Государственный комитет Российской федерации
по высшему образованию.
ГОСУДАРСТВЕННЫЙ САНКТ-ПЕТЕРБУРГСКИЙ ИНСТИТУТ ТОЧНОЙ МЕХАНИКИ И ОПТИКИ (ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)

кафедра вычислительной техники

Администрирование баз данных ORACLE

Санкт-Петербург
2000 год

ОГЛАВЛЕНИЕ.

1. Обязанности администратора базы данных (АБД)

Поскольку система баз данных ORACLE может быть весьма большой и может иметь много пользователей, должно существовать лицо или группа лиц, управляющих этой системой. Такое лицо называется администратором базы данных (АБД).

В любой базе данных должен быть хотя бы один человек, выполняющий административные обязанности; если база данных большая, эти обязанности могут быть распределены между несколькими администраторами.

В обязанности администратора могут входить:

- инсталляция и обновление версий сервера ORACLE и прикладных инструментов
- распределение дисковой памяти и планирование будущих требований системы к памяти
- создание первичных структур памяти в базе данных (табличных пространств) по мере проектирования приложений разработчиками приложений
- создание первичных объектов (таблиц, представлений, индексов) по мере проектирования приложений разработчиками
- модификация структуры базы данных в соответствии с потребностями приложений
- зачисление пользователей и поддержание защиты системы
- соблюдение лицензионного соглашения ORACLE
- управление и отслеживание доступа пользователей к базе данных
- отслеживание и оптимизация производительности базы данных
- планирование резервного копирования и восстановления
- поддержание архивных данных на устройствах хранения информации
- осуществление резервного копирования и восстановления
- обращение в корпорацию Oracle за техническим сопровождением

Сотрудники службы безопасности

В некоторых случаях база данных должна также иметь одного или нескольких сотрудников службы безопасности. СОТРУДНИК СЛУЖБЫ БЕЗОПАСНОСТИ главным образом отвечает за регистрацию новых пользователей, управление и отслеживание доступа пользователей к базе данных, и защиту базы данных.

Разработчики приложений

В обязанности разработчика приложений входит:

- проектирование и разработка приложений базы данных
- проектирование структуры базы данных в соответствии с требованиями приложений
- оценка требований памяти для приложения
- формулирование модификаций структуры базы данных для приложения
- передача вышеупомянутой информации администратору базы данных
- настройка приложения в процессе его разработки
- установка мер по защите приложения в процессе его разработки

2. Подключение в режиме INTERNAL

Запуск и останов базы данных - это мощные административные возможности. В угоду поддержания корректной работоспособности базы данных, функции(команды **STARTUP** или **SHUTDOWN**) остановки и запуска разрешены, только для администратора подключенного к ORACLE в режиме NTERNAL(**CONNECT INTERNAL**), а для возможности подключиться в режиме NTERNAL, вы должны соотвествовать одному из ниже следующих условий:

- Ваше учетное имя в операционной системе имеет привилегии операционной системы, позволяющие вам соединяться в режиме INTERNAL.
- Вы имеете полномочия соединяться в режиме INTERNAL.
- Ваша база данных имеет пароль для INTERNAL, и вы знаете этот пароль.

Все эти требования образуют дополнительный слой защиты, препятствующий несанкционированному запуску или остановке баз данных ORACLE. Для систем, имеющих пароль для INTERNAL, существуют дополнительные, ниже описываемые соображения.

Использование пароля для INTERNAL

Некоторые операционные системы позволяют устанавливать пароль для соединений в режиме INTERNAL. Можно установить пароль для INTERNAL во время инсталляции сервера ORACLE, Oracle предоставляет утилиту для управления этим паролем (создания, изменения и удаления его).

INTERNAL и незащищенные соединения

Если используется незащищенное соединение(как большинство сетевых соединений), то ДОЛЖНО использовать пароль для INTERNAL, для последующего подключения в режиме INTERNAL; это требование подразумевает, что в системе должен быть установлен пароль для INTERNAL.

В некоторых О.С. можно либо включить, либо полностью отключить возможность соединений CONNECT INTERNAL для незащищенных соединений. Выбор делается во время инсталляции ORACLE, и может быть изменен позднее.

3. Утилиты АБД (Import, Export, Loader)

SQL*Loader

Одной из многих проблем, с которыми часто сталкиваются администраторы базы данных, является перемещение данных из внешних источников в базу данных Oracle. Сложность этой задачи возрастает с появлением хранилищ данных, приходится перемещать уже не мегабайты данных, а гигабайты, а в некоторых случаях – терабайты. Oracle предусматривает для решения этой задачи SQL*Loader – универсальное инструментальное средство, которое загружает внешние данные в таблицы базы данных Oracle. Утилита SQL*Loader является гибкой и настраиваемой до такой степени, что часто удается обойтись без процедур на языке третьего поколения с внедренными операторами SQL. Каждый раз, сталкиваясь с задачей преобразования инородных данных в формат Oracle, вначале рассмотрите возможность применения SQL*Loader, прежде чем обращаться к другим средствам.

Основные компоненты SQL*Loader

Для утилиты SQL*Loader необходимы входные данные 2-ух типов: внешние данные, которые могут находиться на диске или ленте, и управляющая информация (содержащаяся в управляющем файле), которая описывает характеристики входных данных. Выходные данные, часть которых является необязательной, включает таблицы Oracle, журналы, файлы некорректных записей и файлы отвергнутых записей.

Входные данные

Утилита SQL*Loader может обрабатывать файлы данных практически любого типа и поддерживает собственные типы данных почти любой платформы. Данные обычно считываются из одного или нескольких файлов данных, однако они могут быть также внесены в управляющий файл после управляющей информации. Файл данных может находиться:

В файлах с переменным форматом данные находятся в записях, которые могут изменяться по длине, в зависимости от размеров данных в полях. Поля имеют длину, необходимую для размещения данных. Поля в файлах с переменным форматом могут быть разделены завершающими символами (такими как запятые и пробелы), а так же заключены в ограничительные символы.

Управляющий файл

Прежде чем утилита SQL*Loader сможет обработать данные в файлах данных, необходимо знать орпеделения даных для SQL*Loader. Используйте управляющий файл для указания физических определений файла данных, а также формата данных в файлах. Упраляющий файл – это файл произвольного формата, который также содержит дополнительные управляющие данные, указывающие SQL*Loader, как обрабатывать эти данные.

Журнал

После выполнения утилита SQL*Loader создает журнал, содержащий подробную информацию о загрузке, включая, следующие сведения:

- Имена фалов входных данных, управляющего файла, файлов некорректных записей и файлов отвергнутых записей.
- Входные данные и связанные с ними определения таблиц
- Ошибки SQL*Loader
- Результаты работы SQL*Loader
- Итоговую статистику

Import и Export

Import и Export — две дополнительные утилиты, поставляемые корпорацией Oracle. Они в основном применяются для копирования и восстановления данных и для перемещения данных либо в другую базу данных Oracle, либо из более старой версии Oracle в более новую. Ниже приведены другие возможности утилит Import и Export :

- Хранение данных в файлах операционной системы для архивирования
- Выборочное резервное копирование частей базы данных
- Перемещение данных из одной пользовательской схемы Oracle в другую
- Перемещение данных с одной аппаратной платформы или операционной системы в другую
- Экономия пространства и повышение производительности за счет уменьшения фрагментации

Работа с утилитами Import и Export весьма проста. Утилита Export записывает информацию о таблицах или объектах базы данных, такую как операторы создания таблицы, операторы создания индекса, разрешения на таблицу, информация о размерах и т.д., а также данные из самих таблиц Oracle. Затем утилита Export сохраняет эту информацию в именованных файлах операционной системы. Файлы операционной системы, создаваемые утилитой Export, известны как файлы дампа. Файлы дампа, которые представлены в двоичном формате Oracle, применяются главным образом только в утилите Import. Можно назвать Файл дампа любым именем, допустимым в операционной системе. Если вы не укажите имя выходного файла для утилиты Export, то по умолчанию будет принято имя EXPDAT.DMP.

Затем можно сохранить выходные файлы, созданные Export, на диске или записать на съемный носитель для дальнейшего хранения, либо воспользоваться утилитой Import для воссоздания экспортируемых данных в целях восстановления или ведения базы данных.

Export

Иногда можно обнаружить, что вы не сделали в свое время то, в чем сейчас остро нуждаетесь. Возьмем, например, утилиты Import и Export. Утилита Export — это самый удобный способ застраховаться от возможных неприятностей. Экспорт — это универсальная утилита, поставляемая корпорацией Oracle. При всей заложенной в ней гибкости ею довольно легко пользоваться на основе обширного списка параметров. Разнообразие параметров дает возможность воспользоваться утилитой Export для решения сложных проблем управления данными. Утилита Export может записывать файлы операционной системы, которые затем можно перемещать в другую операционную систему или версию Oracle.

Проверьте, чтобы для хранения экспортного файла на запоминающем устройстве было достаточно свободного пространства. Можно использовать представление *user_segments* для оценки необходимого дискового пространства.

В следующем коде показан пример использования утилиты Export:

exp userid=system/manager OWNER=scott... *[прочие опции}*

Использование файла параметров

Можно использовать файл параметров как для утилиты Export, так и для утилиты Import. Файл параметров помогает выполнять операции импорта и экспорта, обеспечивая при этом непротиворечивость и простоту. Его удобно использовать для экспорта в ночное время. Файлы параметров гарантируют целостный экспорт для полной уверенности в том, что все необходимые таблицы действительно будут экспортированы. Можно вызывать эти утилиты, задавая параметры в командной строки или в сценарии операционной системы, но использование командной строки может не позволить задать все необходимые параметры. Сценарий *export_ts* записывает необходимый ему файл параметров. Имена таблиц в кавычках являются чувствительными к регистру.

Режим экспорта таблиц

Режим экспорта таблиц используется для экспорта одной таблицы или перечня таблиц, а не всей базы данных. По умолчанию он экспортирует все таблицы, которые принадлежат пользователю, выполняющему экспорт. Пользователи, имеющие доступ к другой схеме, могут экспортировать таблицы из этой схемы, указав имя схемы.

Режим экспорта пользователя

Режим экспорта пользователя в основном используется для экспорта всех таблиц и индексов конкретного пользователя или перечня пользователей. Этот режим работает хорошо при создании пользователя, который является владельцем всех объектов приложения. Например, если существует пользователь с именем sales, который является владельцем всех таблиц и индексов и других объектов в приложении sales, экспорт приложения может выглядеть следующим образом:

exp *VSERlD=system/manager* OWNER=*sales*

Режим экспорта всей базы данных

Режим экспорта всей базы данных используется для экспорта всех объектов базы данных, за исключением объектов, которые обычно создаются и поддерживаются учетной записью SYS. Эту опцию могут применять только пользователи, которым назначена роль EXP_FULL_DATABASE. Здесь можно упомянуть несколько других интересных возможностей. По умолчанию Oracle выполняет полный экспорт при указании режима экспорта всей базы данных (INCTYPE= COMPLETE). Если указана опция INCTYPE= INCREMENTAL, Oracle будет экспортировать только таблицы, содержащие какие-либо изменившиеся строки, начиная с последнего полного экспорта любого типа. Если указана опция INCTYPE=CUMULATIVE, Oracle будет экспортировать только таблицы, содержащие какие-либо измененные строки, начиная с последнего полного или кумулятивного экспорта.

Типы экспорта:

- Полный экспорт
- Инкрементный экспорт
- Кумулятивный экспорт

Import

Утилита Import противоположна утилите Export. Она отвечает за чтение экспортных файлов в целях воссоздания объектов базы данных, а также любого состояния, в котором они экспортировались первоначально. Утилита Import может также преобразовывать данные предоставленные с разных платформ к примеру с UNIX машины в ASCII кодах, на мейнфрем с кодировкой EBCDIC и наоборот, что позволяет перемещать данные с одной платформы на другую. Утилита Import может работать в интерактивном режиме или в режиме командной строки. При использовании интерактивного режима утилита Import запрашивает у пользователя параметры, необходимые для выполнения импорта. Обычно проще задать параметры в командной строке или в файле параметров. Утилита Import, как и Export, использует файлы параметров.

4. Пользователи базы данных и схемы

Каждая база данных ORACLE имеет список имен пользователей. Чтобы получить доступ к базе данных, пользователь должен использовать приложение базы данных, и попытаться соединиться с базой данных, предоставив действительное имя пользователя. С каждым именем пользователя связан пароль, чтобы предотвратить несанкционированный доступ.

С каждым именем пользователя ассоциирована СХЕМА(SCHEMA), имеющая такое же имя. SCHEMA(схема) - это логическая коллекция объектов базы данных (таблиц, представлений, последовательностей, синонимов, индексов, кластеров, процедур, функций, пакетов и связей баз данных). По умолчанию, каждый пользователь базы данных создает и имеет доступ ко всем объектам в соответствующей схеме. Команда CREATE SCHEMA полезна для гарантирования успешного создания всех нужных объектов и грантов за одну операцию; если индивидуальный объект внутри этой операции не может быть создан, все предложение откатывается и аннулируются все его результаты.

Домен защиты

Каждый пользователь имеет ДОМЕН ЗАЩИТЫ - набор свойств, которые определяют такие вещи, как:

- действия (привилегии и роли), доступные пользователю
- квоты табличных пространств (доступной дисковой памяти)
- лимиты на системные ресурсы (например, время процессора) для данного пользователя

Каждое свойство, вносящее вклад в домен защиты пользователя, обсуждается ниже.

Привилегии

ПРИВИЛЕГИЯ(PRIVILEGE) - это право выполнять определенный тип предложений SQL. Некоторые примеры привилегий включают:

- право соединяться с базой данных (создавать сессию)
- право создавать таблицу в вашей схеме
- право выбирать строки из чьей-либо таблицы
- право выполнять чью-либо хранимую процедуру

Привилегии в s базе данных ORACLE могут быть разделены на две различные категории: системные привилегии и объектные привилегии.

Системные привилегии

СИСТЕМНЫЕ ПРИВИЛЕГИИ(SYSTEM_PRIVILEGE) позволяют пользователям выполнять конкретное действие на уровне системы, или конкретное действие над конкретным типом объектов. Таковы, например, привилегия создавать табличное пространство или привилегия удалять строки любой таблицы в базе данных. Большинство системных привилегий доступны только администраторам и разработчикам приложений, поскольку такие привилегии весьма мощны.

Объектные привилегии

ОБЪЕКТНЫЕ ПРИВИЛЕГИИ позволяют пользователям выполнять конкретные действия на конкретном объекте. Такова, например, привилегия удалять строки в указанной таблице. Объектные привилегии назначаются пользователям, так что они могут использовать приложения базы данных для выполнения конкретных задач.

Назначение привилегий

Привилегии назначаются пользователям с тем, чтобы эти пользователи могли обращаться к данным и модифицировать эти данные в базе данных. Пользователь может получить привилегию двумя различными способами:

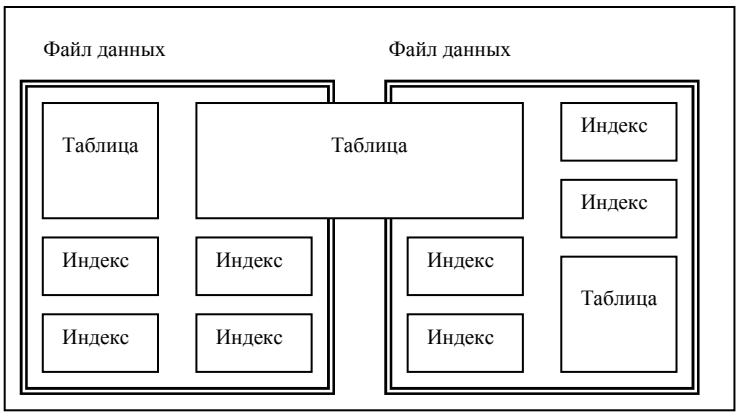
- Привилегии могут назначаться пользователям явно. Например, привилегия вставлять записи в таблицу EMP может быть явно назначена пользователю SCOTT.
- Привилегии могут назначаться РОЛЯМ (именованным группам привилегий), а затем эта роль может быть назначена одному или нескольким пользователям. Например, привилегия вставлять записи в таблицу EMP может быть назначена роли с именем CLERK, а эта роль, в свою очередь, может быть назначена пользователям SCOTT и BRIAN.

Роли (ROLE)

ORACLE предоставляет легкое и контролируемое управление привилегиями через использование ролей. РОЛИ (ROLE) - это поименованные группы взаимосвязанных привилегий, которые назначаются пользователям или другим ролям.

5. Табличные пространства и файлы данных

Используемые данные базы данных ORACLE логически хранятся в ТАБЛИЧНЫХ ПРОСТРАНСТВАХ (TABLESPACE), а физически располагаются в ФАЙЛАХ ДАННЫХ, ассоциированных с соответствующим табличным пространством. Эту взаимосвязь иллюстрирует следующий рисунок:



Файлы данных и табличные пространства

Файлы данных - физические структуры, каждая из которых связана с одним табличным пространством
Объекты - Хранятся в табличных пространствах, и могут располагаться в нескольких файлах данных

Табличные пространства

Каждая база данных ORACLE подразделяется на логические элементы, называемые ТАБЛИЧНЫМИ ПРОСТРАНСТВАМИ(TABLESPACE). 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

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 более гибко осуществлять разнообразные операции администрирования.

Размер табличного пространства - это размер его файла данных или суммарный размер всех файлов данных, составляющих это табличное пространство.

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.

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.

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 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 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.

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.

6.Схемы и объекты схемы

СХЕМА(SCHEMA) - это коллекция объектов. ОБЪЕКТЫ СХЕМЫ - это логические структуры, непосредственно относящиеся к данным базы данных. Объекты схемы включают такие структуры как: таблицы, представления, последовательности, хранимые процедуры, синонимы, индексы, кластеры и связи баз данных. (Не существует взаимосвязи между табличным пространством и схемой; объекты одной и той же схемы могут находиться в разных табличных пространствах, и одно и то же табличное пространство может содержать объекты из разных схем.)

ТАБЛИЦА(TABLE) - это основная единица хранения данных в базе данных ORACLE. Таблицы базы данных хранят все данные, доступные пользователям.

Данные таблицы хранятся в виде СТРОК и СТОЛБЦОВ. Каждая таблица определяется с ИМЕНЕМ ТАБЛИЦЫ и набором столбцов. Каждому столбцу дается ИМЯ СТОЛБЦА, ТИП ДАННЫХ (такой как CHAR, DATE или NUMBER), а также ШИРИНА (которая может быть предопределена типом данных, как в случае DATE) или МАСШТАБ и ТОЧНОСТЬ (только для типа данных NUMBER). После того, как таблица создана, в нее могут быть вставлены строки действительных данных. После этого строки таблицы можно опрашивать, удалять или обновлять. Чтобы учредить организационные правила для данных таблицы, для таблицы можно также определить ограничения целостности и триггеры.

ПРЕДСТАВЛЕНИЯ(VIEW) - это настроенное представление данных из одной или нескольких таблиц. представлений можно рассматривать как "хранимый запрос". Представления в действительности не содержат данных; вместо этого они доставляют данные из тех таблиц, на которых они основаны (так называемых БАЗОВЫХ ТАБЛИЦ представлений). Базовые таблицы, в свою очередь, могут быть как таблицами, так и представлениями.

Как и таблицы, представления можно опрашивать, обновлять и осуществлять в них вставки и удаления, с некоторыми ограничениями. Все операции, осуществляемые над представлениями, в действительности затрагивают базовые таблицы этого представления.

ПОСЛЕДОВАТЕЛЬНОСТЬ(SEQUENCE) - генерирует уникальные порядковые номера, которые могут использоваться как значения числовых столбцов таблиц базы данных. Последовательности упрощают прикладное программирование, автоматически генерируя уникальные числовые значения для строк одной или нескольких таблиц. Номера, генерируемые последовательностью, независимы от таблиц, так что одну и ту же последовательность можно использовать для нескольких таблиц. После ее создания, к последовательности могут обращаться различные пользователи, чтобы получать действительные порядковые номера.

Термином "программная единица" обозначаются хранимые процедуры, функции и пакеты.

ПРОЦЕДУРА(PROCEDURE) или ФУНКЦИЯ(FUNCTION) - это совокупность предложений SQL и PL/SQL, сгруппированных вместе как выполнимая единица, исполняющая специфическую задачу.

Процедуры и функции сочетают легкость и гибкость SQL с процедурными возможностями языка структурного программирования. С помощью PL/SQL такие процедуры и функции можно определять и сохранять в базе данных для продолжительного использования.

Процедуры и функции похожи друг на друга, с той разницей, что функция всегда возвращает вызывающей программе единственное значение, тогда как процедура в общем случае не возвращает значения, однако существуют специфические методики для возвращения значения процедуры.

ПАКЕТЫ(PACKAGE) дают метод инкапсулирования и хранения взаимосвязанных процедур, функций 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, и увеличивают производительность базы данных.

СИНОНИМ(SYNONYM) - это алиас (дополнительное имя) для таблицы, представлений, последовательности или программной единицы. Синоним не есть объект, но он является прямой ссылкой на объект. Синонимы используются для:

1) маскировки действительного имени и владельца объекта;
2) обеспечения общего доступа к объекту
3) достижения прозрачности местоположения для таблиц, представлений или программных единиц удаленной базы данных
4) упрощения кодирования предложений SQL для пользователей базы данных

Синоним может быть общим или личным. Индивидуальный пользователь может создать ЛИЧНЫЙ СИНОНИМ, который доступен только этому пользователю. Администраторы баз данных чаще всего создают ОБЩИЕ СИНОНИМЫ, благодаря которым объекты базовых схем становятся доступными для общего пользования всем пользователям базы данных.

Индексы, кластеры и хешированные кластеры - это необязательные структуры, ассоциированные с таблицами, которые можно создавать для повышения производительности операций извлечения данных. Так же, как индекс в книге позволяет вам быстрее отыскивать нужную информацию, индекс ORACLE предоставляет быстрый путь доступа к данным таблицы. При обработке запроса ORACLE может использовать некоторые или все имеющиеся индексы для эффективного отыскания запрашиваемых строк. Индексы полезны, когда приложения часто опрашивают интервалы строк таблицы либо отдельные строки.

Индексы создаются по одному или нескольким столбцам таблицы. Однажды созданный, индекс автоматически поддерживается и используется ORACLE. Изменения в данных таблицы автоматически отражаются во всех соответствующих индексах при полной прозрачности для пользователей.
КЛАСТЕРЫ(CLUSTER) предоставляют необязательный способ хранения данных таблиц. Кластер - это группа из одной или нескольких таблиц, физически хранящихся вместе.

Взаимосвязанные столбцы таблиц в кластере называются КЛЮЧОМ КЛАСТЕРА. Ключ кластера индексируется, так что строки кластера могут извлекаться с минимальными затратами на ввод-вывод. Поскольку данные ключа кластера в индексированном (не кэшированном) кластере хранятся в одном экземпляре для всех таблиц кластера, достигается экономия пространства по сравнению с обычными (некластеризованными) таблицами.

Кластеризованные таблицы: Связанные данные хранятся вместе, более эффективно Некластеризованные таблицы: Связанные данные хранятся отдельно, занимая больше места.

Кластеры могут также повысить эффективность извлечения данных, в зависимости от распределения данных и от того, какие операции SQL наиболее часто выполняются на кластеризованных данных. В частности, кластеризованные таблицы, опрашиваемые через соединения, выигрывают за счет кластеров, потому что строки, общие для объединяемых таблиц, извлекаются за одну операцию ввода-вывода. Как и индексы, кластеры не влияют на проектирование приложений. Является ли таблица частью кластера или нет, остается прозрачным для пользователей и приложений. Данные, хранящиеся в кластеризованной таблице, доступны через те же операции SQL, как если бы они не были кластеризованы.

ХЭШИРОВАННЫЕ КЛАСТЕРЫ похожи на обычные, индексированные, кластеры. Однако в хэшированных кластерах строки записываются не на основе ключа кластера, а на основе значения ФУНКЦИИ ХЭШИРОВАНИЯ, применяемой к ключу кластера. Все строки с одинаковым значением такого хэш-ключа хранятся на диске вместе. Хэшированные кластеры выигрывают по сравнению с индексированной таблицей и индексированным кластером, когда таблица часто опрашивается на равенство. Для таких запросов значения указанного ключа кластера хэшируются, и результирующие значения хэш-ключа прямо указывают на участок диска, в котором хранятся соответствующие строки.

СВЯЗЬ БАЗ ДАННЫХ - это именованный объект, который описывает "путь" от одной базы данных к другой. Связи баз данных неявно используются при обращении к ГЛОБАЛЬНОМУ ИМЕНИ ОБЪЕКТА в распределенной базе данных.

7. Блоки данных, экстенты и сегменты.

Отношения между сегментами, экстентами и блоками данных

ORACLE распределяет пространство базы данных для всех ее данных. Единицами логического распределения являются блоки данных, экстенты и сегменты.

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. По контрасту с этим, 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.

ORACLE управляет пространством в файлах данных базы данных в единицах, называемых БЛОКАМИ ДАННЫХ. Блок данных - это наименьшая единица ввода-вывода, используемая базой данных. Блок данных соответствует физическому блоку на диске с размером, совпадающим с размером блока данных ORACLE. Этот размер блока может отличаться от стандартного размера блока ввода-вывода операционной системы, в которой выполняется ORACLE.

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

Экстенты

Следующий уровень логического пространства базы данных называется ЭКСТЕНТОМ(EXTENTS). Экстент - это специфическое число смежных блоков данных, распределяемых для хранения специфического типа информации.

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.

Сегменты

Уровень логического пространства базы данных, следующий за экстентом, называется СЕГМЕНТОМ (SEGMENTS). 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 смежные на диске, и могут быть распределены между различными файлами. Каждый экстент, однако, не может находиться в нескольких файлах.

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

8.Структуры памяти и процессы

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.

ПРОЦЕССЫ(PROCESS) - это задания или задачи, работающие в памяти этих компьютеров. 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. (V$PROCESS - информация об активных и V$BGPROCESS – информация о фоновых процессах)

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: 1) симуляция личных окружений для нескольких одновременно работающих процессов, так, как будто каждый процесс имеет свое собственное личное окружение 2) обеспечение разделения между процессами ресурсов компьютера, необходимых каждому процессу, но не на долгое время

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.

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. Много процессной системе все процессы можно разделить на две группы: пользовательские процессы и процессы ORACLE.

Структуры памяти

ORACLE создает и использует свои структуры памяти для выполнения некоторых задач. Например, память используется для размещения исполняемого программного кода и данных, разделяемых между пользователями. С ORACLE ассоциируются несколько базовых структур памяти: глобальная область системы (которая включает буфера базы данных и журнала повторения, а также разделяемый пул) и глобальные области программ. Следующие секции подробно объясняют каждую из этих видов областей.

Глобальная область системы (SGA)

ГЛОБАЛЬНАЯ ОБЛАСТЬ СИСТЕМЫ (SGA) - это область памяти разделяемой между процессами. Область SGA и фоновые процессы ORACLE составляют инстанцию ORACLE. SGA распределяется при запуске инстанции и освобождается при закрытии инстанции. Для оптимальной производительности SGA должна быть максимально большой (пока позволяет реальная память), чтобы держать как можно больше данных в памяти и минимизировать дисковые операции. Информация, хранящаяся в SGA, подразделяется на

несколько типов структур памяти, включая: буфера базы данных, буфера журнала повторения и разделяемый пул. Эти области имеют фиксированные размеры и создаются при запуске инстанции. Буферный кэш базы данных БУФЕРА БАЗЫ ДАННЫХ в SGA хранят наиболее недавно использовавшиеся блоки данных из базы данных; все множество буферов базы данных в инстанции составляет БУФЕРНЫЙ КЭШ БАЗЫ ДАННЫХ.

9. Пример работы Oracle.

Следующее описание иллюстрирует работу конфигурации ORACLE, при которой пользователь и ассоциированный с ним процесс сервера находятся на разных машинах, соединенных через сеть.

1. ЭКЗЕМПЛЯР - работает на том компьютере, на котором выполняется ORACLE (часто называемом ХОСТОМ или СЕРВЕРОМ БАЗЫ ДАННЫХ, информацию о текущем состоянии сервера БД, экземпляре, можно получить по V$INSTANCE)

ЭКЗЕМПЛЯР (INSTANCE она же инстанция) СУБД Oracle – это набор серверных процессов, объединенных единой глобальной системной областью и связанных с общим набором совместно используемых файлов данных. К примеру если на сервере работает две независимые базы данных, и каждая обладает своей глобальной системной областью и независимым набором сервисных процессов то у нас будет два экземпляра базы данных Oracle; каждому из экземпляров присваевается системный идентификатор SID (system identifier)

2. Компьютер, используемый для приложений (ЛОКАЛЬНАЯ МАШИНА или РАБОЧАЯ СТАНЦИЯ КЛИЕНТА) выполняет приложение в рамках пользовательского процесса. Приложение-клиент пытается установить соединение с сервером, используя подходящий драйвер SQL*Net.
3. Сервер также выполняет соответствующий драйвер SQL*Net. Сервер обнаруживает запрос на соединение от приложения и создает (выделенный) процесс сервера от имени пользовательского процесса.
4. Пользователь выполняет предложение SQL и подтверждает свою транзакцию. Например, пользователь изменяет данные в строке таблицы.
5. Процесс сервера принимает это предложение и проверяет разделяемый пул, чтобы отыскать в нем разделяемую область SQL, содержащую идентичное предложение SQL. Если такая разделяемая область SQL найдена, процесс сервера проверяет пользовательские привилегии доступа к запрашиваемым данным и использует существующую разделяемую область SQL для обработки предложения; если нет, то для предложения распределяется новая разделяемая область SQL, в которой осуществляется разбор и исполнение предложения.
6. Процесс сервера извлекает все необходимые значения данных из файла данных (таблицы) или получает их из SGA.
7. Процесс сервера модифицирует данные в SGA. Процесс DBWR записывает модифицированные данные на диск для постоянного хранения, если сочтет это необходимым. Поскольку транзакция подтверждена, процесс LGWR немедленно регистрирует транзакцию в онлайновом файле журнала повторения.
8. Если транзакция успешна, процесс сервера посылает сообщение через сеть приложению. В противном случае передается соответствующее сообщение об ошибке.
9. Во время всей описанной процедуры работают другие фоновые процессы, наблюдая за условиями, которые могут потребовать вмешательства. Кроме того, сервер базы данных управляет транзакциями других пользователей и предотвращает соперничество между транзакциями, запрашивающими одни и те же данные.

Эти шаги описывают лишь самый основной уровень операций, которые осуществляет ORACLE.

10. Журнал Повторений

Каждая база данных ORACLE имеет набор из двух или более ФАЙЛОВ ЖУРНАЛА ПОВТОРЕНИЯ РАБОТЫ. Комплект файлов журнала повторения работы для одной базы данных совместно называется ЖУРНАЛОМ ПОВТОРЕНИЯ (redo log).

В базе данных Oracle имеется файлы двух типов:

- Файлы данных, сгруппированные в табличные пространства.
- Файлы данных, относящиеся к классу журнала повтора.

В базе данных как минимум должно быть не менее двух оперативных журналов повтора.

Журнал повтора часто называют журналом транзакций. Основная функция журнала повторения - регистрация всех изменений, осуществляемых в данных. Все изменения, выполняемые в базе данных, записываются в журнал повторения. Если в результате сбоя модифицированные данные не удастся постоянно записать в файлы данных, эти изменения можно получить из журнала повторения, так что результаты работ никогда не теряется, т.е. можно произвести не только восстановление старой копии БД, но и повторить все транзакции, выполненные к моменту сбоя.

Файлы журнала повторения критичны в вопросе защиты базы данных от сбоев. Чтобы защититься от таких сбоев, которые затрагивают сам журнал повторения, ORACLE допускает ЗЕРКАЛЬНЫЙ ЖУРНАЛ ПОВТОРЕНИЯ, так что две или более копий журнала повторения можно поддерживать одновременно на разных дисках.

Информация в файле журнала повторения используется только для восстановления базы данных после сбоя системы или носителя, в результате которого данные базы данных не могут быть записаны в файлы данных.

Процесс применения журнала повторения в процессе операции восстановления базы данных называется ПРОКРУТКОЙ ВПЕРЕД. База данных может работать в двух режимах. В режиме ARCHIVELOG и в режиме NOARCHIVELOG, соответственно либо, создается архивная копия всех журналов транзакций, либо содержимое транзакций не сохраняется. Для определения текущего режима работы экземпляра выдают команду archive log list, из под Server* Manager. Многие аспекты процесса архивирования указаны параметрами файла INIT.ORA Восстановление после сбоя экземпляра, с использование только журнала обновления, называется - оперативным восстановлением.

Чтобы добавить новые члены в существующую группу журнала, используйте либо диалог Add Online Redo Log Member в SQL*DBA, либо команду SQL ALTER DATABASE с параметром ADD LOGFILE MEMBER. Для удаления группы онлайнового журнала повторения вы должны иметь системную привилегию ALTER DATABASE.

Число онлайновых файлов журнала базы данных ограничивается тремя параметрами:

- Параметр MAXLOGFILES, который использовался в предложении CREATE DATABASE при создании базы данных, определяет максимальное число групп онлайнового журнала на базу данных; номера групп могут изменяться от 1 до MAXLOGFILES. Единственный способ перекрыть эту верхнюю границу - заново создать базу данных или ее управляющий файл; поэтому важно рассмотреть это значение перед созданием базы данных. Если в предложении CREATE DATABASE не указан параметр MAXLOGFILES, то ORACLE использует умалчиваемое значение, зависящее от операционной системы.
- Параметр LOG_FILES в файле параметров может временно уменьшить максимальное число групп файлов журнала на время исполнения текущего экземпляра. Значение этого параметра не может превышать значение MAXLOGFILES. Если в файле параметров не указан параметр LOG_FILES, то ORACLE использует умалчиваемое значение, зависящее от операционной системы.
- Параметр MAXLOGMEMBERS, который использовался в предложении CREATE DATABASE при создании базы данных, определяет максимальное число членов в группе. Как и для MAXLOGFILES, единственный способ перекрыть эту верхнюю границу - заново создать базу данных или ее управляющий файл; поэтому важно рассмотреть это значение перед созданием базы данных. Если в предложении CREATE DATABASE не указан параметр MAXLOGMEMBERS, то ORACLE использует умалчиваемое значение, зависящее от операционной системы.

Журнал повтора можно разделить на две категории: оперативные и архивные. Оперативные журналы повтора – это два и более журналов, записываемые во время работы базы данных;При работе в режиме ARCHIVELOG перед началом перезаписи Oracle копирует содержимое очередного оперативного журнала повтора в соответствующее место на диске, где хранятся архивные журналы повтора. В момент переключения оперативных журналов повтора, каждому создаваемому журналу присваивается определенный порядковый номер, в соответствие с коим происходит процесс переключения журналов повтора.

К примеру: Уже после второго переключения журнала повтора копируется журнал повтора из первой группы, после чего созданная копия будет называется архивным журналом повтора.

Номер цикла	Порядковый номер архивного журнала	Группа журнала повтора	Состояние
Подключение №1	18	1	Активна
		2	Не активна
		3	Архивируется
Подключение №2	19	1	Архивируется
		2	Активна
		3	Не активна
Подключение №3	20	1	Не активна
		2	Архивируется
		3	Активна
Подключение №4	21	1	Активна
		2	Не активна
		3	Архивируется

Создание групп онлайнового журнала

Можно создать группы онлайнового журнала и как часть создания базы данных, и позднее. Если возможно, спланируйте журнал и создайте все необходимые группы файлов журнала во время создания базы данных. Для создания новой группы онлайнового журнала используйте команду SQL: ALTER DATABASE с параметром ADD LOGFILE. Более подробно см Резервное копирование и восстановление.

11. Транзакция (Transaction)

Транзакция (модуль фиксации) – логический модуль, который состоит из набора изменений (вставок, обновлений и удалений). Транзакции либо должны быть сохранены в базе данных, либо должен быть выполнен их откат. Фиксируются либо все изменения в транзакции, либо ни одно из них. Для фиксирования в базе данных результатов

транзакции используется команда COMMIT, для восстановления изменённых данных используется команда ROLLBACK.

Транзакция начинается:

1. Когда пользователь подключается к базе данных и начинает с ней работать.
2. После выполнения оператора COMMIT.
3. После выполнения оператора ROLLBACK.

Откат на уровне оператора – данное понятие означает, что для конкретного оператора либо будут сохранены все внесённые им изменения, либо не будет выполнено ни одно из них.

БЛОКИРОВКИ(LATCH) - это механизмы, используемые для того, чтобы предотвратить деструктивные взаимодействия между пользователями, обращающимися к одному и тому же ресурсу, будь то вся таблица или одна строка в таблице.(V$LATCH)

В многопользовательской базе данных могут применяться два уровня блокировок:

1. Монопольная блокировка – запрещает разделение ассоциированного ресурса. Первая транзакция, получившая монопольную блокировку ресурса, становится единственной транзакцией, которая может изменять этот ресурс до снятия монопольной блокировки.
2. Разделяемая блокировка – позволяет совместно использовать ассоциированный ресурс, в зависимости от того, какие операции выполняются (например, несколько пользователей могут читать данные одновременно). Несколько транзакций могут получить разделяемые блокировки для одного и того же ресурса.

Захват – ситуация, которая возникает, когда два или более пользователей ожидают данных, заблокированных друг другом.

ORACLE автоматически распознает ситуации захватов, и автоматически разрешает такие ситуации, осуществляя откат одного из предложений, вовлеченных в захват и тем самым освобождая одно множество конфликтующих блокировок строк.

Многоверсионная модель согласованности данных.

1. ORACLE обеспечивает для всех запросов согласованность по чтению на уровне предложения, которая гарантирует, что данные, возвращаемые запросом, согласованы по отношению к моменту начала этого запроса.
2. ORACLE также предоставляет необязательную возможность согласованности по чтению на уровне транзакции, которая гарантирует, что данные, которые видят все запросы внутри транзакции, согласованы по отношению к одной точке времени (моменту начала транзакции). ORACLE обеспечивает согласованность по чтению на уровне транзакции с помощью двух методов:
   - Транзакции read-only – могут содержать только запросы и не могут содержать никаких предложений DML (языка манипулирования данными). В течение транзакции её доступны данные, которые были подтверждены к моменту её начала.
   - Монопольные блокировки таблиц и строк. Если повторяемость чтений необходимо обеспечить в транзакциях, содержащих предложения DML, транзакция может явно запросить разделяемые блокировки по таблицам или монопольные блокировки по тем строкам, которые будут считываться неоднократно.

Где:
DDL – Data Definition Language – язык содержащий набор операторов для определения данных.
DML – Data Modifying Language – язык содержащий набор операторов для модификации данных.

По умолчанию Oracle автоматически блокирует строки, на которые воздействуют операторы INSERT, UPDATE или DELETE; причём блокируются только те строки, на которые фактически оказано воздействие, а не вся таблица или весь блок данных.

Назначение блокировок данных (блокировок DML) - защитить данные таблицы, гарантируя их целостность при одновременном доступе к данным нескольких пользователей. Блокировки данных предотвращают деструктивное взаимодействие одновременных конфликтующих операций DML и операций DDL. Операции DML могут получать блокировки на двух различных уровнях: для конкретных строк и для целых таблиц.

Блокировки строк (TX)

Монопольная блокировка данных запрашивается для индивидуальной строки от имени транзакции, если эта строка модифицируется одним из следующих предложений: INSERT, UPDATE, DELETE или SELECT с фразой FOR UPDATE. Строка всегда блокируется монопольно, так что другие пользователи не могут модифицировать эту строку до тех пор, пока транзакция, удерживающая блокировку, не будет подтверждена или отменена. Блокировки строк всегда запрашиваются ORACLE автоматически как результат приведенных выше предложений.

Блокировки таблиц (TM)

Транзакция запрашивает блокировку таблицы, когда таблица модифицируется следующими предложениями: INSERT, UPDATE, DELETE, SELECT с фразой FOR UPDATE, либо блокируется предложением LOCK TABLE. Блокировка таблицы может удерживаться в одном из следующих режимов: разделяемая для строк (RS), монопольная для строк (RX), разделяемая для таблицы (S), разделяемая для строк монопольная (SRX) и монопольная(X). Степень ограничения доступа, налагаемая блокировкой, определяет, какие режимы блокировок по этой же таблице могут быть получены другими одновременными транзакциями.

Разделяемые для строк блокировки таблиц (RS)

Эта блокировка указывает, что транзакция, удерживающая блокировку по таблице, блокировала строки в этой таблице и намерена обновить их. Эта блокировка автоматически запрашивается для ТАБЛИЦЫ, модифицируемой следующими предложениями:

SELECT ... FROM таблица ... FOR UPDATE OF ...;

LOCK TABLE таблица IN ROW SHARE MODE;

Блокировка RS - это наименее ограничительный режим блокировки таблицы, предоставляющий наибольшую степень одновременного доступа к таблице.

Блокировка RS, удерживаемая транзакцией, запрещает другим транзакциям получать монопольный доступ по записи к данной таблице, который запрашивается следующим предложением (и только им):

LOCK TABLE таблица IN EXCLUSIVE MODE;

Монопольные для строк блокировки таблиц (RX)

Эта блокировка обычно указывает, что транзакция, удерживающая блокировку по таблице, уже выполнила одно или несколько обновлений в строках таблицы. Эта блокировка автоматически запрашивается для ТАБЛИЦЫ, модифицируемой следующими предложениями:

- INSERT INTO таблица ...;
- UPDATE таблица ...;
- DELETE FROM таблица ...;
- LOCK TABLE таблица IN ROW EXCLUSIVE MODE;

Блокировка RX, удерживаемая транзакцией, запрещает другим транзакциям вручную блокировать таблицу для монопольного чтения или записи; поэтому другие транзакции не могут одновременно использовать следующие предложения:

- LOCK TABLE таблица IN SHARE MODE;
- LOCK TABLE таблица IN SHARE EXCLUSIVE MODE;
- LOCK TABLE таблица IN EXCLUSIVE MODE;

Разделяемые блокировки таблиц (S)

Блокировка S, удерживаемая транзакцией, позволяет другим транзакциям одновременно лишь опрашивать эту таблицу, блокировать выбираемые строки с помощью команды SELECT ... FOR UPDATE, или успешно выполнять предложения LOCK TABLE ... IN SHARE MODE; никакие обновления они не могут осуществлять. Несколько транзакций могут одновременно удерживать блокировки S для одной и той же таблицы.

Разделяемая блокировка таблицы, удерживаемая транзакцией, также запрещает другим транзакциям выполнять следующие предложения:

- LOCK TABLE таблица IN SHARE EXCLUSIVE MODE;
- LOCK TABLE таблица IN EXCLUSIVE MODE;
- LOCK TABLE таблица IN ROW EXCLUSIVE MODE;

Разделяемые для строк монопольные блокировки таблиц (SRX)

Блокировка SRX, удерживаемая транзакцией, позволяет другим транзакциям одновременно лишь опрашивать эту таблицу или блокировать выбираемые строки с помощью команды SELECT ... FOR UPDATE, но не обновлять эту таблицу.

Блокировка SRX, удерживаемая транзакцией, запрещает другим транзакциям получать блокировки SRX по этой таблице и модифицировать эту таблицу. Транзакция не может вставлять, обновлять или удалять строки в таблице, если какая-то другая транзакция имеет блокировку SRX по этой таблице. Блокировка SRX, удерживаемая транзакцией, также запрещает другим транзакциям получать блокировки SRX, S и RX по этой таблице; иными словами, другие транзакции не могут успешно выполнять следующие предложения:

- LOCK TABLE таблица IN SHARE MODE;
- LOCK TABLE таблица IN SHARE EXCLUSIVE MODE;
- LOCK TABLE таблица IN ROW EXCLUSIVE MODE;
- LOCK TABLE таблица IN EXCLUSIVE MODE;

Монопольные блокировки таблиц (X)

Эта блокировка вводит самый ограничительный режим, который обеспечивает транзакции, удерживающей эту блокировку по таблице, возможность монопольной записи в таблицу. Эта блокировка запрашивается для таблицы следующим предложением LOCK TABLE таблица IN EXCLUSIVE MODE;

Все блокировки данных, получаемые транзакцией, включая все блокировки строк и таблиц, освобождаются при подтверждении или откате этой транзакции. Блокировки данных, полученные после точки сохранения, освобождаются, если транзакция откатывается к этой точке сохранения.

Конверсия и эскалация блокировок данных

ORACLE автоматически конвертирует блокировку таблицы из более слабой в необходимую более строгую степень ограничений. Например, предположим, что транзакция использует предложение SELECT с фразой FOR UPDATE, чтобы заблокировать строки в таблице. Как следствие, она получает монопольные блокировки строк и разделяемую для строк блокировку таблицы. Если эта транзакция позднее обновляет одну или несколько заблокированных строк, блокировка таблицы автоматически конвертируется из режима RS в режим RX.

ORACLE никогда не прибегает к эскалации блокировок, когда СУБД автоматически заменяет многочисленные блокировки, полученные на одном уровне другой блокировкой на более высоком уровне.

Замки

Замки (latches) - это простые, низкоуровневые механизмы очередизации, которые защищают структуры разделяемых данных в SGA. Например, замки защищают список пользователей, обращающихся к базе данных в текущий момент, а также структуры данных, описывающие блоки в буферном кэше. Серверный или фоновый процесс получает замок на очень короткое время, пока он манипулирует или просматривает одну из таких структур. Реализация замков зависит от операционной системы, особенно в вопросе о том, ожидает ли процесс замка и сколь долго.

Внутренние блокировки

Внутренние блокировки - это более сложные механизмы, чем замки, и они служат разнообразным целям. Рассмотрим их назначение ниже для трех различных категорий внутренних блокировок:

1. **Блокировки кэша словаря.** Эти блокировки на очень короткое время удерживаются для записей словаря при использовании или модификации этих записей. Они гарантируют, что предложения SQL во время их разбора видят согласованные определения объектов. Блокировки кэша словаря могут быть разделяемыми и монопольными. Разделяемые блокировки освобождаются по окончании синтаксического разбора. Монопольные блокировки освобождаются по концу операции DDL.
2. **Блокировки управления файлами и журналом**. Эти блокировки защищают различные файлы. Например, одна блокировка защищает управляющий файл, чтобы его мог модифицировать лишь один процесс в каждый момент времени. Другая блокировка координирует использование и архивирование файлов журнала повторения. Файлы данных блокируются для того, чтобы гарантировать, что база данных монтируется несколькими экземплярами в разделяемом режиме или одного экзепляра в монопольном режиме. Поскольку блокировки файлов и журнала отражают состояние файлов, эти блокировки по необходимости удерживаются продолжительное время.
3. **Блокировки табличных пространств и сегментов отката**. Эти блокировки защищают табличные пространства и сегменты отката. Например, все экземпляры, имеющие доступ к базе данных, должны согласованно отражать онлайновое или офлайновое состояние табличного пространства. Сегменты отката блокируются для того, чтобы лишь один экземпляр мог писать в сегмент отката.

Явные блокировки данных

Во всех случаях ORACLE автоматически осуществляет блокировки для обеспечения одновременного доступа, целостности и согласованности данных на уровне предложения. Однако имеются возможности, которые позволяют вам перекрывать умалчиваемые механизмы блокировок ORACLE. Такое перекрытие полезно в следующих ситуациях:

- Приложениям требуется согласованность по чтению на уровне транзакций или "повторяемость чтений"; транзакциям необходимо опрашивать множество данных, согласованное на все время выполнения транзакции, с гарантией, что эти данные не будут изменены другими транзакциями в системе.
- Приложениям требуется, чтобы транзакция имела монопольный доступ к ресурсу; такая транзакция может выполняться, не ожидая завершения других транзакций.

12. Обеспечение защиты базы данных

Защита в среде Oracle обеспечивается с помощью средств создания, изменения и уничтожения учётных записей пользователя базы данных Oracle (см. вопрос 15), а также с помощью привилегий (Grant, role), регулирующих возможность создавать и изменять объекты базы данных (см. вопрос 14).

13. Представления словаря 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.

Введение в словарь данных

Словарь данных является одной из важнейших частей базы данных ORACLE. Словарь данных - это набор таблиц, используемых как справочник только для чтения, 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, представлений, 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 аудита, например, кто обращался к различным объектам и обновлял их
- другую общую информацию о базе данных

Словарь данных структурирован через таблицы и представления, 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 представлениям словаря данных.

Структура словаря данных

В состав словаря данных базы данных входят:

Базовые таблицы	Основу словаря данных составляет совокупность 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.
Доступные пользователю представления	Словарь данных содержит доступные пользователю представления, которые суммируют и отображают, в удобном представлении формате информацию из базовых таблиц словаря. Эти представления 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 представлениям вместо базовых таблиц словаря.

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

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, аудите, грантах и данных.

Вы можете добавлять в словарь данных новые таблицы или представления. Если вы добавляете новые объекты в словарь данных, их владельцем должен быть SYSTEM или какой-либо третий пользователь ORACLE. Когда включен режим аудита, эта таблица может неограниченно расти. Хотя пользователи 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.

Представления словаря данных выступают как справочники для всех пользователей базы данных. Доступ к этим представлениям осуществляется через SQL. Некоторые представления 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 представлений. Во многих случаях такой набор состоит из трех представлений, содержащих аналогичную информацию и отличающихся друг от друга своими префиксами:

Префикс	Назначение
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 представление в наборе идентичны, со следующими исключениями:

- В представлениях с префиксом USER обычно нет столбца с именем OWNER (владелец); в представлениях USER под владельцем подразумевается пользователь, выдавший запрос.
- Некоторые представления DBA имеют дополнительные столбцы, которые содержат информацию, полезную для АБД.

Представления с префиксом USER:

- 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 представлениями, с тем исключением, что столбец OWNER подразумевается (текущий пользователь)
- возвращают подмножество информации, предоставляемой представлениями ALL могут иметь сокращенные общие синонимы для удобства

Представления с префиксом ALL отражают общее представление о базе данных со стороны пользователя. Эти представления возвращают информацию об объектах, к которым пользователь имеет доступ через общие или явные гранты привилегий и ролей, помимо тех объектов, которыми владеет этот пользователь. Представления с префиксом DBA показывают общее представление о базе данных, так что они предназначены только для администраторов базы данных. Точнее, опрашивать представления словаря с префиксом DBA может любой пользователь, имеющий системную привилегию SELECT ANY TABLE.

14. Привилегии (Grant, role).

Привилегии системного уровня.

Существует две основные группы привилегий системного уровня: имеющие в составе имён слово ANY и не имеющие его. Привилегии со словом ANY в имени позволяют пользователю
Привилегии системного уровня могут быть даны либо непосредственно пользователю Oracle, либо роли, которая будет назначена пользователю Oracle.

Если нужно будет дать привилегии системного уровня каждому пользователю Oracle (включая любых пользователей Oracle, которые могут быть созданы в будущем), можно дать системную привилегию пользователю PUBLIC, что означает любого пользователя Oracle.

Привилегии системного уровня могут быть также предоставлены пользователю Oracle с использованием WITH ADMIN OPTION в конце оператора GRANT. Это позволяет пользователю Oracle, получившему такую системную привилегию, передавать её любому другому пользователю Oracle. По сути дела он становится ещё одним администратором этой привилегии.

Для изъятия привилегии или роли системного уровня у пользователя Oracle, можно использовать команду REVOKE. Ею можно определить сразу несколько системных привилегий и имён пользователей.

На любые объекты, созданные в то время, когда действовала эта привилегия, её отмена не повлияет. Кроме того, если системная привилегия изъята у пользователя PUBLIC, не будет затронут другой пользователь, которому эта системная привилегия была предоставлена непосредственно.

Любой пользователь предоставивший системную привилегию посредством WITH ADMIN OPTION, может предоставить эту системную привилегию другому пользователю Oracle – он, по сути дела, является дополнительным администратором этой системной привилегии.

Привилегии объектного уровня.

Если пользователь Oracle владеет объектом наподобие таблицы, другим пользователям Oracle может быть разрешено использовать этот объект путём предоставления им одной или нескольких привилегий объектного уровня. Например, если пользователь user_1 намеревается позволить пользователю user_2 использовать его таблицу, но только для вставки и обновления строк с помощью команд INSERT или UPDATE, то пользователю user_2 могут быть даны привилегии UPDATE и INSERT.

- SELECT – позволяет другому пользователю Oracle сделать запрос к данным таблицы.
- INSERT – позволяет другому пользователю Oracle вставлять строки в таблицу с помощью команды INSERT.
- UPDATE – позволяет пользователю Oracle обновлять строки в таблице вне зависимости от того, были ли эти строки созданы этим пользователем или нет. Привилегия UPDATE
- DELETE – привилегия объектного уровня DELETE позволяет удалять из таблицы любые существующие строки. С использованием представления можно ограничить то, какие строки будут удалены.
- EXECUTE – даёт возможность пользователю Oracle, владеющему процедурным кодом базы данных (процедурами, функциями или пакетами), позволить другому пользователю Oracle вызывать его процедурные объекты.

- ALTER – даёт возможность пользователю Oracle изменить определение таблицы или последовательности.
- INDEX – позволяет пользователю Oracle создавать индексы на таблицы владельца.
- REFERENCES – позволяет пользователю Oracle обращаться к объектам другого пользователя.

В конце оператора GRANT для привилегии объектного уровня может быть определена фраза WITH GRANT OPTION, которая позволяет пользователю Oracle получившему эту привилегию, передать её другому пользователю Oracle.

Имеется ряд представлений словаря данных, которые показывают, какие привилегии объектного уровня были даны непосредственно пользователю и какие привилегии были предоставлены ему другими пользователями:

- USER_TAB_PRIVS
- USER_COL_PRIVS
- TABLE_PRIVILEGES
- COLUMN_PRIVILEGES

Представления словаря данных, начинающиеся с ALL_ и DBA_.

Роль(role) – совокупность системных привилегий и привилегий объектного уровня, которые были сгруппированы под одним именем, чтобы облегчить предоставление и отмену этих привилегий. Когда пользователю даётся роль, он получает все привилегии, связанные с ней.

Если пользователь Oracle удалён из базы данных командой DROP USER, можно сохранить присвоенные ему привилегии, если эти привилегии были назначены посредством роли. Затем можно переназначить эти привилегии любому новому пользователю. Если привилегии первоначально не были назначены посредством роли, то при уничтожении первого пользователя будет потеряна вся относящаяся к нему информация.

С помощью ролей можно расширять и сужать набор привилегий, назначенных пользователю, разрешая и запрещая роли.

Новой роли могут быть назначены привилегии уже существующей роли (роль более высокого уровня наследует все привилегии роли более низкого уровня).

Как только пользователю Oracle был дан набор ролей, предусмотренные ими привилегии могут включаться и выключаться (обычно прикладными программами) посредством разрешения и запрещения ролей.

Вместе с новой базой данных автоматически создаются пять ролей. Это – CONNECT, RESOURCE, DBA, EXP_FULL_DATABASE и IMP_FULL_DATABASE. Первые три предусмотрены для совместимости с предыдущими версиями программного обеспечения Oracle и обычно не требуются.

15. Управление пользователями базы данных.

Центральное место в средствах защиты занимает учётная запись пользователя базы данных Oracle.

CREATE USER – это команда SQL, которая может использоваться для определения учётной записи Oracle в базе данных. После создания учётной записи пользователя Oracle она не может использоваться, пока пользователь не получит, по меньшей мере одну системную привилегию. Системная привилегия CREATE SESSION позволяет пользователю создавать сеанс по отношению к базе данных Oracle. Это – необходимая привилегия, которую должна иметь учётная запись пользователя, без неё учётная запись пользователя Oracle не может использоваться.

При первоначальном создании пользователя Oracle можно определить заданное по умолчанию табличное пространство, в котором будут создаваться объекты пользователя. Если заданное по умолчанию табличное пространство не определено, пользователю будет назначено табличное пространство SYSTEM в качестве заданного по умолчанию, которое будут использовать объекты базы данных. В составе оператора CREATE USER может использоваться фраза DEFAULT TABLESPACE для определения того, что объекты пользователя должны быть помещены в табличное пространство, отличное от SYSTEM. Пользователю Oracle также должна быть назначена квота, которая определяет, сколько памяти он может использовать в табличном пространстве.

Другой способ создания пользователя состоит в том, чтобы предоставить пользователю роли CONNECT, RESOURCE и DBA. Хотя это и быстрый метод, он включён, прежде всего, для совместимости с предыдущими версиями программного обеспечения Oracle. Команда CREATE USER – более предпочтительный метод, поскольку в одной команде можно указать и квоту и другие установки.

Можно использовать команду ALTER USER для изменения таких параметров пользователя, как пароль, заданные по умолчанию временные табличные пространства и квота памяти.

Для удаления пользователя из базы данных используется команда DROP USER, которая удаляет запись пользователя из словаря данных Oracle. Если пользователь Oracle владеет какими-либо объектами базы данных, можно либо удалить каждый из объектов перед использованием команды DROP USER, либо использовать в DROP USER опцию CASCADE для автоматического уничтожения всех объектов при удалении учётной записи пользователя.

16. Аудит базы данных

Словарь данных каждой базы данных содержит таблицу с именем SUS.AUD$, обычно называемую АУДИТОРСКИМ ЖУРНАЛОМ базы данных. Аудиторские записи, генерируемые как результат отслеживания предложений, привилегий или объектов, можно помещать как в аудиторскую таблицу базы данных, так и в аудиторский журнал операционной системы. Использование аудиторского журнала дает возможность просматривать выбранные порции аудиторского журнала с помощью предопределенных представлений словаря

данных, а также использовать инструменты ORACLE, такие как SQL*ReportWriter, для генерации аудиторских отчетов.

Использование аудиторского журнала операционной системы помогает консолидировать аудиторские записи из многих источников, включая ORACLE и другие приложения. Поэтому исследование активности системы может быть более эффективным, так как аудиторские записи сосредоточены в одном месте.

Аудиторский журнал базы данных (SYS.AUD$) - это единственная таблица в словаре данных каждой базы данных ORACLE. Если вы решили использовать аудит, создайте аудиторские представления, подключившись как SYS и запустив скрипт CATAUDIT.SQL. Если вы решили отключить аудит и больше не нуждаетесь в аудиторских представлениях, удалите их, подключившись как SYS и запустив скрипт CATNOAUD.SQL. Точное имя и местоположение этого скрипта зависит от операционной системы.

Установка опций аудита

Все опции аудита генерируют следующую общую информацию:

- имя пользователя, выполнявшего отслеживаемое предложение;
- код действия, указывающий выполненное предложение;
- объекты, адресуемые в отслеживаемом предложении;
- дату и время выполнения отслеживаемого предложения;

Аудиторский журнал не сохраняет информации о каких-либо значениях данных, которые могли быть вовлечены в отслеживаемое предложение; например, при аудите предложения UPDATE не сохраняются старые и новые значения данных. Однако, такой специализированный тип аудита можно осуществить для предложений DML, работающих с таблицами, с помощью триггеров базы данных.

ORACLE позволяет устанавливать опции аудита на трех уровнях:

- предложение аудита базируется на типе предложений SQL, например, на любых предложениях SQL по таблицам (что регистрирует каждое предложение CREATE, TRUNCATE и DROP TABLE)
- привилегия отслеживает использование конкретной системной привилегии, такой как CREATE TABLE
- объект отслеживает конкретные типы предложений на конкретных объектах, например, ALTER TABLE по таблице EMP

Групповые обозначения для опций аудита

Для удобства спецификации часто встречающихся групп связанных опций аудита предоставляются специальные обозначения. Эти обозначения сами не являются опциями; они просто позволяют указать одним словом целую группу опций в предложении AUDIT или NOAUDIT.

Включение и выключение аудита базы данных

Любой пользователь базы данных ORACLE может в любой момент установить опции аудита предложений, привилегий или объектов, но ORACLE не генерирует аудиторских записей и не помещает их в аудиторский журнал, если не включен режим аудита базы данных. Обычно за эту операцию отвечает администратор. Аудит базы данных включается и выключается параметром инициализации AUDIT_TRAIL в файле параметров базы данных. Этот параметр может быть установлен в следующие значения:

- DB - включает аудит базы данных и направляет все аудиторские записи в аудиторский журнал базы данных
- OS - включает аудит базы данных и направляет все аудиторские записи в аудиторский журнал операционной системы
- NONE - выключает аудит (умолчание)

Администратор защиты обязан контролировать рост аудиторского журнала и его размер. Когда аудит включен и генерируются аудиторские записи, аудиторский журнал растет за счет двух факторов:

- числа включенных опций аудита
- частоты выполнения отслеживаемых предложений

Для контроля за ростом аудиторского журнала вы можете использовать следующие методы:

- Включать и выключать аудит базы данных. Когда аудит включен, аудиторские записи генерируются и поступают в журнал; когда аудит выключен, аудиторские записи не генерируются.
- Жестко контролировать возможности осуществлять аудит объектов. Это можно делать двумя различными способами:
  - Всеми объектами владеет администратор защиты, привилегия AUDIT ANY никогда не назначается никаким другим пользователям. Все объекты схемы могут принадлежать схеме, соответствующий пользователь которой не имеет привилегии CREATE SESSION.
  - Все объекты содержатся в схемах, которые не соответствуют реальным пользователям базы данных (т.е. привилегия CREATE SESSION не назначена пользователям, одноименным со схемами), и администратор защиты является единственным лицом, имеющим системную привилегию AUDIT ANY.
- Все объекты содержатся в схемах, которые не соответствуют реальным пользователям базы данных (т.е. привилегия CREATE SESSION не назначена пользователям, одноименным со схемами), и администратор защиты является единственным лицом, имеющим системную привилегию AUDIT ANY.

Очистка аудиторских записей из аудиторского журнала

После того, как аудит был включен в течение некоторого времени, администратор защиты может удалить записи из аудиторского журнала, - как для того, чтобы освободить память, так и для облегчения управления этим журналом. Например, чтобы удалить ВСЕ записи из аудиторского журнала, введите следующее предложение:

DELETE FROM sys.aud$;

Если информация аудиторского журнала должна архивироваться для целей накопления истории, администратор защиты может скопировать соответствующие записи в нормальную таблицу базы данных или экспортировать аудиторскую таблицу в файл операционной системы. Удалять записи из аудиторского журнала базы данных может лишь пользователь SYS, т.е. пользователь, имеющий привилегию DELETE ANY TABLE (или пользователь, которому SYS передал привилегию DELETE по таблице SYS.AUD$).

Уменьшение размера аудиторского журнала

Как и в любой таблице базы данных, после удаления записей из аудиторского журнала базы данных все экстенты, которые были распределены этой таблице, будут по-прежнему существовать. Если аудиторский журнал имеет слишком много экстентов, большинство из которых не используются, то можно уменьшить размер этого журнала, выполнив следующие шаги:

1. Если вы хотите сохранить информацию из аудиторского журнала, скопируйте ее в другую таблицу базы данных, или экспортируйте ее с помощью утилиты экспорта.
2. Соединитесь с базой данных как INTERNAL.
3. Выполните усечение таблицы SYS.AUD$ с помощью команды TRUNCATE.
4. Перезагрузите аудиторские записи, сохраненные вами на шаге 1.

Защита аудиторского журнала

Осуществляя отслеживание подозрительной деятельности в базе данных, защищайте целостность записей аудиторского журнала, чтобы гарантировать точность и полноту аудиторской информации. Чтобы защитить аудиторский журнал от несанкционированных удалений, назначайте системную привилегию DELETE ANY TABLE только администраторам защиты. Чтобы отслеживать изменения, выполняемые над самим аудиторским журналом, организуйте аудит аудиторского журнала с помощью следующего предложения:

AUDIT INSERT, UPDATE, DELETE
ON sys.aud$
BY ACCESS;

Аудит с помощью триггеров базы данных

Вы можете использовать триггеры, чтобы дополнить встроенные средства аудита ORACLE. Хотя вы можете писать триггеры, которые регистрировали бы информацию, аналогичную той, которую генерирует команда AUDIT, старайтесь использовать триггеры лишь в том случае, когда вам необходима более детальная аудиторская информация. Например, с помощью триггеров вы можете отслеживать изменения значений по строкам таблицы.

17. Обеспечение целостности базы данных

Целостность данных определением правил проверки достоверности данных гарантирующих, что недействительные данные не попадут в ваши таблицы. Oracle позволяет определять и хранить эти правила для объектов БД, которых они касаются, таким образом, чтобы кодировать их только однажды. При этом они активируются всякий раз, когда какой-либо вид изменение проводится в таблице, независимо от того, какая программа выполняет вставки, модификации или удаления. Этот контроль осуществляется в форме ограничений и триггеров БД. Ограничения – это правила, применимые к таблицам во времена или после создания, распространяемые на то, как эти таблицы могут заполняться.

Ограничение целостности устанавливает правила на уровне БД, определяя набор проверок для таблиц системы. Эти проверки автоматически выполняются всякий раз, когда вызывается оператор вставки, модификации или удаления данных в таблице. Если какие либо ограничения нарушены, операторы отменяются. Поскольку ограничения условности проверяются на уровне БД, они выполняются независимо от того, откуда были инициированы операторы вставки, модификации или удаления. Для таблиц можно задавать следующие типы ограничений целостности:

- NOT NULL
- PRIMARY KEY
- UNIQUE KEY
- FOREIGN KEY (REFERENCES)
- CHECK
- INDEX (ИНДЕКСЫ)
- TRRIGERS и PROCEDURES

NOT NULL. Это ограничение устанавливается для столбца, чтобы указать, что столбец должен иметь значение в каждой строке, т.е. некоторое непустое значение.

PRIMARY KEY (первичный ключ). Ограничение определяет столбец или группу столбцов, которую можно использовать для уникальной идентификации строки. Никакие две строки в таблице не могут иметь одинаковые значения столбцов первичного ключа. Кроме того, столбцы первичного ключа должны всегда содержать значение. Все эти условия гарантируют то, что в нашем распоряжение будет одна и только одна строка, соответствующая критериям связывания. Первичные ключи могут быть или именованные (пользователем) или неименованные (Oracle составляет имя сам). В первичных ключах не могут использоваться столбцы типа: raw, long, long raw.

UNIQUE (уникальный). Ограничение UNIQUE используется для определения того, что значения в столбце не должно повторяться в другой строке этой таблицы, определяет вторичный ключ для таблицы. Это столбец или группа столбцов, которые можно использовать как уникальную идентификацию строки. Никакие две строки не могут иметь одинаковые значения для столбца или столбцов ключа UNIQUE. Столбцы для ограничения UNIQUE не обязательно NOT NULL. Можно сформировать ограничение таблицы, указав, что в таблице не должна повторяться комбинация столбцов. К примеру: можно в начале объявить стандартно emp_id number(5), person_id date а под конце объявить что: unique(emp_id, person_id) – и получиться, что сочетание значений этих полей, должно быть уникальным в каждой строке.

FOREIGN KEY (внешний ключ). FOREIGN KEY, устанавливает отношение целостности между таблицами. Оно требует, чтобы столбец или набор столбцов в одной таблице совпадал с первичным или вторичным ключом другой таблицы. С момента создания внешнего ключа ссылающегося на первичный ключ некой таблицы удаление таблицы будет – запрещено. И обойти это ограничение можно только удалив ограничение. Внешние ключи могут быть именованные или неименованные.

CHECK. Ограничение CHECK определяет логику проверки, которая должна жать результат true (истина) для оператора вставки, модификации или удаления из таблицы. Ограничение CHECK гарантирует, что значение в измененной строке удовлетворяют заданному набору проверок правильности.

ИНДЕКСЫ (INDEX)**.** Ограничения PRIMARY KEY и UNIQUE автоматически создают индексы на столбцах, для которых они определены, если ограничение активизируется при создании. Если индекс уже существует на столбцах, которые составляют ограничение PRIMARY KEY и UNIQUE, то использует именно этот индекс и Oracle не может создать новый.

TRRIGERS(Триггеры) – с программный элемент хранимый в БД выполняемый автоматически, в определенных ситуациях, не имеющий входных или выходных параметров, что в конечном итоги и является причиной невозможности вызвать его явно, непосредственно, его вызывает только сама база данных Oracle. Выходные данные триггера должны быть также применимы к БД, а не возвращены вызывающей программе или отображены на экране. Процедуры и функции расписаны в вопросе № 6.Схемы и объекты схемы

Таким образом, целостность базы данных может быть рассмотрена на трех уровнях:

1. На уровне типа данных (т.е. соответствия типов данных)
2. На уровне ключей (к примеру, соответствие первичных и внешних ключей)
3. На уровне триггеров, процедур и/или функций(к примеру, триггера отвечают только за свои области данных).

18. Создание базы данных. (файлы параметров)

При создании базы данных необходимо подготовить несколько файлов данных операционной системы, которые будут использоваться вместе как единая база данных. База данных создается один раз, независимо от того, сколько файлов данных она имеет, и сколько экземпляров будут обращаться к ней. Процедуру создания базы данных можно также использовать для того, чтобы стереть информацию в существующей базе данных и создать новую базу данных с тем же именем и физической структурой.

Создание базы данных включает следующие операции:

- создание новых файлов данных, или стирание данных, хранившихся в предыдущих файлах данных
- создание структур, требующихся ORACLE для доступа и работы с базой данных (словаря данных)
- создание и инициализация управляющих файлов и файлов журнала повторения для базы данных

База данных создается с помощью предложения, включающего команду SQL CREATE DATABASE; однако, прежде чем выдавать такое предложение, рассмотрите следующие вопросы:

- Спланируйте ваши таблицы и индексы, и оцените, сколько пространства они потребуют.
- Спланируйте проблемы защиты вашей базы данных, включая конфигурацию ее онлайнового и архивного журналов (с учетом занимаемого ими пространства) и стратегию резервного копирования.
- Выберите набор символов базы данных. Вы должны указать набор символов при создании базы данных, и не сможете изменить его до повторного создания базы данных. Все символьные данные в базе данных, включая данные в словаре данных, хранятся в наборе символов базы данных. Если пользователи предполагают обращаться к базе данных, используя другие наборы символов, то набор символов базы данных должен быть надмножеством всех используемых наборов символов.

Необходимые предпосылки

Для создания новой базы данных вы должны иметь:

- привилегии операционной системы, соответствующие полноправному администратору базы данных

- достаточное количество памяти для запуска экземпляра ORACLE
- достаточный объем дискового пространства на компьютере, выполняющем ORACLE

Создание базы данных ORACLE

Для создания каждой новой базы данных ORACLE необходимо последовательно выполнить следующие шаги:

1. Скопировать все существующие базы данных.
2. Создать файлы параметров.
3. Отредактировать новые файлы параметров.
4. Проверить идентификатор экземпляра ORACLE.
5. Запустить SQL*DBA и соединиться с ORACLE как INTERNAL.
6. Запустить экземпляр.
7. Создать базу данных.
8. Скопировать базу данных.

19. Запуск и останов базы данных

Для запуска базы данных или инстанции(экземпляр) используйте либо диалоговое окно Start Up Instance, либо команду STARTUP (после того, как соединитесь с ORACLE как INTERNAL). Вы можете запустить экземпляр и базу данных различными способами:

- запустить экземпляр без монтирования базы данных
- запустить экземпляр и смонтировать базу данных, но оставить ее закрытой
- запустить экземпляр, смонтировать и открыть базу данных в одном из следующих режимов:
  - неограниченном режиме (доступна всем пользователям)
  - ограниченном режиме RESTRICTED (доступна только АБД)

Кроме того, вы можете форсировать запуск экземпляра, либо заставить экземпляр начать немедленно после запуска полное восстановление носителя.

Прежде, чем запускать экземпляр, нужно подключиться как INTERNAL; также может понадобиться указать, для какой базы данных вы запускаете экземпляр, и специфицировать файл параметров.

Вы также должны подключиться как INTERNAL. Это условие обязательно, независимо от того, используете ли вы графический интерфейс SQL*DBA или команды SQL.

Запуск экземпляра без монтирования базы данных

Можно запустить экземпляр без монтирования базы данных; обычно это требуется лишь при создании базы данных. Чтобы сделать это, используйте одну из следующих опций SQL*DBA:

- кнопку Nomount в диалоговом окне Start Up Instance
- команду STARTUP с опцией NOMOUNT

Запуск экземпляра и монтирование базы данных

Вы можете запустить экземпляр с монтированием базы данных, но не открывать базу данных, чтобы выполнить специфические операции сопровождения. Например, база данных должна быть смонтирована, но не открыта, во время выполнения следующих задач:

- переименования файлов данных
- добавления, удаления или переименования файлов журнала
- включения и выключения опций архивирования журнала
- полного восстановления базы данных

Для запуска экземпляра и монтирования базы данных без ее открытия используйте одну из следующих опций SQL*DBA:

- кнопку Mount в диалоговом окне Start Up Instance
- команду STARTUP с опцией MOUNT

При использовании команды STARTUP с параметром NOMOUNT производится только запуск фоновых процессов Oracle и распределение SGA в памяти. В состоянии NOMOUNT базу данных может использовать только DBA. Опция NOMOUNT обычно используется только при создании базы данных.

Запуск экземпляра, монтирование и открытие базы данных

Нормальная работа базы данных означает, что экземпляр запущен, а база данных смонтирована и открыта. Это позволяет всем действительным пользователям соединяться с базой данных и выполнять типичные операции, требующие доступа к данным. Для запуска экземпляра с монтированием базы данных и ее открытием используйте одну из следующих опций SQL*DBA:

- кнопку Open в диалоговом окне Start Up Instance

- команду STARTUP с опцией OPEN

Задание имени базы данных

При запуске экземпляра базы данных специфицируйте имя базы данных, которая будет монтироваться, одним из следующих способов:

- введите имя базы данных в поле Database Name в диалоговом окне Start Up Instance
- укажите имя базы данных в команде STARTUP

Задание файла параметров

При запуске экземпляра базы данных выберите файл параметров для инициализации характеристик экземпляра, одним из следующих способов:

- введите имя файла параметров в поле Parameter File в диалоговом окне Start Up Instance
- укажите полное имя файла параметров в опции PFILE команды STARTUP

Спецификации имен файлов зависят от операционной системы. Если имя файла не указано, ORACLE использует умалчиваемое имя файла.

Форсированный запуск экземпляра

Форсированный запуск, описанный ниже, следует применять ТОЛЬКО в следующих случаях:

- Текущую работающую инстанцию не удается закрыть с помощью опций Normal или Immediate меню Shut Down (или эквивалентных опций предложения SHUTDOWN)
- Экземпляр не удается запустить обычным способом

Если вы попали в такую ситуацию, обычно удается решить проблему путем запуска нового экземпляра в форсированном режиме, используя одну из следующих опций SQL*DBA:

- переключатель Force в диалоговом окне Start Up Instance
- команду STARTUP с опцией FORCE

Эти опции в действительности сначала останавливают текущую инстанцию, а затем запускают новую инстанцию (возможно, с монтированием и открытием базы данных).

Запуск экземпляра с монтированием базы данных и полным восстановлением носителя

Если вы знаете, что необходимо восстановление носителя, то вы можете запустить инстанцию так, чтобы она автоматически начала процесс восстановления, используя одну из следующих опций SQL*DBA:

- переключатель Recover в диалоговом окне Start Up Instance
- команду STARTUP с опцией RECOVER

Запуск в монопольном или параллельном режимах

Если ваш сервер ORACLE позволяет обращаться к одной базе данных из нескольких инстанций, то вы должны выбрать монтирование базы данных в монопольном или параллельном режимах.

Автоматический запуск базы данных при запуске операционной системы

На многих установках одна или несколько инстанций и баз данных ORACLE автоматически запускаются вслед за загрузкой ОС. Процедуры, позволяющие организовать такой режим, специфичны для каждой операционной системы.

Запуск удаленного экземпляра

Если ваш локальный сервер ORACLE является частью распределенной базы данных, то вам может понадобиться запускать удаленную инстанцию и базу данных. Процедуры запуска и останова удаленных инстанций широко варьируются в зависимости от коммуникационного протокола и операционной системы.

Останов базы данных

Чтобы инициировать останов базы данных, используйте либо меню Shut Down, либо команду SHUTDOWN в SQL*DBA.

Останов базы данных в нормальных обстоятельствах

Нормальный останов базы данных протекает следующим образом:

- После получения команды на останов не допускаются новые соединения с базой данных.
- Прежде чем остановить базу данных, ORACLE ждет отсоединения от нее всех текущих соединенных пользователей.
- Очередной запуск базы данных не потребует никаких процедур восстановления инстанции.

Для останова базы данных в нормальных обстоятельствах используйте одну из следующих опций SQL*DBA:

- опцию Normal меню Shut Down
- команду SHUTDOWN с опцией NORMAL

Немедленный останов базы данных

В чрезвычайных обстоятельствах вы можете остановить базу данных немедленно. Используйте этот способ останова лишь в случаях, подобных следующим:

- Скоро произойдет отключение питания.
- База данных или одно из ее приложений работает неверно.
- Немедленный останов базы данных протекает следующим образом:
- Обработка текущих предложений SQL от клиентов немедленно прекращается.
- Все неподтвержденные транзакции откатываются. (Если есть длинные неподтвержденные транзакции, этот способ останова может оказаться достаточно продолжительным, несмотря на свое название.)
- ORACLE не ждет отключения текущих соединенных пользователей; ORACLE неявно откатывает активные транзакции и разрывает все пользовательские соединения.
- Очередной запуск базы данных может потребовать восстановления инстанции (которое ORACLE выполнит автоматически).

Для немедленного останова базы данных используйте одну из следующих опций SQL*DBA:

- опцию Immediate меню Shut Down
- команду SHUTDOWN с опцией IMMEDIATE

Примеры останова базы данных

Эта секция приводит примеры останова базы данных и инстанции через интерфейс меню и команды SQL*DBA. Во всех примерах предполагается, что АБД уже подключен как INTERNAL.

Меню Shut Down останавливает базу данных.

Команда SHUTDOWN эквивалентна меню Shut Down. Например, следующее предложение является командным эквивалентом меню Shut Down.

20. Различные режимы работы базы данных

Запуск однопроцессных и многопроцессных инстанций

В большинстве операционных систем вы можете запускать инстанцию ORACLE либо в однопроцессном, либо в многопроцессном режиме, независимо от того, как ORACLE был инсталлирован или запускался последний раз. Если компьютер, на котором выполняется сервер ORACLE, поддерживает многопроцессность, то инстанции баз данных обычно запускаются в многопроцессном режиме, так что много пользователей могут одновременно обращаться к разделяемой базе данных; однопроцессные же инстанции поддерживают лишь одного пользователя в каждый момент. Однако, в некоторых экспериментальных ситуациях, вы можете найти полезным запустить инстанцию в однопроцессном режиме. Некоторые операционные системы (такие как MS-DOS) не поддерживают многопроцессности или разделяемой памяти; в таких системах однопроцессная инстанция является единственной возможностью.

21. Резервное копирование базы данных

Если над базой данных производят любое из ниже перечисленных структурных изменений, базы данных, непосредственно перед изменениями и после делается соответствующее копирование базы данных:

- Создание или удаление табличного пространства
- Добавление или переименование (перемещение) файла данных в существующем табличном пространстве
- Добавление, переименование(перемещение) или удаление группы или члена онлайнового журнала повторения.
- Если база данных работает в режиме ARCHIVELOG, то до и после структурного изменения базы данных требуется лишь резервное копирование управляющего файла базы данных (с помощью команды ALTER DATABASE с опцией BACKUP CONTROLFILE). Можно скопировать и другие части базы данных.
- Если база данных работает в режиме NOARCHIVELOG, то непосредственно перед и после изменения базы данных требуется сделать полное копирование файла базы данных, включая все файлы данных, файлы журнала повторения и управляющие файлы.

Существует, по большому счету, два вида резервного копирования :

1. Непротиворечивое (холодное) резервное копирование, ситуация когда, копии создаются, в случае закрытой БД (close) для пользователей. Копия базы данных, созданной в автономном режиме, содержит: все файлы данных, журналы повторов и управляющие файлы. После останова БД, все файлы базы данной по средствам ОС копируются на один из backup дисков.

Этапы:

- Остановка экземпляра БД Oracle – в режиме shutdown normal (игнорирование, новых подключений и ожидание отключение все зарегистрированных пользователей) или shutdown immediate (немедленное прерывание всех соединений, выполнение операции отката на всех транзакциях ожидающих обработки)
- Копирование всех физических файлов, относящихся к базе данных, управляющие файлы, файлы журнала обновления и файлы базы данных.
- Закончить работу, перезагрузить базу данных

2. Резервное(горячее) копирование в оперативном режиме, к примеру, когда БД работает в архивном режиме ARCHIVELOG, БД все время находиться в оперативном режиме таким образом доступна пользователям.

Этапы:

- Перевод табличного пространства в режим резервного копирования.
- Копирование всех файлов базы данных, связанных с табличным пространством.
- Выведение табличное пространство из режима резервного копирования.
- Повторение действий с первого по третье, пока не будет выполнено резервное копирование всех табличных пространств.
- Копирование управляющего файла.
- Копирование оперативного журнала обновления.

Сопоставление режима ARCHIVELOG и режима NOARCHIVELOG

В режиме ARCHIVELOG:
- Требуется дополнительное дисковое пространство
- Управление архивными журналами влечет за собой дополнительные административные непроизводительные затраты.
- Применимо горячее резервное копирование.
- В случае отказа носителя может быть выполнено полное восстановление базы данных.

В режиме NOARCHIVELOG:
- Не требуется дополнительное дисковое пространство или непроизводительные затраты.
- Может использоваться только холодное резервное копирование.
- В случае отказа теряется вся работа, выполненная со времени последнего резервного копирования.

Включение и выключение архивирования

Вы устанавливаете первоначальный режим архивирования базы данных во время ее создания. В большинстве случаев, во время создания базы данных вы можете выбрать режим по умолчанию NOARCHIVELOG, потому что нет необходимости архивировать информацию, генерируемую за этот период. После того как база данных создана, решите, нужно ли изменить первоначальный режим архивирования. После того, как база данных создана, всегда можно переключать режим архивирования базы данных. Однако, как правило, следует выбрать постоянный режим работы базы данных.

Включение автоматического архивирования

Чтобы автоматическое архивирование заполненных групп было включено установите в TRUE значение параметра LOG_ARCHIVE_START в файле параметров базы данных INIT.ORA:
LOG_ARCHIVE_START = TRUE Это значение будет иметь эффект при очередном запуске базы данных.

Выключение автоматического архивирования

Вы можете выключить автоматическое архивирование журнала в любой момент. Однако, выключив автоматическое архивирование, вы должны вручную, периодически и своевременно, архивировать заполняемые группы журнала. Если база данных работает в режиме ARCHIVELOG, автоматическое архивирование выключено, а группы журнала заполняются, но не архивируются, то процесс LGWR не сможет повторно использовать неактивные группы журнала, чтобы продолжать запись информации повторения. Поэтому работа базы данных будет временно приостановлена до тех пор, пока не будет выполнено необходимое архивирование. Автоматическое архивирование может быть выключено как до, так и после запуска инстанции.

Если база данных работает в режиме ARCHIVELOG, то можно копировать индивидуальное табличное пространство или даже индивидуальный файл. Эта возможность полезна, если одна часть базы данных используется более интенсивно, чем другие, - например, табличное пространство SYSTEM или табличные пространства, содержащие сегменты отката. Если сбой диска повреждает один из таких файлов данных, для его реставрации может быть использована ранняя копия, и меньшее число изменений необходимо применить при прокрутке вперед, чтобы восстановить файл к состоянию на момент сбоя, т.е. время, затрачиваемое на восстановление, сокращается.

В принципе, существуют более упрощенные методики резервного копирования, с использованием программных средств администрирования баз данных - копирования с помощью Oracle Enterprise Manager(OEM), но эти методики уступают по критерию надежности SQL*Plus'у.

Для более подробного ознакомления с процессами восстановления и копирования следует дополнительно ознакомиться с вопросом №10 Журнал повторения(redo log)

22. Динамический SQL

Динамический SQL в Oracle

Описанный в стандарте SQL набор операторов SQL предназначен для встраивания в программу на обычном языке программирования. Поэтому в этом наборе перемешаны операторы "истинного " реляционного языка запросов (например оператор удаления из таблицы части строк, удовлетворяющих заданному условию) и операторы работы с курсорами, позволяющими обеспечить построчный доступ к таблице-результату запроса.

В диалоговом режиме набор операторов SQL и их синтаксис должен быть несколько другим. Весь вопрос состоит в том, как реализовывать такую диалоговую программу. Правила встраивания стандартного SQL в программу на обычном языке программирования предусматривают, что вся информация, касающаяся операторов SQL, известна (за исключением значений переменных, используемых в качестве констант в операторах SQL). Не предусмотрены стандартные средства компиляции с последующим выполнением операторов, которые становятся известными только во время выполнения. Поэтому, опираясь только на стандарт, невозможно реализовать диалоговый монитор взаимодействия с БД на языке SQL или другую прикладную программу, в которой текст операторов SQL возникает во время выполнения, т. е. фактически так или иначе стандарт необходимо расширять.

Один из возможных путей расширения состоит в использовании специальной группы операторов, обеспечивающих динамическую компиляцию (во время выполнения прикладной программы) базового подмножества операторов SQL и поддерживающих их корректное выполнение. В дополнительный набор операторов, поддерживающих динамическую компиляцию базовых операторов SQL, входят операторы: PREPARE, DESCRIBE и EXECUTE.

Оператор подготовки

Во время выполнения оператора PREPARE символьная строка, содержащаяся в host-string-variable, передается компилятору SQL, который обрабатывает таким же образом, как если бы получил в статике. Построенный при выполнении оператора PREPARE код остается действующим до конца транзакции или до повторного выполнения данного оператора PREPARE в пределах этой же транзакции.

Оператор получения описания подготовленного оператора

Оператор DESCRIBE предназначен для того, чтобы определить тип ранее подготовленного оператора, узнать количество и типы формальных параметров(если они есть) и количество и типы столбцов результирующей таблицы, если подготовленный оператор является оператором выборки (SELECT). Действие оператора DESCRIBE состоит в том, что в указанную область памяти прикладной программы (структура этой области фиксирована и известна пользователям)помещается информация, характеризующая ранее подготовленный оператор с заданным именем.

Оператор выполнения подготовленного оператора

Оператор EXECUTE служит для выполнения ранее подготовленного оператора SQL типа "N" (не требующего применения курсора) или для совмещенной подготовки и выполнения такого оператора.

Для выполнения подготовленного оператора служит первый вариант оператора EXECUTE. Число и типы фактических параметров должны соответствовать числу и типам формальных параметров подготовленного оператора.

Работа с динамическими операторами SQL через курсоры

Для использования таких операторов используется расширение механизма курсоров стандарта SQL. Во-первых, при определении курсора можно указывать не только литеральную спецификацию курсора, но и имя оператора, вводимое с помощью оператора PREPARE (в этом случае оператор PREPARE должен текстуально находиться выше оператора DECLARE). Тем самым полный синтаксис оператора DECLARE становится следующим:

```
<declare cursor> ::=
DECLARE <cursor name> CURSOR
FOR { <cursor specification> | <statement-name> }
```

Далее, поскольку для такого курсора в статике неизвестна информация о входных и выходных переменных включающей программы, то используется другая форма операторов OPEN и FETCH.

Полный синтаксис этих операторов становится следующим:

```
<open statement> ::=
OPEN <cursor name>
[USING { <host-vars-list> | DESCRIPTOR <descr-name> }]
<fetch statement> ::=
```

```
FETCH <cursor name>
{ INTO <fetch target list> |
USING <host-vars-list> |
USING DESCRIPTOR <descr-name> }
```

Как видно, предлагается два способа задания фактических входных и выходных параметров: прямое с указанием в операторах OPEN и/или FETCH списков имен переменных включающей программы и косвенное, когда число параметров и их адреса сообщаются через дополнительную структуру-дескриптор.

Первый способ предлагается использовать для работы с операторами выборки, для которых фиксирован набор формальных входных и выходных параметров.

Второй способ работы с динамически откомпилированными операторами, требующими использования курсоров, состоит в использовании дескрипторов динамически формируемых списков параметров.

23. Объектно-ориентированные Базы Данных.

Направление объектно-ориентированных баз данных (ООБД) возникло сравнительно давно. Публикации появлялись уже в середине 1980-х. Однако наиболее активно это направление развивается в последние годы. С каждым годом увеличивается число публикаций и реализованных коммерческих и экспериментальных систем.

Общие понятия объектно-ориентированного подхода и их преломление в ООБД

В наиболее общей и классической постановке объектно-ориентированный подход базируется на концепциях:

- объекта и идентификатора объекта;
- атрибутов и методов;
- классов;
- иерархии и наследования классов.

Объектный тип – это расширение типа, определяемого пользователем, позволяющее инкапсулировать методы с элементами данных в едином логическом модуле. Определение объектного типа служит в качестве шаблона, но не распределяет память. Объекты хранятся физически как строки или столбцы таблицы. Также в Oracle введены объекты трёх новых встроенных типов: объекты типа vary, типа ref, и ‘большие объекты’ типа LOB.

- Объекты типа vary представляют собой один из примеров коллекции Oracle8, которая примерно соответствует массиву. Объекты этого типа полезны при обработке небольших коллекций в коде PL/SQL.
- Объект типа ref представляет собой указатель на объектный тип, определяемый пользователем.
- Объект типа LOB предназначен для работы с ‘большими’ объектами, такими как полные документы, видеоизображения, цифровые звукозаписи и т.д.

Любая сущность реального мира в объектно-ориентированных языках и системах моделируется в виде объекта. Любой объект при своем создании получает генерируемый системой уникальный идентификатор, который связан с объектом во все время его существования и не меняется при изменении состояния объекта.

Каждый объект имеет состояние и поведение. Состояние объекта - набор значений его атрибутов. Поведение объекта - набор методов (программный код), оперирующих над состоянием объекта. Значение атрибута объекта - это тоже некоторый объект или множество объектов. Состояние и поведение объекта инкапсулированы в объекте; взаимодействие между объектами производится на основе передачи сообщений и выполнении соответствующих методов.

Множество объектов с одним и тем же набором атрибутов и методов образует класс объектов. Объект должен принадлежать только одному классу (если не учитывать возможности наследования, см. следующий абзац). Допускается наличие примитивных предопределенных классов, объекты - экземпляры которых не имеют атрибутов: целые, строки и т.д. Класс, объекты которого могут служить значениями атрибута объектов другого класса, называется доменом этого атрибута.

Допускается порождение нового класса на основе уже существующего класса - наследование. В этом случае новый класс, называемый подклассом существующего класса (суперкласса) наследует все атрибуты и методы суперкласса. В подклассе, кроме того, могут быть определены дополнительные атрибуты и методы. Различаются случаи простого и множественного наследования. В первом случае подкласс может определяться только на основе одного суперкласса, во втором случае суперклассов может быть несколько. При поддержании множественного наследования классы связаны в ориентированный граф с корнем, называемый решеткой классов. Объект подкласса считается принадлежащим любому суперклассу этого класса.

Одной из более поздних идей объектно-ориентированного подхода является идея возможного переопределения атрибутов и методов суперкласса в подклассе (перегрузки методов). Эта возможность увеличивает гибкость, но порождает дополнительную проблему: при компиляции объектно-ориентированной программы могут быть неизвестны структура и программный код методов объекта, хотя его класс (в общем случае - суперкласс) известен. Для разрешения этой проблемы применяется так называемый метод позднего связывания, означающий, по сути дела, интерпретационный режим выполнения программы с распознаванием деталей реализации объекта во время выполнения посылки сообщения к нему.

Объектно-ориентированные модели данных

Метод - программный код, привязанный к конкретному классу и применимый к объектам этого класса. Определение метода в ООБД производится в два этапа. Сначала объявляется сигнатура метода, т.е. его имя, класс, типы или классы аргументов и тип или класс результата. Методы могут быть публичными (доступными из объектов других классов) или приватными (доступными только внутри данного класса). На втором этапе определяется реализация класса на одном из языков программирования ООБД.

В модели ООБД поддерживается множественное наследование классов на основе отношения супертип/подтип. В подклассе допускается добавление и/или переопределение атрибутов и методов. Возможные при множественном наследовании двусмысленности (по именованию атрибутов и методов) разрешаются либо путем переименования, либо путем явного указания источника наследования. Объект подкласса является объектом каждого суперкласса, на основе которого порожден данный подкласс.

Специфической особенностью модели ООБД является возможность объявления дополнительных "исключительных" атрибутов и методов для именованных объектов. Это означает, что конкретный именованный объект-представитель класса может обладать типом, являющимся подтипом типа класса. Конечно, с такими атрибутами не работают стандартные методы класса, но специально для именованного объекта могут быть определены дополнительные (или переопределены стандартные) методы, для которых дополнительные атрибуты уже доступны. Подчеркивается, что дополнительные атрибуты и методы привязываются не к конкретному объекту, а к имени, за которым в разные моменты времени могут стоять вообще говоря разные объекты.