в предложении CREATE INDEX. Индекс не обязан располагаться в том же табличном пространстве, что и ассоциированная таблица. Более того, производительность запросов, использующих индекс, может быть улучшена за счет размещения индекса и таблицы в различных табличных пространствах, расположенных на разных дисковых устройствах, потому что данные индекса и данные таблицы могут извлекаться параллельно.

Формат блоков индекса

Память, доступная для данных индекса, определяется размером блока ORACLE за минусом накладных расходов блока, накладных расходов записи, ROWID и одного байта длины на каждое индексируемое значение. Число байт, требуемое для накладных расходов на блок индекса, зависит от операционной системы.

Когда создается индекс, индексируемые столбцы извлекаются и сортируются, и с каждым значением индекса для каждой строки записывается ROWID этой строки. Затем индекс загружается снизу вверх.

Например, рассмотрим предложение:

```
CREATE INDEX ware_addr ON warehouses(address);
```

ORACLE сортирует все строки warehouses по столбцу address и загружает индекс значениями address ROWID для каждой строки таблицы warehouses. Хотя ORACLE распознает ключевые слова ASC, DESC, COMPRESS и NOCOMPRESS в команде CREATE INDEX, они не оказывают влияния на данные индекса, которые хранятся с использованием сжатия сзади в узлах ветвей, но не в узлах листьев.

Внутренняя структура индексов

ORACLE использует для индексов B*-деревья, сбалансированные, чтобы выравнять время доступа к любой строке. Верхние блоки (БЛОКИ ВЕТВЕЙ) B*-дерева индекса содержат данные индекса, указывающие на блоки индекса более низкого уровня. Блоки индекса низшего уровня (БЛОКИ ЛИСТЬЕВ) содержат каждое индексированное значение данных и соответствующий ROWID, служащий для нахождения действительной строки; блоки листьев связаны в двусвязный список. Индексы по столбцам, содержащим символьные данные, базируются на двоичных значениях символов в наборе символов базы данных.

Для уникального индекса на каждое значение данных существует один ROWID. Для неуникального индекса ROWID включен в ключ в отсортированном порядке, так что неуникальные индексы отсортированы по ключу и по ROWID. Значения ключей, состоящие только из пустых значений, не индексируются, если это не индексы кластера. Две строки могут состоять из одних пустых значений, и это не нарушает уникальности индекса.

Структура B*-дерева имеет следующие преимущества:

- Все блоки листьев в дереве имеют одинаковую глубину, так что извлечение любой записи из любого места индекса требует приблизительно одинакового времени.

- B*-деревья индексов автоматически балансируются.

- Все блоки в B*-дереве в среднем заполнены на 3/4.

- B*-деревья обеспечивают блестящую производительность извлечений для широкого спектра запросов, включая поиск по точному совпадению и по диапазону.

- Вставки, обновления и удаления эффективны и поддерживают порядок ключей для быстрого извлечения.

- Производительность B*-деревьев хороша как для маленьких, так и для больших таблиц, и не падает с ростом таблиц.

Привилегии

Каждый пользователь в базе данных имеет набор привилегий. Это - то что пользователю разрешается делать. Эти привилегии могут изменяться со временем - новые добавляться, старые удаляться.

Объектные привилегии - позволяют пользователям выполнять конкретные действия на конкретном объекте (удалять строки в конкретной таблице, обращаться к таблице с запросом и т.п.).

Привилегии объекта связаны одновременно и с пользователями и с таблицами. То есть, привилегия дается определенному пользователю в указанной таблице, или базовой таблице или представлении. Привилегии, которые можно назначить пользователю:

- SELECT - Пользователь с этой привилегией может выполнять запросы в таблице.

- INSERT - Пользователь с этой привилегией может выполнять команду INSERT в таблице.

- UPDATE - Пользователь с этой привилегией может выполнять команду UPDATE на таблице. Вы можете ограничить эту привилегию для определенных столбцов таблицы.

- DELETE - Пользователь с этой привилегией может выполнять команду DELETE в таблице.

- REFERENCES - Пользователь с этой привилегией может определить внешний ключ, который использует один или более столбцов этой таблицы, как родительский ключ.

Механизм SQL назначает пользователям эти привилегии с помощью команды GRANT.

GRANT список объектных привелегий ON имя таблицы TO имена пользователей;

Пример. Пусть пользователь Diane имеет таблицу T1 и хочет позволить пользователю Adrian выполнить запрос к ней. Diane должна в этом случае ввести следующую команду:

GRANT SELECT ON T1 TO Adrian;

Теперь Adrian может выполнить запросы к таблице T1. Без других привилегий, он может только выбрать значения; но не может выполнить любое действие, которые бы воздействовало на значения в таблице T1 (включая использование таблицы T1 в качестве родительской таблицы внешнего ключа).

SQL поддерживает два аргумента для команды GRANT, которые имеют специальное значение: **ALL PRIVILEGES** (все привилегии) или просто **ALL** и PUBLIC (общие).

ALL используется вместо имен привилегий в команде GRANT, чтобы отдать все привилегии в таблице. Например, Diane может дать Stephen весь набор привилегий в таблице T1 с помощью такой команды:

GRANT ALL ON T1 TO Stephen;

PUBLIC: когда вы предоставляете привилегии для публикации, все пользователи автоматически их получают. Наиболее часто, это применяется для привилегии SELECT в определенных базовых таблицах или представлениях, которые вы хотите сделать доступными для любого пользователя. Чтобы позволить любому пользователю видеть таблицу T2, вы, например, можете ввести следующее:

GRANT SELECT ON T2 TO PUBLIC;

Отмена привилегий

Потребность удалять привилегии сводится к команде **REVOKE**, фактически стандартному средству с достаточно понятной формой записи. Синтаксис команды REVOKE - похож на GRANT, но имеет обратный смысл.

REVOKE перечень объектных привелегий ON имена объектов FROM имена пользователей;

Чтобы удалить привилегию INSERT для Adrian в таблице T2, вы можете ввести

REVOKE INSERT ON T2 FROM Adrian;

Системные привилегии

Привилегии, которые не определяются в терминах специальных объектов данных, называются привилегиями системы, или правами базы данных.

Системные привелегии - позволяют пользователям выполнять конкретное действие на уровне системы или конкретное действие над конкретным типом объектов (создавать пользователя, таблицу, представление и т.д.).

При общем подходе имеется три базовых привилегии системы:

- CONNECT (Подключить);

- RESOURCE (Ресурс);

- DBA (Аминистратор Базы Данных).

Команда GRANT, в измененной форме, является пригодной для использования с привилегиями объекта, как и с системными привилегиями.

GRANT список системных привелегий TO имена пользователей;

Для начала передача прав может быть сделана с помощью DBA. Например, DBA может передать привилегию для создания таблицы пользователю Rodriguez следующим образом:

GRANT RESOURCE TO Rodriguez;

В следующей команде системный администратор может передать привилегию создавать таблицы CREATE TABLE пользователю Thelonius:

GRANT CREATE TABLE TO Thelonius;

а потом отменить ее:

REVOKE CREATE TABLE FROM Thelonius;

Утилита SQL*Plus

Программа SQL*PLUS может быть использована совместно с языком БД SQL и его процедурно языковым расширением, PL/SQL. Язык БД SQL позволяет запоминать и выбирать данные из СУРБД Oracle, PL/SQL позволяет соединять команды SQL в логические процедуры.

SQL*PLUS позволяет манипулировать командами SQL и блоками PL/SQL, и также выполнять много дополнительных задач. С помощью SQL*PLUS возможно:

- вводить, редактировать, запоминать, выбирать и выполнять команды SQL и блоки PL/SQL;

- форматировать, выполнять вычисления, запоминать и печатать результаты запросов в форме отчетов;

- печатать описания колонок для каждой таблицы;

- иметь доступ и возможность копирования данных между БД SQL;

- посылать сообщения и принимать ответ от конечных пользователей.

Основные положения SQL*PLUS

Приведем основные понятия SQL*PLUS:

command - команда, которую вы передаете SQL*PLUS;

block - группа команд SQL или PL/SQL связанных вместе процедурной логикой;

query - команда SQL, которая выбирает информацию из одной или более таблиц;

query results - данные выбранные с помощью запроса;

report - результаты запроса отформатированные вами командами SQL*PLUS.

Запуск SQL*PLUS

Для запуска SQL*PLUS в режиме командной строки (если известно, где находится файл SQL*PLUS или указан каталог для поиска необходимого файла) необходимо:

- ввести команду SQL*PLUS и нажать [Return]. Это команда ОС стартующая SQL*PLUS;

- ввести имя-пользователя и нажмите [Return]. SQL*PLUS высветит подсказку "Enter password:";

- ввести пароль и нажать [Return] снова. Для обеспечения защиты, пароль не отображается на экране.

Процесс ввода вашего имени и пароля, называется логированием. SQL*PLUS покажет версию ORACLE к которой вы подключились и версии доступных пакетов (например PL/SQL)

При запуске программы с использованием интерфейса Windows (кнопка «Пуск» и выбора утилиты SQL*PLUS из каталога Ora) в диалоговом режиме необходимо зарегистрироваться, введя имя и пароль пользователя, а также базу данных.

Имя пользователя - XXXXXX Scott

Пароль пользователя - XXXXXX tiger

База данных - XXXXXX NTBASE

Обработка команд в SQL*PLUS

В SQL*PLUS можно использовать 3 типа команд:

- команды SQL, для работы с информацией в БД;

- блоки PL/SQL, также для работы с информацией в БД;

- команды SQL*PLUS, для форматирования результатов запроса, установки параметров, редактирования и запоминания команд SQL и блоков PL/SQL.

Способ, которым возможно продолжение команды на дополнительных строках, окончания команды, или выполнение команды, зависит от типа команды которую пользователь хочет ввести и выполнить. После ввода команду и указания SQL*PLUS ее выполнить, SQL*PLUS обрабатывает команду и выводит командную подсказку снова, указывая , что можно вводить следующую команду.

Синтаксис команд SQL*PLUS

SQL*PLUS имеет синтаксические правила, которые управляют процессом сборки слов в команды. Необходимо следовать им, чтобы вводимые команды правильно выполнялись.

Возможно деление команд SQL на отдельные строки в любых местах, но нельзя разрывать отдельные слова между строками. Таким образом, можно вводить запрос на одной строке (или на нескольких).

Точка с запятой сообщает SQL*PLUS, что требуется выполнить команду. Если пользователь по ошибке нажал [Return] до ввода точки с запятой, SQL*PLUS выведет новый номер строки для продолжения команды. Чтобы выполнить команду, надо ввести точку с запятой и нажать [Return] снова.

Наклонная черта (/) в строке также сообщает SQL*PLUS, что пользователь хочет выполнить команду. Требуется [Return] в конце последней строки команды. SQL*PLUS выведет новый номер строки, после чего необходимо ввести наклонную черту и снова нажать [Return]. SQL*PLUS выполнит команду.

Пустая строка сообщает SQL*PLUS, что ввод команды закончен, но пока не требуется ее выполнять. После нажатия [Return] в конце последней строки команды SQL*PLUS подскажет пользователю новый номер строки.

Если снова нажать [Return], то SQL*PLUS выведет уже свою подсказку. SQL*PLUS не выполнит команду, но запомнит ее в памяти называемой буфер SQL.

Область, в которой SQL*PLUS хранит последнюю введенную команду, называется буфером SQL. Команда остается в буфере до тех пор пока пользователь не введет новую команду SQL. Таким образом, если требуется изменить или выполнить команду снова, можно это сделать без повторного ввода команды.

SQL*PLUS не запоминает точку с запятой или наклонную черту в буфере, которые были введены для выполнения команды.

Буфер SQL является буфером по умолчанию. Можно назначить новый буфер, но SQL*PLUS

не требует использования множества буферов.

Можно выполнить (или выполнить заново) текущую команду SQL, посредством ввода команды RUN или командной подсказки команды "/" после командной подсказки.

Команда RUN перед исполнением печатает содержимое буфера; команда "/" просто выполняет команды SQL.

Также можно использовать программы PL/SQL (называются блоками) для манипулирования данными в БД. Блоки PL/SQL начинаются с DECLARE, BEGIN, или имени блока. SQL*PLUS обращается с блока PL/SQL таким же способом как и с командами SQL, кроме точки с запятой или пустой строки, которые не оканчивают и не выполняют блок. Блок PL/SQL завершается точкой (.) на новой строке.

SQL*PLUS хранит блоки, введенные пользователем, в буфере SQL. Выполнить текущий блок можно с помощью команды RUN или "/". SQL*PLUS посылает весь блок PL/SQL к ORACLE RDBMS для обработки. При выполнении блок, команды SQL внутри блока могут вести себя иначе, чем вне блока.

Возможно продолжение длинных команд SQL*PLUS посредством ввода дефиса (соединительной черты) в конце строки и нажатия [Return]. По желанию пользователя, можно ввести пробел до ввода дефиса. SQL*PLUS выведет правую угловую скобку (>) как подсказку для каждой дополнительной строки.

Необязательно заканчивать команду SQL*PLUS с помощью точки с запятой. Когда окончен ввод команды, можно сразу нажать [Return]. Однако, по пожеланию пользователя, можно ввести точку с запятой в конце команды SQL*PLUS.

Переменные, влияющие на выполнение команд

Команда SQL*PLUS SET управляет большинством называемых системные переменные - установка которых воздействует на способ выполнения SQL*PLUS-ом ваших команд.

Системные переменные управляют множеством параметров внутри SQL*PLUS, включая ширину колонок по умолчанию для вывода на экран, или показ количества обработанных записей, или длину страницы. Системные переменные также называют переменными команды SET.

Автоматическое сохранение изменений в БД

Благодаря командам SQL DML UPDATE, INSERT, DELETE- которые могут использоваться в блоках PL/SQL- модно проводить изменения в данных БД. Тем не менее, SQL не изменяет информацию в БД пока не будет выполнена команда SQL COMMIT или команды SQL DCL или DDL, например CREATE TABLE.

Можно не откладывать проведение изменений в БД, пока не будет выполнена одна из выше перечисленных команд. Возможность автоматического проведения изменений SQL*PLUS (autocommit) может заставить его проводить изменения в БД после каждой команды SQL- включая INSERT, UPDATE, DELETE- и после каждого выполненного блока PL/SQL.

Возможностью автоматического проведения изменений SQL*PLUS можно управлять с помощью переменной AUTOCOMMIT команды SET. Она имеет следующий формат:

SET AUTCOMMIT ON включить

SET AUTCOMMIT OFF выключить (по умолчанию)

Когда включен режим автоматического проведения изменений, невозможно совершение отката в БД.

Установка окружения для SQL*PLUS

Для самостоятельной настройки среды SQL*PLUS (например показать текущее время как часть подсказки SQL*PLUS) и затем многократно использовать установки установки данного сеанса. Это можно сделать с помощью файла операционной системы LOGIN.SQL (его называют файлом параметров пользователя [profile]).

Поставляемый ORACLE командный файл DEMOBLD создает файл LOGIN в пользовательской директории. Сначала он будет пустым, кроме предложений DEFINE.

Можно добавить любые команды SQL, блоки PL/SQL, или команды SQL*PLUS в этот файл; при запуске SQL*PLUS, он автоматически ищет пользовательский файл LOGIN (сначала в текущей директории, затем в директориях задействованных в пути [path]) и выполняет его команды (также можно иметь файл параметров "местоположения" [Site]).

Если запустили на выполнение DEMOBLD, а LOGIN файл уже существует в текущей директории, DEMOBLD спросит пользователя, сохранять или нет новый LOGIN с новым именем LOGIN.OLD. При необходимости сохранить свой LOGIN файл, дайте ответ 'yes', а затем переименуйте LOGIN.OLD в LOGIN.SQL. Но DEMODROP не удаляет файл LOGIN; он удаляет только таблицы-примеры.

Форматирование вывода запроса

С помощью команды SQL*PLUS COLUMN, возможно изменение заголовка и переформатирование данных колонки в результатах запросов.

При выводе результаты запросов, по умолчанию SQL*PLUS использует имена колонок и выражений в качестве заголовков колонок. Имена колонок могут быть короткими и не понятными,

а выражения могут быть тяжелы для восприятия. Определение более полезных заголовков колонок достигается с использованием параметра HEADING команды COLUMN в следующем формате:

COLUMN имя_колонки HEADING заголовок_колонки

Для замены заголовка колонки на два или более слова, необходимо заключить новый заголовок в одиночные или двойные кавычки при вводе команды COLUMN. Для вывода заголовка более чем на одной строке допускается использование вертикальной черты (|) там, где необходимо начать новую строку (можно в качестве разделителя использовать другой символ, установив переменную HEADSEP команды SET).

Обычно SQL*PLUS показывает в числах столько цифр, сколько требуется для указанной точности, стандартная точность (ширина) определяется значением переменной NUMWIDTH команды SET (обычно 10). Возможно задание разных форматов для колонок типа NUMBER, с использованием модели формата в команде COLUMN. С помощью модели формата задают вид колонок при печати, для представления цифр используется 9.

Команда COLUMN указывает колонку, которую вы желательно форматировать, и используемую модель, как показано ниже:

COLUMN имя_колонки FORMAT модель

Использовать модели форматов можно, чтобы добавить запятые, знак доллара, угловые скобки, ведущие нули к числа форматируемой колонки. Также можно подвергать числа округлению, выводить знак числа справа (обычно он выводится слева), выводить числа в экспоненциальной форме. Чтобы использовать одну форматную модель для одной колонки, требуется объединить модель в одной команде COLUMN.

При указании ширины меньшей, чем ширина заголовка, SQL*PLUS обрезает заголовок. Заметим, что заголовок для NUMBER никогда не урезается. При необходимости назначить аналогичные атрибуты для нескольких колонок, необходимо уменьшить длину вводимых команд с помощью параметра LIKE команды COLUMN. Параметр LIKE приказывает SQL*PLUS скопировать атрибуты вывода из уже определенной колонки в новую колонку, кроме тех параметров, которые задаются в данной команде.

Возможно подавление и восстановление атрибутов вывода, которые были назначены определенной колонке. Для подавления атрибутов колонки вывода колонки, вводится команда COLUMN в следующем формате:

COLUMN имя_колонки OFF

Фраза OFF сообщает SQL*PLUS, что необходимо использовать атрибуты вывода для данной

колонки определенные по умолчанию, но сами атрибуты вывода удалять не надо. Для восстановления атрибутов необходимо ввести команду COLUMN с фразой ON:

COLUMN имя_колонки ON

Определение заголовков и размеров страниц

Слово страница означает полный экран с информацией на вашем дисплее, или страницу печатаемого отчета. Возможно помещение заголовков в начале и конце каждой страницы, установление количество строк на странице, и определение ширину каждой строки.

Для вывода вверху каждой страницы отчета возможно установление заголовка с использованием команды TTITLE. Также можно устанавливать заголовок, печатаемый внизу каждой страницы (команда BTITLE).

Команда TTITLE или BTITLE состоит из имени команды с последующими параметрами, задающими позицию или формат и значение CHAR, которое пользователь может поместить в заданную позицию или в заданном формате. Возможно включение нескольких наборов фраз и значений CHAR:

TTITLE фраза_позиция значение_char

фраза_позиция значение_char ...

или

BTITLE фраза_позиция значение_char

фраза_позиция значение_char ...

Часто используемые фразы команд TTITLE и BTITLE приведены в таблице 3.

Таблица 3 - Часто используемые фразы TTITLE и Фраза BTITLE

Фразы	Пример	Описание
COL n	COL 72	Помещает следующее CHAR-значение в указанную колонку строки
SKIP n	SKIP 2	Пропуск n строк. Если n > 1, перед следующим CHAR-значением появятся n-1 пустых строк.
LEFT	LEFT	Выравнивает влево следующее CHAR-значение
CENTER	CENTER	Центрирует следующее CHAR-значение
RIGHT	RIGHT	Выравнивает вправо следующее CHAR-значение

Обычно, страница отчета содержит верхний заголовок, заголовки колонок, результаты запроса и нижний заголовок. SQL*PLUS выдает длинные отчеты на нескольких последовательных страницах, каждая со своим заголовком и заголовками колонок. Количество данных, выведенное SQL*PLUS-м на каждой странице, зависит от текущих размеров страницы.

По умолчанию размеры страницы, используемые SQL*PLUS, показаны ниже:

- количество строк перед верхним заголовком - 1;

- количество строк на странице, от верхнего заголовка до нижнего заголовка страницы - 14;

- количество символов в строке - 80.

Можно изменять данные установки настраивая на длину экрана вашей ЭВМ, или для печати с помощью системных переменных NEWPAGE и PAGESIZE. Например, это возможно применить, когда требуется распечатать отчет, так как длина листа равна 66 строкам, а не 15 (общее количество строк на странице равно сумме NEWPAGE и PAGESIZE).

Чтобы установить количество строк между началом каждой страницы и верхним заголовком, используется переменная NEWPAGE команды SET:

SET NEWPAGE количество_строк

При установке NEWPAGE = 0, SQL*PLUS пропустит ноль строк и напечатает символ подачи формы (formfeed) в начале каждой страницы. На большинстве типов экранов компьютеров, символ подачи формы чистит экран и перемещает курсор в начало первой строки. При печати отчета, символ подачи формы загружает новый лист бумаги, даже если допустимая длина страницы меньше, чем реальная длина листа бумаги.

Для установки количества строк на странице от верхнего заголовка, используется переменная PAGESIZE команды SET:

SET PAGESIZE количество_строк

Возможно уменьшение размера строки для центрирования заголовков отчета. Также допустимо увеличение размера строки для печати на широких листах. Изменение ширины строки, производится с использованием переменную LINESIZE команды SET:

SET LINESIZE количество_символов

С помощью команды SQL*PLUS SPOOL, возможно сохранение результатов запроса в файле или распечатка их на принтере.

Чтобы сохранить результаты запроса в файле- и напечатать их на экране требуется ввести команду SPOOL в следующей форме:

SPOOL имя_файла

SQL*PLUS сохранит всю информацию, которая будет выводиться после команды SPOOL, в указанный файл. Если не было указано расширения в имени файла, SPOOL добавит к имени файла расширение по умолчанию, чтобы указать, что это файл вывода. Это расширение зависит от ОС; обычно оно имеет имя LST или LIS. SQL*PLUS продолжает записывать информацию в файл, пока пользователь не выключит запись в файл, используя SPOOL в следующей форме:

SPOOL OFF

Для распечатки результатов запроса, их необходимо записать в файл. Затем, вместо использования SPOOL OFF, вводится команда в следующей форме:

SPOOL OUT

SQL*PLUS закончит буферизацию и скопирует содержимое буферного файла на принтер. SPOOL OUT напечатает, но не уничтожит буферный файл после печати.

Доступ к БД

Для доступа к данным существующей БД, необходимо подключиться к БД. При запуске SQL*PLUS, обычно происходит подключение к БД ORACLE по умолчанию, с именем и паролем, которые указывается во время запуска. После логирования, можно подключиться к БД с другим именем пользователя командой SQL*PLUS CONNECT. Эти имя и пароль должны быть правильными для данной БД.

ORACLE может быть установлен на нескольких компьютерах. Такие компьютеры часто объединяют в сеть, которая позволяет программам на разных ЭВМ быстро и эффективно обмениваться данными. Сетевые компьютеры могут находиться рядом или на больших расстояниях, и объединяться телекоммуникационными связями.

БД на других ЭВМ или другие БД на рабочей ЭВМ (кроме БД по умолчанию) называются удаленными БД (remote). Можно обращаться к удаленной БД, если требуемая БД имеет SQL*NET и обе БД имеют совметимые сетевые драйверы.

К удаленной БД можно подключаться двумя способами:

- из SQL*PLUS, используя команду CONNECT;

- во время запуска SQL*PLUS, используя команду SQL*PLUS;

Чтобы подключиться к удаленной БД используя CONNECT, требуется включить SQL*NET - спецификацию БД в команду CONNECT в следующей форме (вводимые имя и пароль должны быть правильными для удаленной БД):

- CONNECT SCOTT@спецификация_БД

- CONNECT SCOTT/TIGER@спецификация_БД

SQL*PLUS, если необходимо, попросит ввести имя и пароль, и подключится к указанной БД. Данная БД становится базой по умолчанию, пока не произойдет подключение к другой БД, либо не будет выполнено отключение (DISCONNECT), или пользователь не покинет SQL*PLUS.

При подключении к удаленной БД данным способом, можно использовать полный набор команд SQL*PLUS, SQL и блоков PL/SQL. Строка, вводимая как спецификация БД, зависит от протокола SQL*NET на компьютере.

Чтобы подключится к удаленной БД во время запуска SQL*PLUS, во время запуска требуется включить SQL*NET-спецификацию БД в команду SQLPLUS в одном из следующих форматов:

- SQLPLUS SCOTT@спецификация_БД

- SQLPLUS SCOTT/TIGER@спецификация_БД

Требуется указать допустимые имя и пароль пользователя удаленной БД и подставить соответсвующую спецификацию удаленной БД. При необходимости, SQL*PLUS попросит ввести имя и пароль, стартует SQL*PLUS, и подключит пользователя к указанной БД. Данная БД становится базой по умолчанию для данного сеанса работы

Работа со словарем данных

Словарь данных - не только центральное хранилище в каждой базе данных ORACLE. Это также важный инструмент для всех пользователей, от конечных пользователей до разработчиков приложений и администраторов базы данных. Даже начинающие пользователи могут извлекать выгоду из понимания и использования словаря данных.

Однако ни один пользователь не должен никогда изменять (обновлять, удалять или вставлять) никакие строки или объекты в словаре данных; такие действия могут нарушить целостность данных.

Словарь данных является одной из важнейших частей базы данных ORACLE.

СЛОВАРЬ ДАННЫХ - это набор таблиц, используемых как ТОЛЬКО-ЧИТАЕМЫЙ справочник, который предоставляет информацию об ассоциированной с ним базе данных.

Например, словарь данных может предоставлять следующую информацию:

- имена пользователей ORACLE;

- привилегии и роли, которые были предоставлены каждому пользователю;

- имена объектов схем (таблиц, обзоров, снимков, индексов, кластеров, синонимов, последовательностей, процедур, функций, пакетов, триггеров и т.д.);

- информацию об ограничениях целостности;

- умалчиваемые значения для столбцов;

- сколько пространства было распределено и в настоящее время используется объектами в базе данных;

- информацию аудитинга, например, кто обращался к различным объектам и обновлял их;

- другую общую информацию о базе данных.

Словарь данных структурирован через таблицы и обзоры, как и любые другие данные в базе данных. Для обращения к словарю данных вы используете SQL. Так как словарь данных можно только читать, пользователи могут выдавать лишь запросы (предложения SELECT) по таблицам и обзорам словаря данных.

Структура словаря данных

В состав словаря данных базы данных входят:

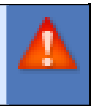
а) базовые таблицы: основу словаря данных составляет совокупность таблицы базовых таблиц, хранящих информацию о базе данных. Эти таблицы читаются и пишутся ТОЛЬКО самим ORACLE; они редко используются непосредственно пользователем ORACLE любого типа, потому что они нормализованы, и большая часть данных в них закодирована.

б) доступные пользователю обзоры: словарь данных содержит доступные пользователю обзоры, которые суммируют и отображают в удобном обзоры формате информацию из базовых таблиц словаря. Эти обзоры декодируют информацию базовых таблиц, представляя ее в полезном виде, таком как имена пользователей или таблиц, и используют соединения и фразы WHERE, чтобы упростить информацию. Большинство пользователей имеют доступ к этим обзорам вместо базовых таблиц словаря.

SYS, владелец словаря данных

Все базовые таблицы и обзоры словаря данных принадлежат пользователю ORACLE с учетным именем SYS. Поэтому ни один пользователь ORACLE не должен НИКОГДА изменять никаких объектов, содержащихся в схеме SYS, а администратор безопасности должен строго контролировать использование этого центрального учетного имени.

Замечание: Изменение данных или манипулирование ими в базовых таблицах словаря данных может нанести необратимый ущерб работоспособности базы данных.



Как используется словарь данных

Словарь данных базы данных ORACLE имеет два основных применения:

- ORACLE обращается к словарю данных каждый раз, когда выдается предложение DDL.

- Каждый пользователь ORACLE может использовать словарь данных как только-читаемый справочник по базе данных.

Как ORACLE и другие продукты Oracle используют словарь данных

Данные в базовых таблицах словаря данных необходимы для функционирования ORACLE. Поэтому только ORACLE должен записывать или изменять информацию словаря данных. Во время операций по базе данных ORACLE читает словарь данных, чтобы удостовериться, что нужные объекты существуют, и что пользователи имеют к ним должный доступ. ORACLE также непрерывно обновляет словарь данных, чтобы отражать происходящие изменения в структурах базы данных, аудитинге, грантах и данных.

Например, если пользователь KATHY создает таблицу с именем PARTS, в словарь данных добавляются новые строки, чтобы описать новую таблицу, ее столбцы, сегмент, экстенты, а также привилегии, которые KATHY имеет по этой таблице. Эта новая информация становится видимой при очередном опросе обзоров словаря.

Кэширование словаря данных для быстрого доступа

Поскольку ORACLE постоянно обращается к словарю данных для проверки прав доступа пользователей и состояния объектов, большая часть информации словаря данных кэшируется в SGA. Вся информация поддерживается в памяти с помощью алгоритма LRU (т.е. вытесняется наиболее давно использовавшаяся информация). Обычно в кэше поддерживается информация, требуемая для разбора предложений. Столбцы COMMENTS, описывающие таблицы и столбцы, не кэшируются, если они не используются часто.

Словарь данных и другие программы

Другие продукты Oracle могут дополнительно создавать в словаре данных свои собственные таблицы и обзоры, а также обращаться к существующим обзора.

Разработчики приложений, которые пишут программы, обращающиеся к словарю данных, должны использовать общие синонимы, а не сами базовые таблицы: эти синонимы менее подвержены изменениям между версиями ORACLE.

Добавление новых элементов данных словаря

Вы можете добавлять в словарь данных новые таблицы или обзоры. Если вы добавляете новые объекты в словарь данных, их владельцем должен быть SYSTEM или какой-либо третий пользователь ORACLE. Никогда не создавайте новых объектов под учетным именем SYS, если только вы не выполняете скрипт, специально предоставляемый фирмой Oracle для создания новых объектов словаря данных.

Удаление элементов словаря данных

Так как все изменения в словаре данных осуществляются ORACLE в ответ на предложения DDL, НИКАКИЕ ДАННЫЕ В ТАБЛИЦАХ СЛОВАРЯ ДАННЫХ НЕ ДОЛЖНЫ УДАЛЯТЬСЯ ИЛИ ИЗМЕНЯТЬСЯ непосредственно пользователем. Единственное исключение из этого правила представляет таблица SYS.AUD$. Когда включен режим аудитинга, эта таблица может неограниченно расти. Хотя вы не должны удалять эту таблицу, администратор безопасности может удалять из нее данные, потому что строки этой таблицы служат лишь для информации и не являются необходимыми для работы ORACLE.

Общие синонимы для обзоров словаря данных

Общие синонимы создаются для многих обзоров словаря данных, так что они легко доступны всем пользователям. Администратор безопасности может создавать дополнительнные общие синонимы для общесистемных объектов. Однако другие пользователи должны избегать давать своим собственным объектам имена, совпадающие с общими синонимами.

Как пользователи ORACLE могут использовать словарь данных

Обзоры словаря данных выступают как справочники для всех пользователей базы данных. Доступ к этим обзорам осуществляется через SQL. Некоторые обзоры доступны всем пользователям, тогда как некоторые другие предназначены лишь для администраторов.

Словарь данных всегда доступен при открытой базе данных. Он размещается в табличном пространстве SYSTEM, которое всегда находится в состоянии онлайн, когда база данных открыта.

Словарь данных состоит из нескольких наборов обзоров. Во многих случаях такой набор состоит из трех обзоров, содержащих аналогичную информацию и отличающихся друг от друга своими префиксами:

- Префикс USER - обзор пользователя (что есть в схеме пользователя);

- ALL - расширенный обзор пользователя (к чему есть доступ);

- DBA - обзор администратора (к чему все пользователи имеют доступ).

Столбцы в каждом обзоре в наборе идентичны, со следующими исключениями:

- В обзорах с префиксом USER обычно нет столбца с именем OWNER (владелец); в обзорах USER под владельцем подразумевается пользователь, выдавший запрос.

- Некоторые обзоры DBA имеют дополнительные столбцы, которые содержат информацию, полезную для АБД.

Обзоры с префиксом USER

Обзоры с префиксом USER представляют наибольший интерес для типичных пользователей базы данных.

Эти обзоры:

- отражают собственное окружение пользователя в базе данных, включая информацию об объектах, созданных этим пользователем, грантах, предоставленных им, и т.д.;

- выдают лишь строки, имеющие отношение к пользователю;

- имеют столбцы, идентичные с другими обзорами, с тем исключением, что столбец OWNER подразумевается (текущий пользователь);

- возвращают подмножество информации, предоставляемой обзорами ALL;

- могут иметь сокращенные общие синонимы для удобства.

Например, следующий запрос возвращает все объекты, содержащиеся в вашей схеме:

```
SELECT object_name, object_type FROM user_objects;
```

Обзоры с префиксом ALL

Обзоры с префиксом ALL отражают общее представление о базе данных со стороны пользователя. Эти обзоры возвращают информацию об объектах, к которым пользователь имеет доступ через общие или явные гранты привилегий и ролей, помимо тех объектов, которыми владеет этот пользователь. Например, следующий запрос возвращает информацию обо всех объектах, к которым вы имеете доступ:

```
SELECT owner, object_name, object_type FROM all_objects;
```

Обзоры с префиксом DBA

Обзоры с префиксом DBA показывают общее представление о базе данных, так что они

предназначены только для администраторов базы данных. Точнее, опрашивать обзоры словаря с префиксом DBA может любой пользователь, имеющий системную привилегию SELECT ANY TABLE.

Для таких обзоров не создаются синонимы, так как обзоры DBA должны использовать лишь администраторы. Поэтому, чтобы выдать запрос по обзору DBA, администратор должен префиксовать имя обзора именем владельца этого обзора, т.е. SYS, например:

SELECT owner, object_name, object_type FROM sys.dba_objects;

Администраторы могут выполнить скрипт DBA_SYNONYMS.SQL, чтобы создать личные синонимы для обзоров DBA под своим учетным именем, если они имеют системную привилегию SELECT ANY TABLE. Выполнение этого скрипта создает синонимы только для текущего пользователя.

DUAL

Таблица с именем DUAL - это крошечная таблица, к которой обращаются как ORACLE, так и пользовательские программы, чтобы гарантировать известный результат. Эта таблица имеет одну строку и один столбец.

Работа со словарем данных

Во время работы с SQL*PLUS может возникнуть ситуация, когда необходимо распечатать описание колонок таблицы, или начать и прервать вывод на экран. Также понадобиться объяснение сообщений об ошибках, полученных при неправильном вводе команд или когда возникают проблемы с ORACLE или SQL*PLUS.

Чтобы посмотреть описание каждой колонки в таблице, возможно использование команды SQL*PLUS DESCRIBE (DESC).

Подробную информацию об ошибках можно получить, используя команду SHOW ERROR, результатом выполнения которой будет расшифровка всех ошибок последней команды (или блока команд) SQL с указанием кода ошибки.

После ознакомления с данным материалом рекомендуется выполнить лабораторную работу №1

Введение в аудитинг

АУДИТИНГ - это отслеживание и регистрация выбранных действий пользователей в базе данных.

Аудитинг обычно используется:

- для расследования подозрительной деятельности. Например, если данные удаляются из таблиц без санкции, то администратор безопасности может принять решение следить за всеми подключениями к базе данных и за всеми успешными и безуспешными попытками удаления строк во всех таблицах базы данных.

- для сбора информации о конкретных действиях в базе данных. Например, администратор базы данных может собирать статистику о том, какие таблицы обновляются, сколько выполняется логических операций ввода-вывода, или сколько пользователей одновременно подключено в пиковые моменты времени.

Типы аудитинга

ORACLE поддерживает три основных типа аудитинга:

- Выборочный аудит предложений SQL по типам предложений предложений, безотносительно к конкретной структуре или объекту, например, AUDIT TABLE. Обычно такой аудитинг проводится в широком спектре опций, отслеживая несколько типов связанных действий для каждой опции. Аудитинг предложений можно проводить по выбранным пользователям или по всем пользователям в базе данных.

- Выборочный аудит использования мощных системных привилегий привилегий на определенные действия, например, AUDIT CREATE TABLE. Этот режим более сфокусирован, чем аудит предложений, отслеживая лишь использование определенной привилегии. Аудитинг привилегий можно проводить по выбранным пользователям или по всем пользователям в базе данных.

- Выборочный аудит конкретных предложений на объектов конкретном объекте схемы, например, AUDIT SELECT ON EMP. Этот режим наиболее сфокусирован, отслеживая лишь конкретное предложение на определенном объекте. Аудитинг объектов всегда применяется ко всем пользователям базы данных.

Опции аудитинга устанавливаются для того, чтобы определить тип собираемой информации.

ORACLE позволяет как фокусировать опции аудита, так и распространять их:

- отслеживать успешные выполнения предложений, безуспешные попытки, или и те, и другие;

- отслеживать выполнение предложения один раз на сессию пользователя или при каждом выполнении предложения;

- отслеживать деятельность всех или конкретного пользователя.

Аудиторские записи и аудиторский журнал

Аудиторские записи включают такую информацию, как отслеживаемая операция, пользователь, выполнивший эту операцию, и дата/время операции. Аудиторские записи могут сохраняться либо в таблице словаря данных, называемой аудиторским журналом (audit trail), либо в аудиторском журнале операционной системы.

Аудиторский журнал базы данных - это единственная таблица с именем AUD$ в схеме SYS в словаре данных каждой базы данных ORACLE. Предоставляются несколько предопределенных обзоров, чтобы помочь вам использовать эту информацию.

Механизмы аудитинга

ORACLE позволяет включать и отключать регистрацию аудиторской информации. Эта возможность позволяет любому полномочному пользователю в любой момент устанавливать опции аудита, но сохраняет контроль за записью аудиторской информации за администратором безопасности.

Если аудитинг включен, то аудиторская запись генерируется на фазе исполнения при выполнении предложения. Предложения SQL внутри программных единиц PL/SQL отслеживаются индивидуально, в соответствии с режимами аудитинга, при выполнении программной единицы. Генерация аудиторской записи и ее включение в аудиторский журнал осуществляются независимо от транзакции пользователя; поэтому, если даже транзакция пользователя откатывается, запись в аудиторском журнале остается подтвержденной.

Аудиторские записи никогда не генерируются сессиями, установленными от имени SYS, или соединениями в режиме INTERNAL. Подключения в этих режимах обходят некоторые внутренние средства ORACLE с целью выполнения специфических административных операций (например, запуска или закрытия базы данных, восстановления и т.д.).

Опции аудитинга предложений и аудитинга привилегий, которые действовали на момент соединения пользователя с базой данных, т.е. на момент начала сессии, остаются действительными на все время этой сессии; сессия не видит установок или изменений опций аудитинга, осуществленных в течение ее выполнения. Пользователь базы данных попадет под действие изменившихся режимов аудита лишь после того, как закончит текущую сессию и начнет новую. Напротив, что касается аудитинга объектов, сессия чувствительна ко всем установкам и изменениям опций аудитинга, осуществляемым в течение ее выполнения.

Аудитинг в распределенной базе данных

Аудитинг автономен для узла; инстанция отслеживает лишь предложения, выдаваемые непосредственно присоединенными пользователями. Локальный ORACLE не может отслеживать действий, имеющих место на удаленной базе данных. Так как удаленные соединения устанавливаются через имя пользователя и связь баз данных, предложения, выданные через соединение по связи баз данных, отслеживаются удаленным ORACLE. Обратитесь к главе 21 для дополнительной информации о распределенных базах данных и связях баз данных.

Аудитинг предложений

Аудитинг предложений - это выборочное отслеживание связанных групп предложений, которые попадают в две категории:

- Предложения DDL, касающиеся конкретного ТИПА структуры или объекта базы данных, но не конкретного экземпляра структуры или объекта (например, AUDIT TABLE отслеживает все предложения CREATE TABLE и DROP TABLE);

- Предложения DML, касающиеся конкретного ТИПА структуры или объекта базы данных, но не конкретного экземпляра структуры или объекта (например, AUDIT SELECT TABLE отслеживает все предложения SELECT ... FROM TABLE, VIEW или SNAPSHOT, независимо от имени этой таблицы, обзора или снимка).

Аудитинг предложений может быть широким, отслеживая действия всех пользователей базы данных, или узким, отслеживая действия выбранных пользователей.

Аудитинг привилегий

Аудитинг привилегий - это выборочное отслеживание предложений, которые выполняются носителями системных привилегий. Например, аудитинг по системной привилегии SELECT ANY TABLE отслеживает те предложения пользователей, которые выполняются благодаря системной привилегии SELECT ANY TABLE.

Можно отслеживать использование любой системной привилегии. Во всех случаях аудитинга привилегий, перед проверкой системных привилегий выполняется проверка привилегий владельца и объектных привилегий. Если установлены аналогичные опции как для аудитинга предложений, так и для аудитинга привилегий, то будет сгенерирована лишь одна аудиторская запись. Например, если отслеживаются как опция предложения TABLE, так и системная привилегия CREATE TABLE, то при каждом создании таблицы генерируется только одна аудиторская запись.

Аудитинг привилегий более сфокусирован, чем аудитинг предложений, потому что здесь каждая опция отслеживает лишь специфические типы предложений, а не список связанных предложений. Например, опция аудитинга предложений TABLE отслеживает предложения CREATE TABLE, ALTER TABLE и DROP TABLE, тогда как опция аудитинга привилегий CREATE TABLE отслеживает только предложение CREATE TABLE, ибо лишь для этого предложения требуется системная привилегия CREATE TABLE.

Аудитинг привилегий может быть широким, отслеживая действия всех пользователей базы данных, или узким, отслеживая действия выбранных пользователей.

Аудитинг объектов

Аудитинг объектов - это выборочное отслеживание специфических предложений DML (включая запросы), а также предложений GRANT и REVOKE, для конкретных объектов схем.

Аудитинг объектов отслеживает те операции, которые выполняются благодаря объектным привилегиям, например, предложения SELECT и DELETE по данной таблице.

Можно отслеживать предложения, которые обращаются к таблицам, обзорам, последовательностям, НЕЗАВИСИМЫМ хранимым процедурам и функциям, а также пакетам. (Нельзя индивидуально отслеживать процедуры и функции, определенные внутри пакетов.) Заметьте, что предложения, которые обращаются к кластерам, связям баз данных, индексам или синонимам, непосредственно не отслеживаются; однако доступ через один из таких объектов может быть косвенно отслежен через аудит операций, воздействующих на базовую таблицу. Опции аудитинга объектов всегда действуют на всех пользователей базы данных; нельзя установить эти опции лишь для выбранного списка пользователей. ORACLE предоставляет механизм установки умалчиваемых опций аудита объектов для всех отслеживаемых объектов схем.

Опции аудита объектов для обзоров и процедур

Так как обзоры и процедуры (включая хранимые функции, пакеты и триггеры) обращаются в своих определениях к базовым объектам, аудитинг для обзоров и процедур обладает некоторыми уникальными свойствами. Так, в результате использования обзора или процедуры потенциально может быть сгенерировано несколько аудиторских записей. Под действие включенных опций аудитинга попадает не только факт использования обзора или процедуры, но и предложения SQL, выполняемые в результате обращения к этому обзору или процедуре, подвергаются аудитингу согласно действующим опциям аудитинга объектов (в том числе умалчиваемых опций аудитинга).

ORACLE позволяет фокусировать все виды аудита в двух областях:

- по успешным или безуспешным выполнениям отслеживаемого предложения SQL;

- по режиму генерации аудиторских записей: BY SESSION или BY ACCESS;

Кроме того, аудитинг предложений и привилегий можно проводить для специфических пользователей или для всех пользователей в базе данных.

Для всех видов аудитинга, ORACLE разрешает проводить выборочный аудит успешных выполнений предложений, безуспешных выполнений предложений, либо и тех, и других. Это позволяет вам отслеживать действия пользователей, даже если отслеживаемые предложения не выполняются успешно.

Безуспешное выполнение предложения может быть отслежено лишь в том случае, если было выдано законное предложение SQL, сбившееся из-за отсутствия требуемых полномочий или из-за обращения к несуществующему объекту; оно не может быть отслежено, если выдано синтаксически некорректное предложение SQL. Так, включенная опция аудитинга привилегий, требующая отслеживания безуспешных выполнений предложений, будет регистрировать те

предложения, которые используют соответствующую системную привилегию, но сбиваются по другим причинам (например, когда привилегия CREATE TABLE установлена, но предложение CREATE TABLE сбивается из-за недостатка квоты в табличном пространстве).

Используя любую форму команды AUDIT, вы можете включить:

- опцию WHENEVER SUCCESSFUL, чтобы отслеживать только успешные выполнения отслеживаемого предложения;

- опцию WHENEVER NOT SUCCESSFUL, чтобы отслеживать только безуспешные выполнения отслеживаемого предложения;

- ни одной из этих опций, чтобы отслеживать как успешные, так и безуспешные выполнения отслеживаемого предложения.

Аудитинг BY SESSION и BY ACCESS

Большинство опций аудитинга позволяют указать, как должны генерироваться аудиторские записи, когда отслеживаемое предложение выдается многократно в течение одной сессии пользователя. Следующие секции описывают различия между опциями BY SESSION и BY ACCESS команды AUDIT.

BY SESSION вставляет в аудиторский журнал лишь одну аудиторскую запись для отслеживаемого действия, для данного пользователя и данного объекта, на все время сессии.

Чтобы продемонстрировать, как генерируются аудиторские записи в режиме BY SESSION, рассмотрим следующий пример:

Предположим следующее:

- Установлена опция аудитинга предложений SELECT TABLE в режиме BY SESSION.

- Пользователь JWARD подключается к базе данных и выдает пять предложений SELECT для таблицы DEPT, после чего отключается от базы данных.

- Пользователь SWILLIAMS подключается к базе данных и выдает три предложения SELECT для таблицы EMP, после чего отключается от базы данных.

В этом случае аудиторский журнал будет содержать две аудиторские записи для восьми выданных предложений SELECT (по одной для каждой сессии).

Опцию BY SESSION можно использовать и в тех случаях, когда аудиторские записи направляются в аудиторский журнал операционной системы, но при этом аудиторская запись генерируется и записывается при каждой попытке доступа. Поэтому режим BY SESSION в этой конфигурации эквивалентен режиму BY ACCESS.

BY ACCESS вставляет в аудиторский журнал аудиторскую запись при каждом выполнении отслеживаемого предложения.

Пример, предположим следующее:

- Установлена опция аудитинга предложений SELECT TABLE в режиме BY SESSION.

- Пользователь JWARD подключается к базе данных и выдает пять предложений SELECT для таблицы DEPT, после чего отключается от базы данных.

- Пользователь SWILLIAMS подключается к базе данных и выдает три предложения SELECT для таблицы DEPT, после чего отключается от базы данных.

В этом случае аудиторский журнал будет содержать восемь аудиторских записи для восьми выданных предложений SELECT.

Команда AUDIT позволяет вам специфицировать как режим BY SESSION, так и режим BY ACCESS; по умолчанию принимается режим BY SESSION. Однако некоторые опции аудитинга могут быть установлены только в режиме BY ACCESS, в том числе:

- все опции аудитинга предложений для предложений DDL;

- все опции аудитинга привилегий для предложений DDL.

Аудитинг по пользователям

Опции аудитинга предложений и привилегий могут быть либо широкими, отслеживая предложения, выданные любым пользователем, либо сфокусированными, отслеживая предложения, выдаваемые конкретным списком пользователей. Эта возможность позволяет минимизировать количество генерируемых аудиторских записей, ограничивая их лишь определенными пользователями.

Практика

Практика 1. Настройка структур памяти при создании основных объектов БД

Создание базы данных складов

Рассмотрим пример создания таблиц с заданием параметров управления памятью с учетом операций, которые чаще всего будут выполняться над этими таблицами. Эта схема данных будет в дальнейшем использоваться в лекциях.

Схема базы данных приведена на рисунке 11.

Рисунок 11 - Схема БД

Описание структур каждой таблицы приведено в таблицах 4-6.

Таблица 4 - Таблица «Склады» (Warehouses)

Имя поля таблицы	Тип данных	Описание поля
Ware_id	NUMBER(3) NOT NULL PRIMARY KEY	Номер склада

Address	VARCHAR2(30)	Адрес склада
Volume	NUMBER(6,2)	Общий объем склада
Volume_rest	NUMBER(6,2)	свободный объем склада
Storage_have	VARCHAR2(40)	тип склада

Таблица 5 - Таблица «Товары» (Goods)

Имя поля таблицы	Тип данных	Описание поля
Goods_id	NUMBER(3) NOT NULL PRIMARY KEY	Номер товара
Name	VARCHAR2(20)	Название товара
Volume	NUMBER(6,2)	Объем единицы товара
Storage_need	VARCHAR2(20),	Условие хранения товара
Price	NUMBER(6,2)	Цена единицы товара

Таблица 6 - Таблица «Хранение» (Storages)

Название поля	Тип данных	Описание полей
Storage_id	NUMBER(5) NOT NULL PRIMARY KEY	Номер партии товара
Ware_id	NUMBER(3)	Номер склада
Goods_id	NUMBER(5)	Номер товара
Quantity	INTEGER	Количество товара в партии
Begin_time	DATA	Время начала хранения партии товара
End_time	DATA	Время окончания хранения партии товара

Итак, вы можете создать выше описанные таблицы следующими командами:

- Таблица «Склады»: основные действия - чтение информации, редактирование; объем таблицы существенно не увеличивает.

```
CREATE TABLE WAREHOUSES (

WARE_ID NUMBER(3) NOT NULL PRIMARY KEY,

ADDRESS VARCHAR2(30),

VOLUME NUMBER(6,2),

VOLUME_REST NUMBER(6,2),

STORAGE_NAVE VARCHAR2(40))

PCTFREE 10

PCTUSED 80
```

```
STORAGE (
INITIAL 50K
NEXT 50K
MAXEXTENTS 5
PCTINCREASE 0);
```

- Таблица «Товары»: часты операции добавления и удаления, но редактирование существенно не изменяет объем таблицы.

```
CREATE TABLE GOODS (
GOODS_ID NUMBER(4) NOT NULL PRIMARY KEY,
NAME VARCHAR2(20),
VOLUME NUMBER(6,2),
STORAGE_NEED VARCHAR2(20),
PRICE NUMBER(6,2))
PCTFREE 10
PCTUSED 60
STORAGE (
INITIAL 100K
NEXT 100K
MAXEXTENTS 20
PCTINCREASE 10);
```

- Таблица «Хранение»: часты операции добавления и удаления, но редактирование объем таблицы практически не изменяет.

```
CREATE TABLE STORAGES (
STORAGE_ID NUMBER(5) NOT NULL PRIMARY KEY,
WARE_ID NUMBER(3) REFERENCES WAREHOUSES (WARE_ID),
```

```
GOODS_ID NUMBER(5) REFERENCES GOODS (GOODS_ID),
QUANTITY INTEGER,
BEGIN_TIME DATE,
END_TIME DATE)
PCTFREE 5
PCTUSED 60
STORAGE (
INITIAL 200K
NEXT 200K
MAXEXTENTS 50
PCTINCREASE 20);
```

Создание базы данных магазина

Рассмотрим примеры создания таблиц базы данных магазина с заданием параметров управления памятью с учетом операций, которые чаще всего будут выполняться над этими таблицами.

Схема базы данных приведена на рисунке 12.

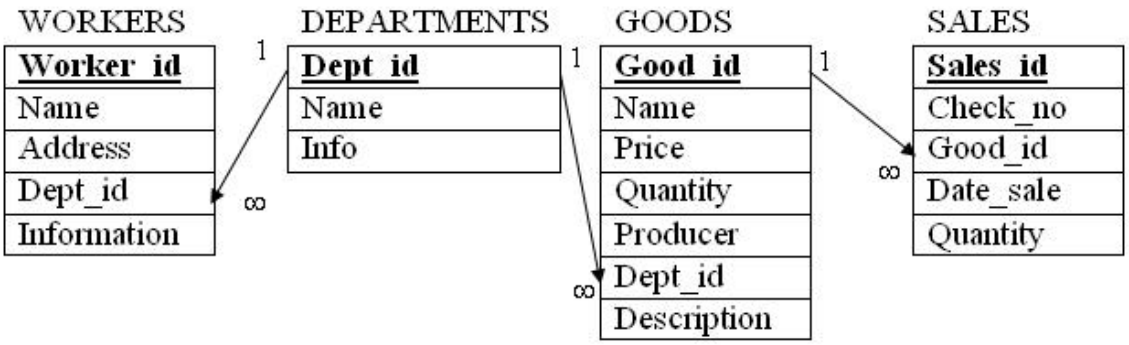


Рисунок 12 - Схема БД магазина

Описание структур каждой таблицы приведено в таблицах 7-10.

Таблица 7 - Таблица «Отделы» (Departments)

Имя поля таблицы	Тип данных	Описание поля
DEPT_ID	NUMBER (4) NOT NULL PRIMARY KEY	Идентификатор отдела
NAME	VARCHAR2(20)	Название отдела
INFO	VARCHAR2(40)	Информация об отделе

Таблица 8 - Таблица «Товары» (Goods)

Имя поля таблицы	Тип данных	Описание поля
Good_id	NUMBER(4) NOT NULL PRIMARY KEY	Номер товара
Name	VARCHAR2(20)	Название товара
Price	NUMBER(10,2)	Цена за единицу товара
QUANTITY	INTEGER	Количество единиц товара
PRODUCER	VARCHAR2(20)	Производитель товара
DEPT_ID	NUMBER (4) NOT NULL	Код отдела, в котором находится товар
DESCRIPTION	VARCHAR2(50)	Описание товара

Таблица 9 - Таблица «Сотрудники» (Workers)

Название поля	Тип данных	Описание полей
WORKERS_ID	NUMBER (5) NOT NULL PRIMARY KEY	Идентификатор сотрудника
NAME	VARCHAR2(20)	Название отдела
ADDRESS	VARCHAR2(40)	Адрес
DEPT_ID	NUMBER (4) NOT NULL	Код отдела, в котором работает сотрудник
INFORMATION	VARCHAR2(40)	Информация о сотруднике

Таблица 10 - Таблица «Продажи» (Sales)

Название поля	Тип данных	Описание полей
SALES_ID	NUMBER (4) NOT NULL PRIMARY KEY	Код продажи
CHECK_NO	NUMBER (6)	Номер чека, по которому проведена продажа
GOOD_ID	NUMBER (4) NOT NULL	Код товара
DATE_SALE	DATE NOT NULL	Дата продажи
QUANTITY	INTEGER	Количество проданного товара

Обратите внимание на установление связей между соответствующими полями родительских и дочерних таблиц.

Итак, вы можете создать выше описанные таблицы следующими командами:

- Таблица «Отделы»: основные действия - чтение информации, редактирование; объем таблицы существенно не увеличивает.

```
CREATE TABLE Departments (
Dept_id NUMBER (4) NOT NULL PRIMARY KEY,
Name VARCHAR2(20) NOT NULL,
Info VARCHAR2(40) DEFAULT NULL)
PCTFREE 10
PCTUSED 80
STORAGE (INITIAL 7K
NEXT 7K
PCTINCREASE 0
MAXEXTENTS 10);
```

- Таблица «Рабочие»: бывают операции добавления и удаления, но редактирование существенно не изменяет объем таблицы.

```
CREATE TABLE WORKERS (
WORKER_ID NUMBER (5) NOT NULL PRIMARY KEY,
NAME VARCHAR2 (20),
ADDRESS VARCHAR2(40),
DEPT_ID NUMBER(4) NOT NULL REFERENCES DEPARTMENTS (DEPT_ID),
INFORMATION VARCHAR2(40) DEFAULT NULL)
PCTFREE 10
PCTUSED 80
STORAGE (INITIAL 10K
NEXT 10K
```

PCTINCREASE 0

MAXEXTENTS 10);

Самостоятельно напишите команды создания таблиц Goods и Sales

Заполнение таблиц

Добавление записей в создании таблицы производится с помощью команды INSERT:

INSERT INTO Departments VALUES (1,'Fruits', NULL);

Напишите команды заполнения таблиц так, чтобы в них было по 5-6 записей.

Лабораторная работа 1. Работа с основными объектами базы и служебной информацией

Цель работы

Ознакомление и приобретение практических навыков работы с утилитой SQL*Plus. Закрепление теоретических знаний по командам языка SQL создания и работы с таблицами баз данных. Приобретение практических навыков создания различных объектов базы данных (пользовательских таблиц, индексов, синонимов) с использованием утилиты SQL*Plus, а также навыков использования словарей данных для получения информации о параметрах распределения памяти, настройках Oracle .

Методические указания к выполнению работы

При подготовке к выполнению работы необходимо ознакомиться с материалами этого тематического раздела. Ознакомится с синтаксисом создания таблиц, индексов, синонимов, предоставления привилегий. Вы должны иметь представление об основных концепциях сервера Oracle , процессах распределения памяти, ведении словарей данных и т.п.

Ход работы

Запустить утилиту SQL*PLUS и зарегистрироваться как один из стандартных пользователей базы данных.

Разработать свою схему базы данных. В схеме должно быть не менее трех таблиц, со связями «один - ко - многим».

Написать скрипты на создание таблиц для разработанной схемы данных (CREATE table...) с учетом распределения памяти и поддержки ссылочной целостности.

При создании таблиц задайте параметры управления памятью с учетом операций, которые чаще всего будут выполняться над этими таблицами.

С помощью словарей данных получите информацию о созданных таблицах (из словаря данных USER _ TABLES).

Написать команды заполнения таблиц (по 3-6 записей в каждой таблице).

Написать команды создания основных объектов БД в своей схеме данных (индексов, синонимов, привелегий и представлений). Результаты выполнения всех команд по созданию этих объектов необходимо находить в словарях данных и отразить их в отчете.У таблицы, для которой был создан индекс, следует просмотреть в словаре данных количество индексов. Объяснить результат.

При создании других объектов БД продолжайте находить информацию об этих объектах в словарях данных: информации о собственном пользовательском статусе (представление USER _ USERS), о других пользователях (ALL _ USERS), о собственных таблицах (USER _ TABLES), ролях (USER _ ROLE _ PRIVS) и т.д..

Рекомендация: перед выборкой значений из таблицы (представления/словаря данных) с неизвестным количеством, названиями и типами полей использовать DESC (DESCRIBE) (команда SQL * Plus), например:

```
DESC ALL _ USERS ;
```

Через словари данных можно просмотреть:

- информацию об объектах пользователя (представление USER _ OBJECTS), например, о функциях:

```
SELECT *FROM USER_OBJECTS WHERE OBJECT_TYPE='FUNCTION';
```

- исходные тексты процедур и функций:

```
SELECT TEXT FROM USER_SOURCE WHERE NAME=' < имя процедуры или функции > ';
```

Просмотреть информацию о представлениях можно в словаре USER _ VIEWS, об индексах - в словаре USER _ INDEXES, о представлениях - в словаре USER _ VIEWS , о синонимах - в словаре USER _ SYNONYMS, о привилегиях пользователя - в словаре USER _ TAB _ PRIVS.

Вопросы для самоконтроля:

а) Приведите синтаксис удаления синонима, представления.

б) Приведите синтаксис просмотра структуры словаря данных.

в) Как создать представление с вычислимыми полями. Приведите примеры.

г) Как создать таблицу с проверкой ограничений целостности?

д) Какие параметры управления распределения памяти Вы знаете?

Указания к оформлению отчета

Отчеты по лабораторной работе можно оформлять и представлять преподавателю в электронном виде или в твердой копии.

Отчет должен содержать:

- название работы;

- цель работы;

- формулировку заданий для самостоятельного выполнения;

- разработанную схему данных;

- команды создания таблиц;

- команды заполнения таблиц;

- команды создания других объектов базы данных;

- результаты выполненных заданий, включающие запросы к словарям данных, а также экранные формы утилиты SQL * Plus , демонстрирующие вывод этих запросов;

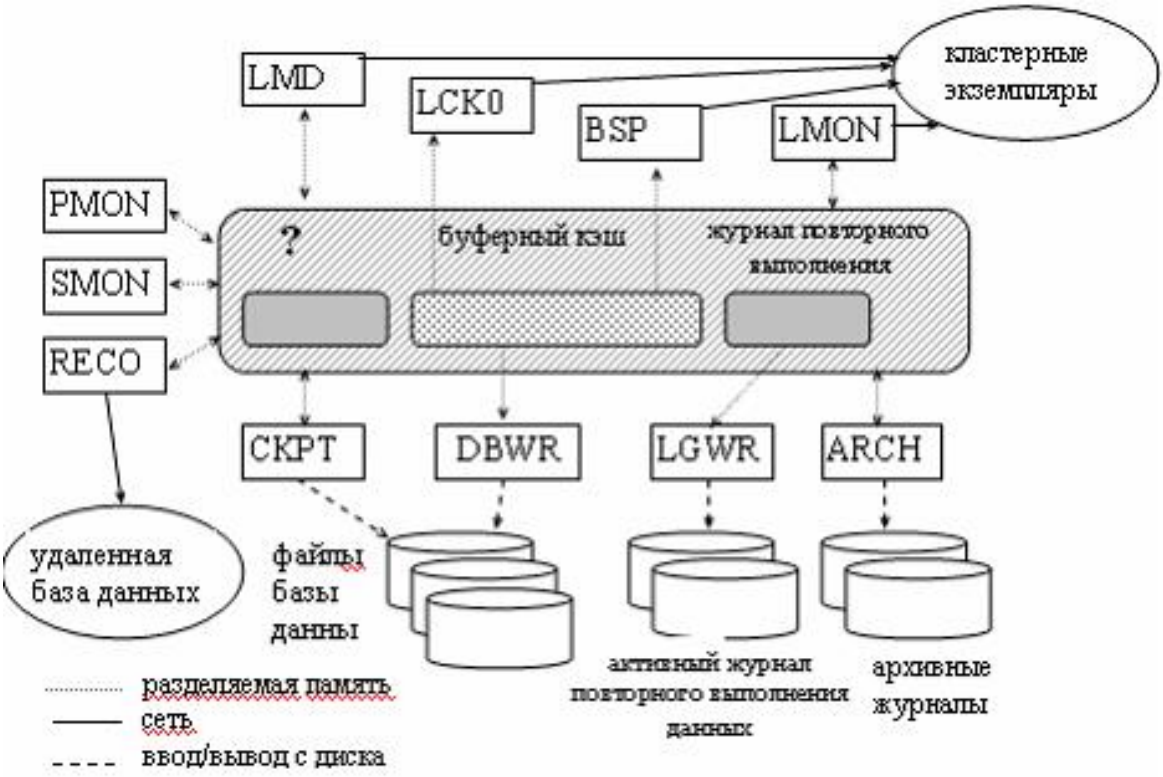
- выводы по работе.

Текущий контроль знаний

Введение в сервер Oracle

Физическая и логическая структура Oracle.

Какая область системы соответствует заштрихованному блоку из на схеме?



- SGA - глобальная область системы
- PGA - глобальная область процесса
- UGA - глобальная область пользователя

Физическая и логическая структура Oracle.

Создание нескольких табличных пространств позволяет достичь следующих целей:

❑ Выполнить настройку производительности приложений путем распределения объектов по табличным пространствам

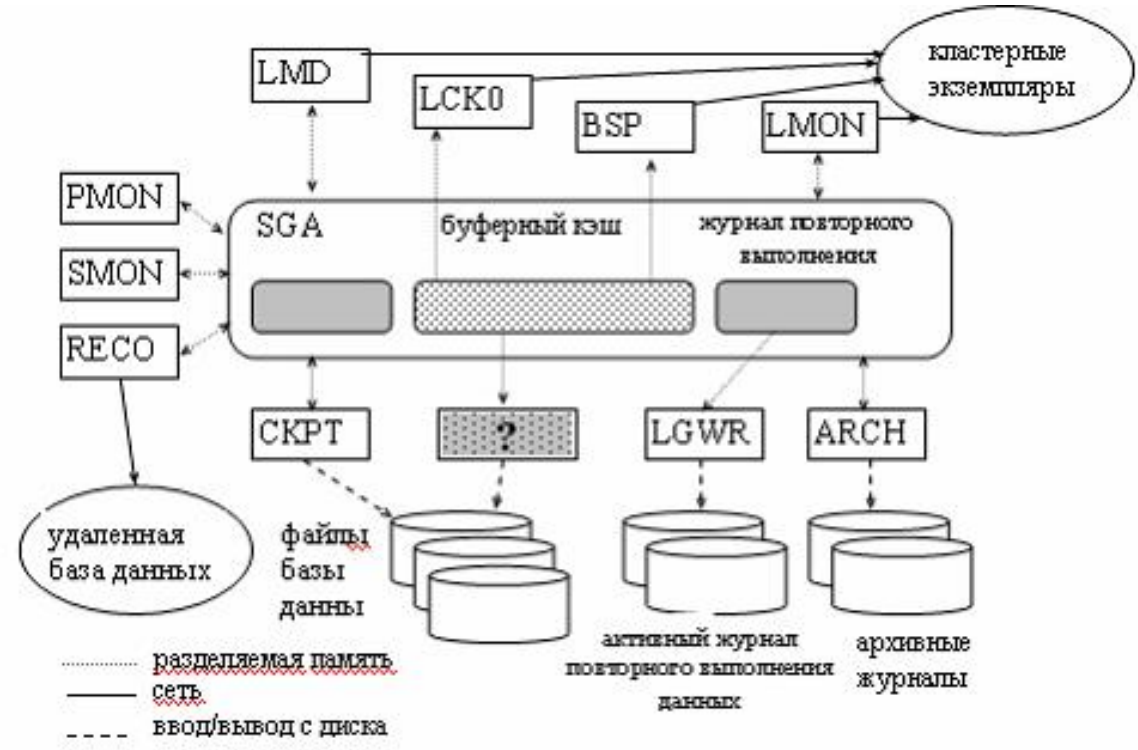
❑ Обеспечить возможность переноса приложения на более мощную компьютерную платформу

❑ Определить разные параметры распределения памяти, применяемые по умолчанию.

❑ Все перечисленное

Физическая и логическая структура Oracle.

Какой из обязательных процессов не указан на схеме?



❑ PMON (монитор процессов)

❑ DBWR (запись блоков базы данных)

- ❑ QMNn (монитор очередей)
- ❑ SNPn (обработка списков)

Физическая и логическая структура Oracle.

Какой процесс выполняет автоматическое восстановление экземпляра?

- ❑ PMON
- ❑ ARCH
- ❑ SMON
- ❑ DBWR

Физическая и логическая структура Oracle.

Когда освобождаются экстенты?

- ❑ Только при удалении таблицы.
- ❑ При автоматическом освобождении экстентов (при превышении размеров сегмента отката)
- ❑ Толька при удалении индексов или ассоциированных таблиц.

Физическая и логическая структура Oracle.

Для чего администратор базы данных не может использовать табличные пространства?

- ❑ Для управления распределением памяти для объектов базы данных.
- ❑ Для резервного копирования данных.

❑ Для установления квот памяти для пользователей базы данных.

❑ Для управления доступностью данных путем перевода отдельных табличных пространств в состояния online или offline.

❑ Для распределения данных по устройствам для повышения производительности.

❑ Для копирования и восстановления данных.

❑ Для хранения словарей базы данных.

Физическая и логическая структура Oracle.

Какие из приведенных высказываний неверны?

❑ Файлы данных табличного пространства физически хранят соответствующие данные базы данных на диске.

❑ Сегмент объекта может размещаться лишь в одном табличном пространстве базы данных.

❑ Пространство для сегмента данных таблицы распределяется лишь в одном файле данных.

❑ Каждое табличное пространство базы данных ORACLE составлено из одного или нескольких файлов данных.

❑ При создании объекта схемы, такого как таблица или индекс, не создается автоматически сегмент этого объекта в табличном пространстве базы данных.

Физическая и логическая структура Oracle.

Что представляет специфическое число смежных блоков данных, распределяемых для хранения специфического типа информации?

❑ Экстент

❑ Сегмент

❑ Разделяемый пул

❑ Блоки данных

Физическая и логическая структура Oracle.

Какие параметры используется для оптимизации использования пространства в блоках данных экстентов внутри сегмента данных?

- ❑ Только PCTFREE
- ❑ Только PCTUSED
- ❑ PCTFREE, PCTUSED
- ❑ FREELISTS
- ❑ MINEXTENTS, MAXEXTENTS
- ❑ INITRANS

Физическая и логическая структура Oracle.

Какие области памяти используются для кэширования информации словаря данных?

- ❑ Кэш буфер базы данных.
- ❑ PGA
- ❑ Журнальный буфер.
- ❑ Разделяемый пул.

Объекты схемы базы данных Oracle

Объекты схемы базы данных Oracle

Ограничения предназначены для выполнения следующих функций:

- ❑ поддержание родительско-дочерних связей между таблицами;
- ❑ обеспечения наличия в поле только данных определенного типа;
- ❑ обеспечение уникальности данных столбца;
- ❑ все перечисленное

Объекты схемы базы данных Oracle

При выборе какого поля происходит увеличение значения последовательности?

- ❑ CURRVAL
- ❑ NEXTVAL
- ❑ LASTNUMBER
- ❑ PREVAL

Объекты схемы базы данных Oracle

Что из перечисленного не является объектом Oracle?

- ❑ Индекс
- ❑ Таблица
- ❑ Запись
- ❑ Транзакция

- Последовательность
- Пакет

Объекты схемы базы данных Oracle

В каких случаях после выполнения приведенных команд будут созданы индексы?

- Create table T1(id_ number unique, str varchar2(20));
- Create table T1(id_ number not null, str varchar2(20));
- Create table T1(id_ number, str varchar2(20)); create index i1 on t1(id_);
- Create table T1(id_ number foreign key, str varchar2(20));
- Create table T1(id_ number primary key, str varchar2(20));
- Create table T1(id_ number, str varchar2(20)); Alter table T1 add index i1(id_);

Объекты схемы базы данных Oracle

Индекс с обратным ключом целесообразно использовать в следующих ситуациях:

- Значение ключа индекса формируется на основе последовательности Oracle.
- Доступ обычно осуществляется для выборки диапазона значений.
- Значение ключа индекса всегда является уникальным.

Объекты схемы базы данных Oracle

Какой метод способ доступа к отдельной строке в таблице является самым быстрым?

- Использование уникального индекса.
- Использование первичных ключей.

- Использование ROWID.

Объекты схемы базы данных Oracle

Какой вариант является верным для создания таблицы «Служащие» (Workers), содержащей поля:

Id_ Идентификационный код

Name ФИО

Address Адрес

Birthday Дата рождения

Position Должность

Salary Зарплата

- Create or replace table workers

(id_ number(6),

name varchar2(30),

address varchar2(30),

birthday date,

position varchar2(15),

salary number (6,2));

- Create table workers

(id_ number(6),

name varchar2(30),

address varchar2(30),

birthday date,

position varchar2(15),

salary number (6,2));

- Create or replace table

(id_ number(6),

name varchar2(30),

address varchar2(30),

birthday date,

position varchar2(15),

salary number (6,2));

- Create table (id_ number(6),

name varchar2(30),

address varchar2(30),

birthday date,

position varchar2(15),

salary number (6,2))

workers;

Объекты схемы базы данных Oracle

Каким образом организуется поддержание ссылочной целостности при создании таблицы «Дети» (Children) (id_child (номер записи о детях), id_ (идентификационный код служащего), name (ФИО ребенка), birthday (дата рождения ребенка) с таблицей «Служащие» (Id_ (Идентификационный код - ключевое поле), Name (ФИО), Address (Адрес), Birthday (Дата рождения), Position (Должность), Salary (Зарплата)) и таблицей)

- Create table Children (id_child integer not null primary key,

id_ number(6) foreign key,

name varchar2(30),

birthday date);

- Create or replace table Children (id_child integer primary key,

id_ number(6) primary key,

name varchar2(30),

birthday date);

- Create table Children (id_child integer not null references,

id_ number(6) foreign key (workers),

name varchar2(30),

birthday date);

- Create table Children (id_child integer not null primary key,

id_ workers.id_%type references workers,

name varchar2(30),

birthday date);

Объекты схемы базы данных Oracle

Использованием каких команд можно удалить таблицу Children?

- Drop Children;
- Delete from Children;
- Drop object Children;
- Drop table Children;
- Drop table Children cascade constraints;
- Delete table Children cascade constraints;

Объекты схемы базы данных Oracle

Каким образом может быть создана копия объекта?

- Использованием оператора CREATE TABLE table_name1 AS SELECT ...
- Использованием оператора CREATE TABLE table_name1 AS COPY ...
- Использованием оператора CREATE COPY OBJECT table_name1...
- Использованием команды INSERT INTO table_name1 SELECT

Словарь терминов