ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П.КОРОЛЕВА»

ORACLE – СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Утверждено Редакционно-издательским советом университета в качестве методических указаний

САМАРА
Издательство СГАУ
2006

УДК 004.4(075)
ББК 32.97
Л 69

Инновационная образовательная программа «Развитие центра компетенции и подготовка специалистов мирового уровня в области аэрокосмических и геоинформационных технологий»

Составители: ***Логанова Л.В., Колчин Ю.В.***

Рецензенты: канд. техн. наук, доц. Л. А. Ж а р и н о в а

Л 69 **ORACLE – СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ**: метод. указания / *Л.В. Логанова, Ю.В. Колчин.* – Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2006. – 80 с.

Данные учебно-методические указания предназначены для студентов направления 010500 «Прикладная математика и информатика» и специальности 010501 «Прикладная математика и информатика». Лабораторные работы 1-8 содержат краткие теоретические сведения и задания для выполнения студентами лабораторных работ по курсу «Базы данных и экспертные системы», работы 9-12 – по курсу «Распределенные базы данных». Методические указания могут быть использованы для выполнения курсовых проектов по вышеперечисленным курсам.

УДК 004.4(075)
ББК 32.97

Лабораторная работа №1

Тема «Создание и управление базами данных»

Базы данных Oracle9i можно создавать с помощью DCA (Database configuration Assistant) или же вручную, используя такие инструментальные средства SQL, как SQL *Plus. Для реальной базы данных, прежде чем использовать ее, необходимо дополнительно создать пользователей, дополнительные табличные области и др. Можно написать специальные командные файлы, которые помогают при создании базы данных. В этой лабораторной работе рассматривается создание базы данных с помощью DCA, т.к. для создания и настройки параметров инициализации вручную требуется определенный опыт работы с базами данных Oracle.

DCA позволяет создавать и удалять базы данных. С его помощью можно создавать новые шаблоны, в том числе и на основе имеющихся, а впоследствии использовать их для создания базы данных.

В состав DCA входит мастер, который направляет ваши действия, позволяя осуществить вышеперечисленные возможности.

При создании базы данных Вам необходимо выбрать шаблон базы данных, определить глобальное имя базы данных, причем оно должно быть уникальным сетевым. Стандартным расширением глобальной базы данных является .word или же вместо него вы можете ввести конкретное имя домена. На следующем шаге возможно зарегистрировать БД в службе каталогов, если сервер каталога уже сконфигурирован. Вариант “Не регистрировать базу данных” (No, don’t register the database) является значением по умолчанию. Если БД должна поддерживать небольшое число клиентов или клиенты будут сохранять соединение с ней в течение длительного времени, выбором может быть Dedicated Server Mode (режим выделенного сервера). Иначе, можно выбрать Shared Server Mode (Режим разделяемого сервера), что позволит клиентам

совместное использование пула ресурсов. В процессе создания базы данных можно установить значения параметров инициализации, тем самым фактически определить ее характеристики, задать положения и имена ее файлов, файлов журналов и управляющих файлов. Как уже упоминалось выше, вновь созданные и стандартные шаблоны ORACLE позволяют сэкономить время, необходимое для создания базы данных. Параметры базы данных можно быстро изменить. Шаблоны можно использовать и на других машинах, входящих в сеть, т.к. они хранятся в виде XML-файлов.

После создания базы данных можно создавать следующие внутренние структуры для поддержки приложений: таблицы, разделы и подразделы, пользователей и схемы, индексы, кластеры хеш-кластеры, представления, последовательности, процедуры, функции, модули, триггеры, синонимы, привилегии и роли, связи баз данных, сегменты, экстенты и блоки, сегменты отката, моментальные снимки и материализованные представления.

Консоль OEM (ORACLE ENTERPRISE MANAGER) имеет графический интерфейс пользователя, который позволяет легко перемещаться и управлять всеми объектами БД. Выбрав в навигаторе БД и раскрыв папку Instance, можно просмотреть текущую конфигурацию экземпляра. В том числе возможны просмотр и изменение параметров инициализации после выбора соответствующей кнопки.

Задание на лабораторную работу

1. Создайте базу данных BASENNN (NNN- первые буквы ваших фамилии, имени, отчества).

2. Определите в качестве пользователей SYS с паролем SYS, и SYSTEM с паролем SYSTEM (во время создания).
3. Запустите консоль OEM и соединитесь с вашей БД, указав соответствующего пользователя и пароль.
4. Отключите базу данных и запустите ее вновь.
5. Создайте табличное пространство.

Лабораторная работа №2

Тема «Создание и заполнение таблиц»

Таблицы представляют собой механизм сохранения информации в базе данных Oracle. Структура таблицы включает перечень имен полей (столбцов) таблицы. Столбец характеризуется типом данных и длиной. Для столбцов типа NUMBER можно задать дополнительные характеристики точности и масштаба. Точность определяет число значащих цифр. Масштаб показывает место десятичной точки. Основные типы полей приведены ниже:

CHAR	Символьное поле фиксированной длины до 2000 байт;
NCHAR	Поле фиксированной длины для набора символов, состоящих из нескольких байт.

	Максимальный размер - 2000 символов или 2000 байт в зависимости от набора символов;
VARCHAR2	Символьное поле переменной длины до 4000 символов;
NVARCHAR2	Поле переменной длины для набора символов, состоящих из нескольких байт. Максимальных размер - 4000 символов или 4000 байт в зависимости от набора символов;
DATE	7-байтовое поле фиксированной длины, используемое для хранения дат;
NUMBER	Числовой столбец переменной длины;
LONG	Поле переменной длины, до 2 Гбайт;
RAW	Поле переменной длины (до 2 000 байт), используемое для хранения двоичных данных;
LONG RAW	Поле переменной длины (до 2 Гбайт), используемое для хранения двоичных данных;
BLOB	Двоичный большой объект длиной до 4 Гбайт;
CLOB	Символьный большой объект длиной до 4 Гбайт;
NCLOB	Тип данных CLOB для набора символов, состоящих из нескольких байт; длина до 4 Гбайт;
BFILE	Внешний двоичный файл; размер ограничивается операционной системой;
ROWID	Двоичные данные, представляющие идентификатор RowID. Все RowID занимают 6 байт для нормальных индексов в таблицах, не разбитых на разделы; локальных индексов в таблицах, разбитых на разделы; и указателей строк, используемых для строк с указателями или с перегруппировкой. RowID занимает 10 байт только для глобальных индексов в таблицах,

разбитых на разделы;

UROWID Двоичные данные, используемые для адресации данных; длина до 4000 байт. Могут поддерживать как логические, так и физические значения RowID, а также внешние таблицы, доступ к которым осуществляется через шлюз.

Таблицы, принадлежащие SYS, называются таблицами словаря данных. Они содержат системный каталог, с помощью которого система управляет своей работой. Таблицы можно создавать с помощью OEM или вручную, пользуясь средствами SQL *Plus. Их можно связывать друг с другом. База данных реализует эти отношения с помощью ограничений ссылочной целостности. На столбцы таблицы можно налагать ограничения; при этом каждая ее строка должна удовлетворять указанному в описании ограничению.

Создание нового отношения (таблицы) выполняется с помощью команды DDL CREATE TABLE. Упрощенный синтаксис этой команды выглядит следующим образом:

CREATE TABLE <имя таблицы>

({<имя поля> <тип данных> [(<размер>)]

[<ограничения целостности поля> …]},..

[,<ограничения целостности таблицы>,…]);

Для обязательных полей устанавливается ограничение not null.

- <имя поля> - имя поля (столбца) таблицы;
- <тип данных> - один из выше перечисленных типов;
- <размер> - размер поля в символах;
- <ограничения целостности поля> - предполагает использование следующих ограничений:
  - ✓ primary key – первичный ключ (обязательный и уникальный);

- ✓ unique – уникальное значение поля в пределах таблицы;
- ✓ [not] null – [не] возможность не указывать значение поля;
- ✓ check(<условие>) – проверка условия для поля (полей);
- ✓ default<выражение> - задание значения поля по умолчанию;
- ✓ references<имя таблицы>[<имя столбца>] внешний ключ;

- <ограничения целостности таблицы> - те же ограничения, что и для поля и дополнительно::
  - ✓ foreign key [<список полей>, …] references <имя таблицы> [(<список полей>)] внешний ключ

Например,

Create table tab (id numeric(6) primary key, class numeric(3), fdata date, group char(6), foreign key (class, fdata) references exam(class, fdata));

К командам манипулирования данными (DML) относятся операторы добавления, удаления и изменения кортежа (записи).

INSERT INTO <имя таблицы> [(<имя поля>), …] VALUES (<список выражений>)| <запрос>;

Под <запросом> подразумевается команда Select, результаты работы которой добавляются в таблицу (данный оператор будет рассмотрен в следующей лабораторной работе). В предложении VALUES указываются значения или выражения, которые принимают атрибуты таблицы. Если в списке значений отсутствует

хотя бы одно обязательное поле или нарушаются другие ограничения целостности, то данная команда отвергается.

INSERT INTO tab (id, class, group) VALUES (1,2,'good')

DELETE FROM <имя таблицы> [WHERE <условие>];

В указанной таблице удаляются записи, удовлетворяющие условию отбора.

UPDATE <имя таблицы> SET {<имя поля>=<выражение>},…[WHERE <условие>];

В указанной таблице изменяются значения перечисленных полей, которые удовлетворяют условию отбора.

UPDATE tab SET group='exl' WHERE id=1;

Задание на лабораторную работу

1. Создайте таблицу с именем Stud, в которой хранятся сведения о студентах.:

номер зачетной книжки	Num,
фамилия, имя, отчество	Fname,
год поступления в университет	Year,
дата рождения	Bday,

2. вид обучения (платное или бесплатное, по умолчанию бесплатное) Plata,
   средний балл при поступлении Mb,

стоимость обучения в платной группе	Money,
адрес	Adr

3. Выберите в качестве первичного ключа столбец Num.
4. Задайте ограничения NUM>1000.
5. Заполните 4-6 записями созданную таблицу.
6. Удалите 1 запись таблицы.
7. Обновите поле адрес у студента, с заданной фамилией.
8. Создайте таблицу Session, хранящую результаты сессий, включающую поля: номер зачетной книжки, номер сессии и оценки по трем предметам.
9. В качестве внешнего ключа выберите поле Num.
10. Заполните 4-5 записями созданную таблицу.

Лабораторная работа №3

Тема «Создание запросов и использование их результатов»

Для извлечения данных, содержащихся в таблицах БД, используется оператор SELECT, имеющий в общем случае сложный и многовариантный синтаксис. В данной лабораторной работе рассматриваются наиболее часто используемые примеры конструкций оператора SELECT. Упрощенно оператор SELECT выглядит следующим образом:

SELECT [ALL | DISTINCT] <*список выбора*>, ...
  FROM *имя_табл* [*син_табл*], ...

[WHERE <*предикат- условие выборки или соединения*>]
[GROUP BY <список полей>]
[ORDER BY < [ASC|DESC], ...]
[HAVING <*предикат- условие выборки*>];

Расширение возможностей команды SELECT достигается за счет применения различных операторов, предикатов, функций.

Операторы:

- сравнения: =, >, <, >=, <=;
- логические: AND, OR, NOT

Предикаты, используемые в запросах:

- IN - определяет множество значений, с которыми будет сравниваться значение указанного поля *field.* Предикат считается истинным, если значение поля *field* равно хотя бы одному из элементов множества.

field IN (*список значений)*

- BETWEEN - : определяет, входит ли значение поля *field* в указанные границы. Если лежит вне границ, то предикат возвращает "ложь".

field BETWEEN *значение1* AND *значение2*

- LIKE - используется для поиска подстрок, применяется только в полям типа CHAR, VARCHAR и т.д. Возможно использование шаблонов '_' - один любой символ и '%' - произвольное количество символов (в т.ч. ни одного);

field LIKE *'образец'*

- IS [NOT] NULL - : определяет, установлено ли значение поля.

field IS [NOT] NULL

Функции агрегирования

- COUNT - определяет в результате количество строк (записей) или значений поля, не являющихся NULL-значениями.
- SUM - определяет арифметическую сумму значений указанного числового поля в результирующем множестве записей.
- AVG - определяет среднее арифметическое значений указанного числового поля в результирующем множестве записей;
- MAX, MIN - определяет максимальное (минимальное) значение указанного поля в результирующем множестве.

Задание на лабораторную работу

Создать запросы к базе данных «Результаты сессии студентов».

1. Вывести сведения о студентах, год поступления которых в институт равен заданному.
2. Вывести фамилии студентов и размер оплаты за обучение для студентов, обучающихся платно.
3. Вывести сведения о студентах, обучающихся платно и средний балл при поступлении которых меньше заданного.
4. Вывести фамилии студентов в алфавитном порядке, заданной даты рождения.

5. Вывести год и количество студентов, поступивших в каждом году.
6. Вывести фамилии студентов, сдавших сессию указанного номера с оценками не ниже 4 по всем предметам.
7. Создать запрос с информацией о студентах, имеющих хотя бы одну неудовлетворительную оценку.

Лабораторная работа №4

Тема «Создание и использование представлений SQL»

Для избирательного доступа к данным БД используется достаточно мощное средство – представление SQL, которое объединяет гибкость запроса с возможностью модификации данных в представлении. Представление позволяет выбрать данные из одной или нескольких таблиц, причем в качестве исходных таблиц могут быть выбраны как обычные таблицы, так и другие представления. В отличие от таблицы, представление так же, как и хранимый оператор SQL, не содержит данных. Во время исполнения запроса, который обращается к представлению, из словаря данных извлекается и выполняется хранимый оператор SQL. Отобранные данные представляются в виде таблицы. Как и при работе с таблицей, становится возможным добавление, изменение, удаление и выбор данных из представления. Представление позволяет в одних

случаях ограничить доступ к информации из той или иной таблицы, в других скрывает сложность организации данных.

Оператор определения представления имеет следующий вид:

CREATE VIEW <ИМЯ ПРЕДСТАВЛЕНИЯ> [(<СПИСОК СТОЛБЦОВ>)]

AS <SQL-ЗАПРОС>

Если не указывать имена столбцов в <список столбцов>, то они получат имена, соответствующие именам столбцов, перечисленных в списке выбора запроса. Имена столбцов представления следует указывать, если список выбора содержит агрегирующие функции или столбцы с одинаковыми именами из разных таблиц.

Материализованные представления

В отличие от обычного представления, материализованное сохраняет строки данных, полученные в результате выполнения SQL – запроса. При наличии изменений в исходной таблице БД сохраняет запись о каждом изменении. Можно установить автоматическую синхронизацию представления с исходными таблицами через интервалы времени, определенные пользователем. Материализованные представления могут сохраняться в той же базе данных, что и исходные таблицы, или в другой, удаленной БД. Материализованные представления могут использоваться в хранилищах данных. Часто они применяются для предварительного вычисления и хранения агрегированных данных. Следует отметить, что сохранение предварительно вычисленной и обобщенной информации позволяет значительно повысить скорость выполнения запросов в больших хранилищах данных. Перед созданием материализованных представлений необходимо создать журналы материализованных представлений для всех исходных таблиц. Материализованные представления, сохраняемые в той же базе данных, что и исходные таблицы, позволяет оптимизатору запросов

использовать повторное построение запроса, что опять же способно повысить производительность БД.

Задание на лабораторную работу

Создать представления для базы данных «Результаты сессии студентов».

1. Фамилии студентов, поступивших в институт в указанном году и обучающихся в платной группе.
2. Сведения о студентах и добавить нового студента.
3. Фамилии студентов, сдавших сессию указанного номера на «отлично».
4. Для студентов из п.3, обучающихся платно, изменить (уменьшить) сумму оплаты.

Лабораторная работа №5

Тема «Интерфейс ODBC, обеспечение доступа к данным»

Интерфейс ODBC (Open DataBase Connectivity – открытый интерфейс связанности базы данных) является стандартным интерфейсом программирования, который позволяет приложениям получать доступ к разнообразным системам управления базами данных, расположенных на самых различных платформах. Интерфейс ODBC обеспечивает большую степень независимости от

базы данных посредством стандартного синтаксиса SQL, который может быть переведен специальными драйверами баз данных на собственный язык SQL DBMS.

Доступ к любой версии СУБД Oracle осуществляется следующим образом. Сначала нужно запустить сервер и настроить клиентскую часть Oracle. Создать описание псевдонима базы данных Oracle. При создании этого описания важны три параметра.

Первый из них - сетевой протокол, с помощью которого осуществляется доступ к серверу Oracle (IPX/SPX, TCP/IP и др.). Второй параметр - местоположение сервера в сети. В случае Personal Oracle это обычно компьютер с IP-адресом 127.0.0.1 (это специальный адрес для доступа к локальному компьютеру, так называемый TCP Loopback Address, который обычно имеет URL http://localhost/). Третий параметр - имя базы данных. В общем случае имя может быть любым, но это должно быть имя уже существующей базы данных, с которой вы собираетесь работать. В принципе все описания псевдонимов Oracle хранятся в текстовом файле TNSNAMES.ORA, который можно редактировать вручную. Далее следует проверить соединение клиента с сервером. Обычно в качестве имени пользователя используется имя SYSTEM и пароль MANAGER (если вы сами администрируете сервер). Если же сервер был установлен раньше, узнайте у администратора базы данных, каким именем и паролем следует воспользоваться. При удачном соединении появится соответствующее сообщение. Если соединение с сервером было неудачным, стоит проверить, поддерживается ли указанный сетевой протокол, виден ли в сети компьютер, на котором установлен сервер, и, если нужно, внести изменения в описание сервиса.

Независимость от базы данных и простота использования относятся к основным преимуществам данного интерфейса. Он поддерживается многими популярными инструментальными средствами разработки.

Чтобы получить доступ к базам данных ORACLE с помощью любого инструментального средства (Builder C++, Delphy и т.д.) или с помощью другой СУБД (например MS Acces), необходимо предварительно зарегистрировать целевую базу данных как источник данных ODBC.

Задание на лабораторную работу

1. Обеспечить доступ к таблицам базы данных через источник ODBC.
2. Создать приложение, содержащее форму для модификации данных.
3. Модифицировать данные.
4. Создать отчет с информацией о результатах сдачи сессии студентами.

Лабораторная работа №6

Тема «Использование утилит импорта, экспорта для резервного копирования»

Существуют три стандартных способа резервного копирования БД :экспорт, автономное резервное копирование (offline backup) и оперативное резервное копирование (online backup). Экспорт представляет собой логическое копирование базы данных; два остальных способа – это физическое копирование файлов. Физическое копирование файлов может производиться с помощью

пользовательских сценариев или утилиты Recovery Manager (RMAN). Как правило, промышленные базы данных используют в качестве основного метода физическое резервное копирование, а логическое служит вспомогательным методом. Для баз данных разработки и некоторых операций обработки с перемещением небольшого объема данных больше подходят операции логического резервного копирования.

Логическое резервное копирование базы данных предполагает чтение ее записей и внесение их в файл. Записи считываются независимо от их физического расположения. Этот тип копирования выполняет утилита экспорта. Для восстановления БД в это случае применяется утилита импорта.

Утилита экспорта ORACLE обращается к базе данных, включая словарь данных, и записывает результат в двоичный файл, называемый файлом дампа экспорта. Эта утилита имеет 4 уровня функциональности: полный (Full) режим, режим табличного пространства (Tablespace), пользовательский (User) и табличный (Table). Модифицированная версия режима Table позволяет экспортировать разделы.

Режим Full позволяет экспортировать всю базу данных. В своих схемах этот файл будет содержать команды создания всех табличных пространств, всех пользователей, все объекты, данные и привилегии.

Режим Tablespace предполагает экспорт всех объектов, содержащихся в данном табличном пространстве, включая индексы в содержащихся объектах, даже если они находятся в другом табличном пространстве.

Режим User позволяет экспортировать объекты пользователя и данные в них, все привилегии и индексы, созданные пользователем для своих объектов. Не экспортируются привилегии и индексы, созданные пользователями, не являющимися владельцами этих объектов.

В режим Table экспортируется структура указанной таблицы, индексы, привилегии вместе с данными или без них. Если указать владельца схемы , а не таблицу, то возможно экспортирование всех таблиц, принадлежащих пользователю. Возможен экспорт определенного раздела таблицы.

Экспорт можно осуществлять как посредством OEM, так и с помощью командных файлов, применяя следующие параметры:

USERID Имя пользователя и пароль учетной записи, выполняющей экспорт. Этот параметр должен быть первым в командной строке;

BUFFER Размер буфера, используемого для считывания строк данных. Значение по умолчанию зависит от системы; обычно оно сравнительно велико (> 64000) ;

FILE Имя файла дампа экспорта; по умолчанию expdat.dmp;

COMPRESS Флаг Y/N указывает, должна ли утилита экспорта сжимать фрагментированные сегменты в единые экстенты. Этот параметр влияет на то, какие конструкции storage будут содержаться в файле экспорта для объектов. По умолчанию задается Y. Хранение всех объектов в большом экстенте не всегда является лучшим решением, Для больших таблиц установите compress = N;

GRANTS Флаг Y/N указывает, будут ли экспортироваться привилегии на объекты базы данных. По умолчанию задается Y;

INDEXES Флаг Y/N указывает, будут ли экспортироваться индексы таблиц. По умолчанию задается Y;

DIRECT	Флаг Y/N указывает, будут ли производиться прямой экспорт, который обходит кэш буфера во время экспорта, что существенно повышает его производительность. По умолчанию задается N;
LOG	Имя файла, в который будет записан журнал экспорта;
ROWS	Флаг Y/N указывает, будут ли экспортироваться строки. Если задано N, то в экспортном файле будут создаваться только DDL для объектов базы данных. По умолчанию задается Y;
CONSISTENT	Флаг Y/N указывает, сохранять ли для всех экспортированных объектов вариант, согласованный по чтению. Это необходимо, если в процессе экспорта связанные друг с другом таблицы модифицируются пользователем;
FULL	Если значение этого параметра равно Y, будет выполнен полный экспорт базы данных. По умолчанию задается N;
OWNER	Список экспортируемых учетных записей базы данных. Для этих учетных записей может быть выполнен экспорт в режиме User;
TABLES	Список экспортируемых таблиц. Для них может быть выполнен экспорт в режиме Table. В Oracle 9i этот параметр поддерживает использование символов шаблона ‘%’ и ‘_’ ;
RECORD-LENGTH	Длина записи файла дампа экспорта в байтах. Обычно оставляют значение по умолчанию,

	если не предполагается переносить файлы экспорта между различными операционными системами;
TRIGGERS	Флаг Y/N указывает, следует ли экспортировать триггер. По умолчанию задается Y;
STATISTICS	Параметр, который указывает, будут ли использованы команды analyze для экспорта объектов в файл дампа экспорта. Допустимыми значениями являются COMPUTE, ESTIMATE (по умолчанию) и N. В более ранних версиях Oracle этот параметр назывался analyze;
PARFILE	Имя файла параметров, передаваемого в утилиту Export. Этот файл может содержать значения для всех перечисленных здесь параметров;
CONSTRAINTS	Флаг Y/N указывает, экспортируются ли ограничения на таблицы. По умолчанию задается Y;
FEEDBACK	Число строк, по достижении которого на экране будет отображаться процесс экспорта таблицы. Значение по умолчанию равно 0; значит, никакой обратной связи не предусматривается, пока вся таблица не будет полностью экспортирована;
FILESIZE	Максимальный размер файла дампа экспорта. Если в элементе file перечислено несколько файлов, результат экспорта будет записываться в них на основе значений filesize;
FLASHBACK_ SCN	Указывает, что для активизации нарушения последовательности действий будет использоваться SCN Export. Экспорт

	выполняется с такой согласованностью данных, которая задана в этом SCN;
FLASHBACK_ TIME	Время, используемое для получения SCN, ближайшее к заданному времени. Экспорт производится с такой согласованностью данных, которая задана' в этом SCN;
QUERY	Конструкция where, которая будет применяться к каждой таблице во время экспорта;
RESUMABLE	Флаг Y/N указывает, будет ли возобновляться сеанс, если произойдут ошибки. По умолчанию задается как N;
RESUMABLE_ NAME	Заданное значение вставляется в представление DBA_RESUMABLE, что помогает идентифицировать возобновляемую команду;
RESUMABLE_ TIMEOUT	время ожидания для возобновляемой команды;
TTS_FULL_ CHECK	Выполняет полную или частичную проверку зависимости для транспортируемых табличных пространств;
VOLSIZE	Число байт, которые записываются в каждый том ленты;
TABLESPACES	В Oracle9i табличные пространства, таблицы которых будут экспортированы, включая все таблицы, которые имеют разделы, расположенные в специальных табличных пространствах;
TRANSPORT_ TABLE_SPACE	Задается Y, если используется возможность вставляемого табличного пространства. Используйте вместе с ключевым tablespace. По умолчанию задается N;
TEMPLATE	Имя шаблона, используемое для вызова режима

экспорта iAS;

В зависимости от того, как владельцы схем приложений и их объекты распределены по табличным пространствам, задача экспорта может решать легче или сложнее. Так, если объекты в табличном пространстве принадлежат нескольким пользователям, можно экспортировать все эти объекты с помощью одной команды независимо от того, какой схеме принадлежит объект:

Exp demo/demo tablespaces=DATA

Экспорт всех таблиц в версии ORACLE 9i, находящихся в определенном табличном пространстве, можно производить с помощью параметра tablespace. Если в определенном табличном пространстве какая-нибудь таблица имеет разделы, будет экспортирована вся таблица. Задание параметра indexes=y, позволяет экспортировать связанные с этой таблицей индексы независимо от того, в каком табличном пространстве они находятся.

Для экспорта всей таблицы используется параметр tables утилиты Export:

Exp system/manager FILE=expdat.dmp TABLES=(Thumper.SALES).

Для экспорта определенного раздела или подраздела следует указать его имя после имени таблицы, отделив двоеточием:

Exp system/manager FILE=expdat.dmp
TABLES=(Thumper.SALES:Part1).

Для экспорта нескольких строк для таблицы можно воспользоваться мастером экспорта OEM Server Manager Export Wizard. Выбрав таблицу и задав опции экспорта, включая задание дополнительных свойств на вкладках:

- Общие (General), можно указать желаемую форму сбора статистических данных, местонахождение файла

журнала, необходимость производить экспорт в режиме прямого пути (direct path);

- Настройка (Tuning) позволяет указать требуется ли представление данных с согласованными операциями чтения (consistent=y), слияние экстентов (compress=y) и будут ли изменяться заданные по умолчанию длина записи и размера буфера;
- Запрос (Query) используется для задания конструкции Where.

Так же можно задать время выполнения экспорта, имя файла и его местонахождение.

Задание на лабораторную работу

1. Экспортировать объекты, принадлежащие пользователю System.
2. Выполнить экспорт таблиц Stud и Session.
3. Выполнить полный экспорт.

Лабораторная работа №7

Тема «Утилита импорта»

Утилита импорта считывает файл дампа экспорта и запускает находящиеся в нем команды. Импорт можно осуществлять интерактивно или с помощью командных файлов. Параметры времени выполнения, которые можно определить для импорта следующие:

USERID Имя пользователя и пароль учетной записи, выполняющей импорт; это должен быть первый параметр; ключевое слово "userid=" указывать не обязательно;

BUFFER Размер буфера, используемого для считывания строк данных. Значение по умолчанию зависит от операционной системы; обычно его задают высоким (>100000) ;

FILE Имя импортируемого файла дампа экспорта;

SHOW Флаг Y/N, определяющий, нужно ли отображать или исполнять содержание файла дампа экспорта. По умолчанию задается N;

IGNORE Флаг Y/N, определяющий, должна ли утилита импорта игнорировать ошибки, возникающие при выполнении команд create. Используется, если импортируемые объекты уже существуют. По умолчанию задается N;

GRANTS Флаг Y/N, определяющий, должны ли быть импортированы привилегии на объекты базы данных;

INDEXES Флаг Y/N, определяющий, должны ли быть импортированы индексы таблицы. По

	умолчанию задается Y;
ROWS	Флаг Y/N, определяющий, должны ли быть импортированы строки. Если значение этого флага равно N, будут выполнены только команды языка DDL для объектов базы данных. По умолчанию задается Y;
LOG	Имя файла, для которого будет записываться импорт;
FULL	Флаг Y/N; если он равен Y, то импортируется файл дампа полного импорта. По умолчанию задается N;
FROMUSER	Список учетных записей базы данных, объекты которых должны быть считаны из файла дампа экспорта (если full=n) ;
TOUSER	Список учетных записей базы данных, в которые нужно импортировать объекты из файла дампа экспорта. Параметры fromuser и touser не обязательно совпадают;
TABLES	Список импортируемых таблиц. В Oracle9i для имен таблиц допускается использование символов ‘%’ и ‘_’ ;
RECORDLENGTH	Длина (в байтах) записи файла дампа экспорта. Обычно оставляют значение по умолчанию, если вы не собираетесь передавать файл экспорта между различными операционными системами;
COMMIT	Флаг Y/N, показывающий, должен ли импорт завершаться командой commit после ввода

	каждого массива (размер которого определяется в параметре buffer). Если он равен N (по умолчанию), импорт будет завершаться командой commit после ввода каждой таблицы. Для больших таблиц значение commit требует такого же по размеру сегмента отката;
PARFILE	Имя файла параметров, передаваемого в утилиту Import. Этот файл может содержать элементы для всех перечисленных здесь параметров;
CONSTRAINTS	Флаг Y/N, показывающий, будут ли импортироваться ограничения на таблицы. По умолчанию Y;
DESTROY	Флаг Y/N, показывающий, будут ли выполняться команды create tablespace, обнаруженные в файлах дампа полных экспортов (это приведет к уничтожению файлов данных в базах данных, в которые осуществляется импорт). По умолчанию N;
INDEXFILE	Очень мощное средство, позволяющее записывать все команды create table, create cluster и create index в файл, а не выполнять их. Все команды, кроме create index, будут закомментированы. Если задано constraints=y, то ограничения тоже будут записаны в файл. Затем (внеся незначительные изменения) этот файл можно запустить после завершения импорта с параметром indexes=n. Средство

	очень полезно для распределения таблиц и индексов по различным табличным пространствам;
SKIP_UNUSABLE_ INDEXES	Флаг Y/N, показывающий, следует ли в процессе импорта пропускать индексы разделов, помеченные как неиспользуемые. Чтобы повысить производительность создания индексов, можно пропустить этот этап и впоследствии создать индексы вручную. По умолчанию N;
FEEDBACK	Количество строк, по достижении которого на экране будет отображаться прогресс импорта таблицы. Если значение по умолчанию равно нулю, никакого прогресса не будет отображено, пока таблица не будет полностью импортирована;
TOID_ NOVALIDATE	Позволяет в процессе импорта пропустить проверку указанных типов объектов на допустимость;
FILESIZE	Максимальный размер дампа, заданный при экспорте, если использовался этот параметр;
STATISTICS	Флаг, показывающий, нужно ли импортировать подсчитанные ранее статистические данные. По умолчанию ALWAYS; другие значения: NONE, SAFE (для статистики, не вызывающей сомнения) и RECALCULATE (подсчитать заново во время импорта) ;
RESUMABLE	Флаг Y/N, показывающий, возобновляется ли

	сеанс после возникновения ошибки;
RESUMABLE_ NAME	Заданное значение вставляется в представление DBA_RESUMABLE, что помогает идентифицировать возобновляемую команду;
RESUMABLE_ TIMEOUT	Время ожидания для возобновляемой команды;
COMPILE	Флаг Y/N, показывающий, будут ли во время импорта перекомпилироваться процедуры, функции и модули. По умолчанию Y;
VOLSIZE	Максимальное число байт в файле каждого тома ленты;
TRANSPORT_ TABLESPACE	Флаг Y/N, показывающий, что транспортируемые метаданные табличного пространства должны импортироваться в базу данных. По умолчанию N;
TABLESPACES	Имя или список имен табличных пространств, транспортируемых в базу данных;
DATAFILES	Список файлов данных, транспортируемых в базу данных;
TTS_OWNERS	Имя или список имен владельцев данных в транспортируемом табличном пространстве.

Некоторые параметры импорта конфликтуют друг с другом или приводят к несогласованности инструкций для импорта. Например, если параметр full=y, а owner=hr, то импорт закончится неудачно, поскольку параметр full означает полный импорт, а owner указывает на пользовательский import.

Параметр destroy может быть очень полезным для администраторов, обеспечивающих работу нескольких баз данных на одном сервере. Поскольку в процессе полного экспорта БД осуществляется запись всего словаря данных, в файл дампа экспорта заносятся определения табличных пространств и файлов данных. При импорте результатов полного экспорта первой базы во вторую будут выполнены команды create tablespace, находящиеся в файле дампа экспорта. Эти команды определяют создание файлов в том же каталоге и с теми же именами, что и файлы из первой БД. И если не указать значение destroy=n (параметр по умолчанию), файлы данных первой БД могут быть переписаны. Имена табличных пространств экспортируются как часть определения объекта. Если вы импортируете данные из одного экземпляра в отдельный экземпляр, создайте табличные пространства с целевой базой данных, имеющие те же имена, как и в исходном экземпляре.

Требования для сегментов отката

По умолчанию база данных выполняет команду commit после полного импорта каждой таблицы. Это означает, что сегмент отката должен вместить RowID для каждой импортируемой строки. Чтобы уменьшить размеры элементов сегментов отката, определите commit=y и задайте значение для параметра buffer. Теперь команда commit будет выполняться после каждого ввода информации объемом buffer, как показано в следующем примере. В первой приведенной в нем команде импорта операция commit выполняется после загрузки всей таблицы, во второй команде — после загрузки каждый 64000 байт данных.

```
imp system/manager file=expdat.dmp
imp system/manager file=expdat.dmp buffer=64000 commit=y
```

Значение параметра buffer должно быть достаточно большим, чтобы обработать самую большую импортируемую строку. При использовании параметра commit=y следует помнить, что данная команда выполняется для каждого массива buffer. Поэтому даже в случае неудачного импорта таблицы некоторые ее строки все же могут оказаться импортированными.

Для перемещения объектов от одного пользователя к другому с помощью утилиты импорта/экспорта выполняется пользовательский экспорт владельца объектов. В ходе импорта нужно определить владельца объектов как fromuser, а учетную запись, которой будут принадлежать объекты, как touser.

Например, чтобы скопировать объекты пользователя THUMPER в учетную запись FLOWER, нужно выполнить две команды. Первая из них экспортирует владельца записи THUMPER, а вторая импортирует принадлежащие ему объекты в запись FLOWER.

```
exp system/manager fil_e=thumper.flat owner=thumper grants=N
  indexes=Y compress=Y rows=Y
imp system/manager file=thumper.dat FROMUSER=thumper
TOUSER=flower rows=Y indexes=Y
```

Задание на лабораторную работу

1. Импортировать объекты, принадлежащие пользователю System.
2. Выполнить импорт таблиц Stud и Session.
3. Выполнить полный импорт.
4. Осуществить перемещение объектов другому пользователю.

Лабораторная работа №8

Тема: «Менеджер восстановления»

Хотя утилита RMAN является отличным средством защиты базы данных и обеспечивает успешное восстановление, ее использование не отменяет необходимости процедур резервного копирования. Вместе с тем следует отметить, что утилита обеспечивает интерфейс прикладной программы (Application Program Interface, API) для управления носителем информации, что позволяет легко устанавливать стороннее программное обеспечение для такого управления. Т.к. RMAN выполняет резервное копирование в оперативном режиме, каждый разрушенный блок считывается повторно в целях обеспечения согласованности данных. После завершения процедур резервного копирования RMAN вызывает те же процедуры ядра, которые используются всеми процедурами для проверки целостности блоков данных ORACLE. Во время инкрементального резервного копирования RMAN захватывает только те блоки, которые изменились со времени предыдущего резервного копирования. С помощью инкрементального копирования возможно восстанавливать базу данных, даже если она находится в режиме NOARCHIVELOG. Для восстановления базы данных с дисков или ленты инструмент RMAN обеспечивает автоматическое распараллеливание операций резервного копирования и восстановления. Он позволяет создавать дублирующие базы данных.

Для запуска RMAN вручную в командной строке операционной системы введите команду RMAN. При обращении к RMAN можно использовать следующие аргументы:

TARGET	Строка соединения для целевой базы данных;
CATALOG	Строка соединения для каталога восстановления;
NOCATALOG	Если задан, то каталог восстановления не используется;

CMDFILE	Имя входного командного файла;
LOG	Имя файла журнала выходных сообщений;
TRACE	Имя файла журнала выходных сообщений при отладке;
APPEND	Если задан, журнал открывается в режиме добавления в конец;
DEBUG	Активизирует отладку;
MSGNO	Показывает префикс RMAN-nnnn для всех сообщений ;
SEND	Посылает команду менеджеру носителя информации;
PIPE	Структурный элемент для имен абстрактных файлов;
TIMEOUT	Время ожидания (в секундах) ввода абстрактных файлов.

Если вы создадите и используете репозиторий RMAN, вы получите преимущество в виде набора метаданных, в которых хранится информация о базе данных и ее операциях резервного копирования и восстановления. Значения конфигурации RMAN можно хранить в управляющем файле и при необходимости повторно синхронизировать их с репозиторием. Параметры конфигурации по умолчанию переписываются с помощью команды configure.

Например,

Configure retention policy to none

Сообщает утилите, что стратегия хранения не определена.

Наличие каталога восстановления дает много преимуществ, хотя и не является обязательным.

Каталог восстановления следует создавать в его собственной схеме и его собственном табличном пространстве в базе данных, отделенной от всех целевых баз данных, резервное копирование которых вы намерены осуществить. Установка каталога восстановления в целевой БД не имеет смысла, поскольку при утрате базы данных будет потерян не только сам каталог, но и возможность восстановления целевой БД. Хотя этот каталог можно хранить в табличном пространстве SYSTEM, мы настоятельно рекомендуем использовать совершенно отдельное табличное пространство.

Если в базе данных нет учетной записи RMAN, то для создания каталога восстановления сначала создайте пользователя, у которого будет храниться этот каталог, и назначьте ему привилегии RECOVERY_CATALOG_OWNER, CONNECT и RESOURCE. Для создания пользователя можно использовать следующие команды.

```
connect SYSTEM/MANAGER as sysdba
create user RMAN identified by RMAN temporary tablespace TEMP
default tablespace CATTBS;
grant CONNECT, RESOURCE. RECOVERY_CATALOG_
OWNER to RMAN,
```

После создания схемы RMAN и назначения соответствующих привилегий (CONNECT, RESOURCE, RECOVERY_CATALOG_ OWNER) можно установить соединение с выполняемой утилитой RMAN (см. выше) и создать каталог восстановления, введя команды:

```
>rman target / catalog RMAN/RMAN@HR
RMAN> create catalog
```

После создания каталога зарегистрируйте базу данных, введя команду: RMAN> register database;

Приведем команды RMAN, которые можно применять для резервного копирования и восстановления базы данных.

@	Запускает командный файл;
@@	Запускает командный файл в том же каталоге в качестве другого командного файла, работающего в данный момент;
ALLOCATE CHANNEL	Устанавливает канал, который является соединением между RMAN и экземпляром базы данных;
ALLOCATE CHANNEL FOR MAINTENANCE	Выделяет канал при подготовке к вводу команд обслуживания, например delete;
ALTER DATABASE	Устанавливает или открывает базу данных;
BACKUP	Выполняет резервное копирование базы данных, табличного пространства, файла данных, архивного журнала или набора резервных копий;
BLOCKRECOVER	Восстанавливает отдельный блок данных или набор блоков данных в одном или нескольких файлах данных;
CATALOG	Добавляет в хранилище информацию о копии файла данных, архивного журнала повторов или копии управляющего файла;
CHANGE	Помечает фрагмент резервной копии, копию изображения или архивный журнал повторов как имеющий статус UNAVAILABLE или

	AVAILABLE; удаляет запись из репозитория для резервной или обычной копии; перезаписывает стратегию хранения для резервной копии ;
CONFIGURE	Конфигурирует постоянные настройки RMAN, которые применяются во всех сеансах до тех пор, пока их не изменят в явном виде или не деактивизируют;
CONNECT	Устанавливает соединение между RMAN и базой данных (целевой, вспомогательной или каталога восстановления) ;
COPY	Создает копию изображения файла данных, управляющего файла или архивного журнала повтора;
CREATE CATALOG	Создает схему для каталога восстановления;
CREATE SCRIPT	Создает сохраняемый сценарий и хранит его в каталоге восстановления;
CROSSCHECK	Определяет, существуют ли еще на диске или ленте файлы под управлением RMAN;
DELETE	Удаляет резервные и обычные копии; удаляет из каталога восстановления ссылки на них и изменяет записи о них в управляющих файлах на статус DELETED;
DELETE SCRIPT	Удаляет сохраняемый сценарий из каталога восстановления;
DROP CATALOG	Удаляет схему из каталога восстановления;

DUPLICATE	Использует резервные копии целевой базы данных для создания копии базы данных, которую можно использовать в целях тестирования или создания дублирующей БД;
EXECUTE SCRIPT	Запускает сохраняемый сценарий RMAN;
EXIT	Осуществляет выход из выполняемой утилиты RMAN;
HOST	Вызывает из RMAN подоболочку командной строки операционной системы или запускает определенную команду операционной системы;
LIST	Создает подробный листинг наборов резервных копий или копии;
PRINT SCRIPT	Выводит на экран сохраняемый сценарий
QUIT	Выход из выполняемой утилиты RMAN;
RECOVER	Применяет журналы повторов или операции инкрементального резервного копирования к восстанавливаемым наборам резервных копий или копии для их обновления в указанное время;
REGISTER	Регистрирует целевую базу данных в каталоге восстановления;
REPLICATE	Копирует управляющий файл во все места,

	указанные в параметре инициализации CONTROL_FILES;
REPORT	Осуществляет подробный анализ содержимого каталога восстановления;
RESET DATABASE	Сообщает утилите RMAN, что выполнена команда SQL alter database open resetlogs или переводит целевую базу данных в предшествующее состояние;
RESTORE	Восстанавливает файлы из наборов резервных копий или из копий на дисках; их местонахождение задается по умолчанию или может быть иным;
RESYNC	Выполняет полную повторную синхронизацию, которая создает управляющий файл моментальных снимков, а затем копирует всю новую или измененную информацию из этого файла в каталог восстановления;
RUN	Выполняет последовательность из одной или нескольких команд RMAN, которые указываются в скобках в команде run;
SEND	Посылает в один или несколько заданных каналов задаваемую поставщиком строку в кавычках;
SET	Задает следующие настройки на уровне сеанса: управляет выводом команд RMAN в журнале сообщений. Задает DBID (идентификатор БД) при восстановлении

	управляющего файла. Задает новые имена файлов для восстанавливаемых файлов данных. Задает максимальное число разрешенных повреждений блоков. Перезаписывает назначения по умолчанию архивированных журналов повторов. Задает число копий каждого элемента резервной копии. Определяет, какому каналу соответствует каждая сессия сервера. Контролирует, производит ли RMAN поиск резервных копий при использовании конфигурации Oracle Real Application Clusters. Перезаписывает формат по умолчанию для автоматической резервной копии управляющего файла;
SHOW	Выводит на экран текущие настройки configure;
SHUTDOWN	Закрывает целевую базу данных. Эта команда эквивалентна команде SQL *Plus Shutdown;
SPOOL	Записывает выходные данные RMAN в файл журнала;
SQL	Выполняет команду SQL из утилиты Recovery Manager;
STARTUP	Запускает целевую базу данных. Эта команда эквивалентна команде SQL *Plus Startup
SWITCH	Указывает, что копия файла данных является теперь текущим файлом, т.е. файлом данных,

	на который ссылается управляющий файл. Эта команда эквивалентна команде SQL alter database rename file, так как она применяется к файлам данных;
UPGRADE CATALOG	Расширяет схему каталога восстановления от более старой версии к той, которая затребована выполняемой утилитой RMAN;
VALIDATE	Проверяет набор резервных копий и сообщает, остаются ли его данные нетронутыми. RMAN сканирует все элементы резервных копий в заданных наборах резервных копий и проверяет контрольные суммы, чтобы подтвердить успешность восстановления содержимого.

Использование OEM Backup Manager

Для того чтобы использовать Backup Manager, необходимо воспользоваться таким инструментом, как OEM Management Server Console. Перед первым обращением необходимо запустить OEM Configuration Assistant, чтобы сконфигурировать репозиторий OEM. В нем будет содержаться информация об OEM. После его создания следует убедиться, что запущена служба Management Server.

Выберите соответствующую базу данных и в контекстном меню выберите Backup, после чего активизируется мастер резервного копирования (Backup Wizard). В последующих окнах вы можете выбрать предустановленную стратегию резервного копирования или создать свою собственную, частоту резервного копирования и т.д. Из инструмента OEM Management Server Console возможно восстановление файлов данных, табличных пространств, управляющих файлов и архивных журналов.

Мастер восстановления

Когда вызывается инструмент Recovery из OEM Management Server Console для проведения восстановления, он проверяет, чтобы целевая база данных была запущена и установлена, но не открыта до начала его работы. Для запуска и установки базы данных можно использовать раскрываемое меню Object. После запуска БД можно запускать мастер восстановления (Recovery Wizard) из раскрываемого меню Tools (Database Wizards | Backup Management | Recovery

Задание на лабораторную работу

1. С помощью RMAN произведите полное резервное копирование.
2. Используя RMAN, выполните резервное копирование заданного файла данных .
3. Восстановите табличную область с помощью RMAN.
4. Используя RMAN, восстановите базу данных.
5. Используя OEM Backup Manager, создайте резервную копию и восстановите базу данных.

Лабораторные работы № 9, 10

Тема «Планирование разрешений и системы защиты»

Системы управления базами данных, в особенности реляционные СУБД, стали доминирующим инструментом хранения больших массивов информации, при этом стоимость информации непрерывно растет, а ущерб от ее повреждения и несанкционированного использования может быть значительным. Сколько-нибудь развитые информационные приложения полагаются на многопользовательские СУБД, выполненные в технологии клиент/сервер. В этой связи обеспечение информационной безопасности СУБД, и в первую очередь их серверных компонентов, приобретает решающее значение для безопасности данных организации.

Для реализации защиты данных в СУБД ORACLE применяют несколько уровней безопасности:

- уровень учетных записей пользователей
- безопасность на уровне доступа к объектам БД
- уровень системы, реализующий управление глобальными привилегиями.

Рассмотрим эти уровни подробнее.

Для получения доступа к информации необходимо пройти процедуру аутентификации. Доступ в систему ORACLE может быть реализован двумя способами: либо через соответствующие механизмы операционной системы, либо используя SQL-оператор CONNECT.

CONNECT <пользователь>[/<пароль>] [<@база_данных>];

Уже после инсталляции ORACLE содержит ряд пользователей с заданными паролями. Помимо пользователей для администрирования ORACLE – SYS, SYSTEM, дополнительно для демонстрирования возможностей ORACLE был создан ряд пользователей и набор взаимосвязанных схем, каждая из которых имеет свой уровень сложности и назначение.

- hr (Human Resources - Персонал),

- oe (Order Entry - Заказы),

- pm (Product Media – Медиа-товары),

- sh (Sales History - История продаж) и некоторые другие.

Для подключения новых пользователей используется команда create user.

Команда имеет следующие параметры.

- Username имя пользователя,

- Password пароль,

- Default tablespace табличное пространство, в котором будут храниться объекты,

- Temporary tablespace табличное пространство для временных объектов,

- Quota предельный размер в табличном пространстве,

- Password limit ограничения на срок действия пароля,

- Account служит для блокирования учетной записи,

- Default role роли пользователя по умолчанию.

Например:

```
CREATE USER USER_1
IDENTIFIED BY EX#OF#PASS1
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP;
```

Все параметры пользователя, кроме имени, можно изменить, используя команду alter user.

Например:

```
ALTER USER USER_1
QUOTA 100M ON USERS;
```

Позволяет пользователю USER_1 создавать в табличном пространстве USERS сегменты размером до 100M.

Для удаления пользователя используется команда drope user.

DROP USER <пользователь> [cascade]

Параметр cascade активирует удаление всех объектов из схемы пользователя перед его удалением.

Использование команды alter user меняет параметры, оставляя схему неизменной.

Условия, необходимые для создания и изменения пользователей

– наличие системных привилегий CREATE USER и ALTER USER.

Применение консоли OEM для создания новых пользователей

После установления соединения с консолью OEM щелкните по знаку + рядом с пунктом Security, чтобы раскрыть его. Выберите пункт Users.

Для выбора пункта создания – Create возможны варианты:

- воспользоваться пунктом Create меню Object.

- через контекстное меню (правая клавиша мыши), затем выбрать пункт Create.

- щелкнуть мышью на пиктограмме в виде зеленого кубика – третья сверху в левой колонке.

На появившемся экране выберите объект User, затем нажмите кнопку Create в нижней части экрана.

После завершения этих операций на экране появится окно Create User с восемью вкладками:

- General для ввода имени и пароля (пароль необходимо вводить дважды для предотвращения ошибки). В нижней части вкладки можно установить (Locked) и снять (Unlocked) блокировку пользователя.

- Role вкладка содержит две части: вверху потенциально доступные Available и внизу предоставленные Granted пользователю роли.

- System вкладка содержит две части: вверху потенциально доступные Available и внизу предоставленные Granted пользователю системные привилегии.

- Object вкладка содержит две части: вверху потенциально доступные Available и внизу предоставленные Granted пользователю объектные привилегии по каждой схеме и объекту.

- Quota вкладка содержит размер квоты по каждому табличному пространству.

- XML

- Consumer Group

- Proxy User

Для проверки действенности установленной защиты может возникнуть необходимость войти в систему под именем тестируемого пользователя. Для этого используется команда connect.

Сложность заключается в том, что необходимо знать пароль, обычно известный только самому пользователю. Попытка просмотра пароля с помощью запроса:

SELECT

Username, Password

FROM DBA_USERS

WHERE Username=<пользователь>

не приведет к ошибке, но мы увидим зашифрованный вариант пароля. При регистрации в системе предъявленный пользователем пароль сначала шифруется, а затем сравнивается со значением, хранящимся в поле password записи о пользователе из DBA_USERS.

Есть способ обойти это препятствие.

- просмотреть и запомнить зашифрованный вариант пароля,

- сформировать и сохранить команду для восстановления пароля в зашифрованном виде, при этом не нужно знать исходного значения пароля,

- установить новый пароль для пользователя,

- войти в систему при помощи учетной записи пользователя и провести тестирование защиты,

- применить ранее сформированную команду восстановления прежнего варианта пароля пользователя.

Теперь можно проводить тестирование защиты, после окончания которого можно восстановить привычный для пользователя вариант пароля.

Команда, используемая для присвоения шифрованного пароля:

ALTER USER <пользователь> identified by VALUES <шифрованный пароль>

Задание непосредственно шифрованного варианта пароль можно также применять для пользователей, которым возможен доступ только для входа в систему при помощи авторегистрации.

Доступ к объектам БД осуществляется через механизм предоставления GRANT и отзыва REVOKE привилегий.

Объектная привилегия позволяет пользователю выполнять определенное действие на конкретной таблице - TABLE, представлении - VIEW, последовательности - SEQUENCE, процедуре - PROCEDURE, функции или пакете. Эти привилегии разрешают пользователю доступ к данным, которыми он не владеет, и могут предоставляться явно или через роли. В СУБД ORACLE под ролью понимается набор привилегий. Такие роли служат средством структуризации привилегий и облегчают их модификацию. В зависимости от типа объекта существуют различные типы объектных привилегий;

- ALTER изменение объекта (таблицы или последовательности)

- DELETE удаление из объекта (таблицы или представления)

- EXECUTE выполнение (процедуры или функции)

- INDEX создание индекса для объекта (таблицы или представления)

- INSERT добавление в объект (таблицу или представление)

- REFERENCES создание внешних ключей (только таблицы)

- SELECT извлечение данных из объекта (таблицы, представления, снимка)

- UPDATE обновление строк в объекте (таблице или представлении)

Каждое имя пользователя в базе данных считается СХЕМОЙ - доменом защиты, который может содержать объекты схемы. Доступ к базе данных и ее объектам контролируется привилегиями, назначенными каждой схеме. Большинство схем можно рассматривать как имена пользователей, т.е. учетные имена, которые позволяют пользователям соединяться с базой данных и обращаться к объектам в базе данных.

Назначение объектных привилегий

Объектные привилегии могут назначаться ролям и пользователям с помощью команды SQL GRANT. Например, следующее предложение назначает объектные привилегии SELECT, INSERT и DELETE для всех столбцов таблицы JOBS пользователям SEROV и STUD:

GRANT select, insert, delete ON jobs TO serov, stud;

Чтобы назначить объектную привилегию INSERT только для столбцов JOB_ID и JOB_TITLE таблицы JOBS тем же пользователям, введите следующее предложение:

GRANT insert(job_id, job_title) ON jobs TO serov, stud;

Чтобы назначить все объектные привилегии по представлению SALARY пользователю WALLEN, используйте групповое обозначение ALL:

GRANT ALL ON salary TO wallen;

Объектные привилегии НЕЛЬЗЯ назначать вместе с системными привилегиями и ролями в одном предложении GRANT.

ОПЦИЯ GRANT OPTION. Объектная привилегия может быть назначена пользователю с опцией GRANT OPTION. Пользователь, получивший эту специальную опцию, имеет несколько расширенных возможностей:

- Он может назначать эту объектную привилегию любому пользователю или роли в базе данных.

- Он может далее назначать эту объектную привилегию с опцией GRANT OPTION или без таковой.

- Если он получил объектные привилегии для таблицы с опцией GRANT OPTION и имеет системную привилегию CREATE VIEW или CREATE ANY VIEW, то он может создавать представления по этой таблице и назначать соответствующие привилегии по этому представлению любому пользователю или роли в базе данных.

Пользователь, схема которого содержит объект, автоматически имеет все ассоциированные объектные привилегии для этого объекта с опцией GRANT OPTION.

Особо заметьте, что опция GRANT OPTION недопустима при назначении объектной привилегии роли. ORACLE предотвращает распространение объектных привилегий через роли, так что получатели роли не могут дальше продвигать свои объектные привилегии, полученные через роли.

Условия, необходимые для назначения объектных привилегий

Чтобы назначить кому-либо объектную привилегию, вы должны удовлетворять одному из следующих условий:

- Владеть соответствующим объектом.

- Иметь для объектных привилегий, которые вы хотите назначить, опцию GRANT OPTION.

Отзыв объектных привилегий

Объектные привилегии можно отзывать с помощью команды SQL REVOKE. Например, предполагая, что вы предоставляли эти привилегии, чтобы отозвать привилегии SELECT и INSERT по таблице JOBS от пользователей SEROV и STUD, введите следующее предложение:

```
REVOKE select, insert ON jobs
 FROM serov, stud;
```

Вы можете также отозвать все привилегии по таблице DEPARTMENTS (даже если вы назначали лишь одну привилегию), назначенные вами роли HUMAN_RESOURCES, введя следующее предложение:

```
REVOKE ALL ON departments FROM human_resources;
```

Это предложение отзовет лишь те привилегии, на которые вы имеете соответствующие полномочия, но не все привилегии, которые были назначены другими. Нельзя выборочно отозвать опцию GRANT OPTION, не отзывая привилегию на объект; чтобы сделать это, следует отозвать объектную привилегию и заново назначить ее без опции GRANT OPTION. Пользователь не может отозвать объектную привилегию у самого себя.

Отзыв выборочных объектных привилегий для столбцов

Хотя пользователи могут назначать выборочные привилегии SELECT, UPDATE и REFERENCES по отдельным столбцам таблиц и представлений, они не могут выборочно отзывать такие привилегии аналогичным предложением REVOKE. Вместо этого следует сначала отозвать объектную привилегию по всем столбцам таблицы или представления, а затем вновь выборочно назначить привилегии по тем столбцам, которые должны остаться.

Например, предположим, что роли HUMAN_RESOURCES была назначена привилегия UPDATE по столбцам DEPTNO и DNAME таблицы DEPARTMENTS. Чтобы отозвать привилегию UPDATE по

столбцу DEPTNO и оставить ее по столбцу DNAME, вы должны ввести следующие два предложения:

```
REVOKE UPDATE ON departments FROM human_resources;
GRANT UPDATE (dname) ON departments TO human_resources;
```

Здесь предложение REVOKE отзывает привилегию UPDATE по всем столбцам таблицы DEPARTMENTS от роли HUMAN_RESOURCES. Предложение GRANT заново назначает привилегию UPDATE по столбцу DNAME для этой роли.

Отзыв объектной привилегии REFERENCES

Если пользователь, получивший привилегию REFERENCES, использовал эту привилегию для создания ограничения внешнего ключа (которое существует в данный момент), то пользователь, назначавший эту привилегию, может отозвать ее, лишь специфицировав в предложении REVOKE опцию CASCADE CONSTRAINTS, например:

```
REVOKE REFERENCES ON departments FROM serov CASCADE CONSTRAINTS;
```

Когда специфицирована опция CASCADE CONSTRAINTS, все ограничения внешних ключей, использующие отзываемую привилегию REFERENCES, удаляются.

Условия, необходимые для отзыва объектных привилегий

Для отзыва объектной привилегии требуется, чтобы вы были тем лицом, которое назначало эту объектную привилегию.

Назначение и отзыв объектных привилегий при помощи консоли OEM

После установления соединения с консолью OEM щелкните по знаку + рядом с пунктом Security, в раскрывшемся списке выберите Users. Аналогично раскройте Users и выберите пользователя, которому собираетесь определять привилегии.

В раскрывшемся окне выберите вкладку Object.

Вкладка содержит три окна. Для назначения объектной привилегии в левом верхнем окне:

- раскройте нужную схему (знак +)
- раскройте список объектов, например Tables
- выберете требуемый объект

В верхнем правом верхнем окне появится список доступных Available Privileges привилегий для данного объекта.

Нижнее окно содержит назначенные Granted пользователю привилегии в виде таблицы, содержащей следующие поля:

- привилегия
- схема
- объект
- наличие у пользователя права распространения привилегии другим пользователям Grant Options.

Добавлять и отзывать привилегии можно следующим образом:

Выбрать одну или несколько (клавиши Ctrl и Alt) привилегий и переместить их из окна в окно при помощи кнопок со стрелками. Вниз – назначить, Вверх – отозвать.

После настройки объектных привилегий щелкнуть на кнопке Apply – применить.

Назначение системных привилегий и ролей

Для назначения системных привилегий и ролей другим ролям и пользователям используйте команду SQL GRANT, как показано в следующем примере:

```
GRANT create session, accts_pay
TO serov, finance;
```

Объектные привилегии НЕЛЬЗЯ назначать вместе в системными привилегиями и ролями в одном предложении GRANT.

ОПЦИЯ ADMIN. Системная привилегия или роль может быть назначена с опцией ADMIN OPTION. (Нельзя включать опцию

ADMIN OPTION в предложение GRANT, назначающее роль другой роли.) Пользователь, получивший эту специальную опцию, имеет несколько расширенных возможностей:

- Он может назначать или отзывать эту системную привилегию или роль у ЛЮБОГО пользователя или роли в базе данных. (Однако он не может отозвать эту роль у самого себя.)

- Он может далее назначать эту системную привилегию или роль с опцией ADMIN OPTION.

- Если это роль, он может изменить или удалить эту роль.

Пользователь, не получивший опцию ADMIN OPTION, не может выполнять таких операций.

Когда пользователь создает роль, эта роль автоматически назначается ее создателю с опцией ADMIN OPTION.

Предположим, что администратор защиты назначает роль NEW_DBA пользователю MICHAEL с помощью следующего предложения:

GRANT new_dba TO michael WITH ADMIN OPTION;

Пользователь MICHAEL может теперь не только использовать все привилегии, подразумеваемые ролью NEW_DBA, но может также назначать, отзывать или удалять роль NEW_DBA по своему усмотрению.

Условия необходимые для назначения системных привилегий или ролей

Для назначения системной привилегии или роли пользователю требуется опция ADMIN OPTION для всех назначаемых им системных привилегий и ролей. Кроме того, пользователь с системной привилегией GRANT ANY ROLE может назначать любую роль в базе данных.

Отзыв системных привилегий и ролей

Для отзыва системных привилегий и ролей используйте команду SQL REVOKE, как показано в следующем примере:

REVOKE create table, accts_rec FROM stud, finance;

Нельзя выборочно отобрать лишь опцию ADMIN OPTION для системной привилегии или роли, не отбирая самой этот системной привилегии или роли; чтобы сделать это, отзовите системную привилегию или роль и заново назначьте ее без опции ADMIN OPTION.

Условия, необходимые для отзыва системных привилегий и ролей

Любой пользователь, имеющий опцию ADMIN OPTION для системной привилегии или роли, может отозвать эту системную привилегию или роль у любого пользователя или роли базы данных (не требуется, чтобы пользователь, у которого отзывается привилегия или роль, в свое время получил ее у отзывающего пользователя). Кроме того, любой пользователь с системной привилегией GRANT ANY ROLE может отозвать любую роль.

Назначение и отзыв системных привилегий и ролей при помощи консоли OEM.

После установления соединения с консолью OEM щелкните по знаку + рядом с пунктом Security, в раскрывшемся списке выберите Users. Аналогично раскройте Users и выберите пользователя, которому собираетесь определять привилегии.

В раскрывшемся окне выберите вкладку System для работы с системными привилегиями или Role с ролями.

Вкладки содержат по два окна.

В верхнем окне список доступных Available привилегий или ролей.

Нижнее окно содержит назначенные Granted пользователю привилегии (роли).

Добавлять и отзывать привилегии(роли) можно следующим образом:

Выбрать одну или несколько (клавиши Ctrl и Alt) привилегий(ролей) и переместить их из окна в окно при помощи кнопок со стрелками. Вниз – назначить, Вверх – отозвать.

После настройки щелкнуть на кнопке Apply – применить.

Создание роли

Для создания роли можно воспользоваться командой SQL create role или в консоли OEM. Использование роли может быть защищено ассоциированным паролем:

CREATE ROLE <имя роли> [IDENTIFIED BY <пароль>];

Если вам назначена роль, защищенная паролем, то вы можете включать или отключать эту роль, лишь предоставляя правильный пароль в команде SET ROLE.

Для удаления роли используйте команду SQL DROP ROLE, как показывает следующий пример:

DROP ROLE clerk;

Условия, необходимые для удаления ролей

Для удаления роли вы должны либо иметь системную привилегию DROP ANY ROLE, либо данная роль должна была быть вам назначена с опцией ADMIN OPTION.

Назначение и отзыв привилегий и ролей для группы PUBLIC

Привилегии и роли можно также назначать и отзывать у группы пользователей PUBLIC. Поскольку группа PUBLIC доступна каждому пользователю базы данных, все привилегии и роли, назначенные PUBLIC, доступны каждому пользователю базы данных

Вы должны назначать группе PUBLIC лишь те привилегии и роли, которые действительно необходимы каждому пользователю. Эта рекомендация согласуется с общим правилом, согласно

которому в любой момент каждый пользователь базы данных должен иметь лишь те привилегии, которые требуются ему для успешного выполнения текущей задачи.

Отзыв прав у PUBLIC может повлечь за собой значительные каскадные эффекты, в зависимости от того, какая привилегия отзывается. Если у PUBLIC отзывается любая привилегия, связанная с операцией DML (например, SELECT ANY TABLE, UPDATE ON, и т.п.), то все процедуры в базе данных (включая функции и пакеты) должны быть заново авторизованы, прежде чем их можно будет использовать снова. Поэтому будьте осторожны, назначая группе PUBLIC привилегии, связанные с операциями DML.

Политика защиты приложений

Разрабатывайте политику защиты для каждого приложения.

Например, каждое создаваемое приложение базы данных должно иметь одну или несколько ролей приложения, чтобы предоставлять различные степени защиты при исполнении этого приложения.

Роли приложений могут назначаться ролям пользователей, либо непосредственно конкретным именам пользователей.

Роли и управление привилегиями приложений

Так как большинство приложений базы данных требуют множества разнообразных привилегий на различные объекты схем, отслеживание того, какие привилегии требуются для каждого приложения, может быть непростым делом. Кроме того, авторизация пользователей для работы с приложением может потребовать большого количества операций GRANT. Чтобы упростить управление привилегиями приложений, для каждого приложения необходимо создать роль и назначить ей все привилегии, требуемые для выполнения приложения. На самом деле, приложение может иметь несколько ролей, каждую с

собственным набором привилегий, дающим больше или меньше возможностей при работе с приложением.

Группирование привилегий приложения в одну роль облегчает управление привилегиями. Рассмотрите следующие административные возможности:

- Вы можете назначать пользователям, работающим с приложением, одну роль, а не множество индивидуальных привилегий. По мере того как изменяются потребности пользователей в доступе, необходимо выполнить всего одну операцию REVOKE или GRANT для роли вместо многочисленных операций для индивидуальных привилегий.

- Если нужно изменить привилегии, ассоциированные с приложением, это необходимо сделать лишь с привилегиями, назначенными одной роли, а не с привилегиями, которые были назначены всем пользователям этого приложения.

- Вы можете определить, какие привилегии необходимы для выполнения конкретного приложения, опросив представления словаря данных ROLE_TAB_PRIVS и ROLE_SYS_PRIVS.

- Вы можете определить, какие пользователи имеют привилегии для тех или иных приложений, опросив представление словаря данных DBA_ROLE_PRIVS.

Включение ролей приложений

Каждый пользователь может использовать много приложений и ассоциированных ролей. Однако вы должны разрешать пользователю иметь в каждый момент лишь те привилегии, которые ассоциированы с ролью текущего приложения, с которым работает этот пользователь.

Явное включение ролей

Пользователь (или приложение) может явно включить роль с помощью команды SQL SET ROLE. Предложение SET ROLE включает все специфицированные в нем роли, при условии, что они

были назначены пользователю. Все назначенные пользователю роли, которые явно не специфицированы в предложении SET ROLE, выключаются, независимо от того, были ли они включены в этот момент.

Когда имеют эффект назначения и отзывы?

В зависимости от того, что назначается или отзывается, эффект этой операции может проявляться в различные моменты:

- Все назначения/отзывы привилегий (системных и объектных) кому угодно (пользователям, ролям, PUBLIC) наблюдаются немедленно.

- Все назначения/отзывы ролей кому угодно (пользователям, ролям, PUBLIC) наблюдаются лишь после того, как текущая сессия пользователя выдаст предложение SET ROLE для повторного включения роли, или при создании новой сессии пользователя.

Задание на лабораторные работы

1. Изучить возможности создания новых пользователей с помощью команд SQL и в консоли OEM.
2. Используя обе эти возможности, создать три типа пользователей:

- с правами администратора БД,

- пользователь с правами создавать свои объекты в БД,

- сторонний посетитель.

3. От имени второго пользователя создать взаимосвязанные таблицы.
4. На основе этих таблиц создать представления и процедуры, делегировать стороннему пользователю права на просмотр данных при помощи представлений, запретив

непосредственный доступ к исходным таблицам. Для этого использовать сначала непосредственное предоставление привилегий, а затем через специально созданные роли. Можно также воспользоваться таблицами из схем стандартных пользователей ORACLE или экспортировать собственные БД, созданные заранее.

5. Протестировать права доступа пользователей в системе, включая проверку прав одного из стандартных пользователей ORACLE. После окончания процедуры тестирования вернуть ему прежний пароль.

Лабораторная работа № 11

Тема «Репликации»

Для копирования и обмена информацией между базами данных можно использовать репликации. Одним из механизмов реплицирования является использование моментальных снимков – snapshot в последнее время в ORACLE используется другое название – материализованное представление materialized view. При этом можно выбирать таблицы, а также результаты их вертикальной и горизонтальной фильтрации.

Если материализованное представление является точной копией записей в удаленной таблице, снимок называется простым simple snapshot. В этом случае база данных может использовать журнал материализованных представлений для пересылки только изменений. Иначе снимок является сложным complex snapshot и выполняется полное копирование.

Обычно репликации используются при следующих условиях:

- Создание редко обновляемых хранилищ данных, предназначенных только для чтения.

- Ситуация, когда поддерживается несколько узлов в системе распределенной базы данных.

Принимая решение о том, создавать ли снимки по данной таблице, вы должны принимать во внимание также и стоимость. Каждая база данных, вовлекаемая в связь главная таблица/снимок, принимает на себя дополнительные накладные расходы, - как по обработке (для обновления каждого снимка), так и по памяти (для хранения каждого снимка). Неоправданное использование снимков без необходимости снижает производительность базы данных и увеличивает расходы дисковой памяти.

Создание снимков

Создать локальный снимок можно с помощью команды SQL CREATE SNAPSHOT (CREATE MATERIALIAZED VIEW). Как и при создании таблицы, вы можете специфицировать характеристики памяти для блоков данных снимка, размеры и распределение экстентов, а также табличное пространство, в котором создается снимок; альтернативно вы можете специфицировать кластер, в котором будет создан снимок. Как уникальную особенность снимков, вы можете также указать, как должен обновляться снимок, и задать распределенный запрос, который определяет этот снимок. Например, следующее предложение CREATE SNAPSHOT определяет локальный снимок для дублирования таблицы EMP, расположенной в базе данных BASE:

```
CREATE SNAPSHOT emp_sf
 PCTFREE 5 PCTUSED 60
 TABLESPACE USERS
 STORAGE (INITIAL 50K NEXT 50K PCTINCREASE 50)
 REFRESH FAST
```

```
START WITH sysdate
NEXT sysdate + 7
AS SELECT * FROM emp@base;
```

Когда снимок создается, он немедленно заполняется строками, возвращаемыми запросом, который определяет этот снимок. Впоследствии снимок периодически обновляется, как специфицировано фразой REFRESH.

Именование снимков

Снимки содержатся в схеме пользователя. Имя снимка должно быть уникальным по отношению к другим объектам в этой схеме (таким как таблицы и представления).

Спецификация определяющего запроса снимка

Определяющий запрос снимка может быть любым действительным запросом по таблицам, представлениям или другим снимкам, не принадлежащим пользователю SYS. Этот запрос не может содержать фраз ORDER BY или FOR UPDATE. Кроме того, простые снимки определяются через представления, которые не могут содержать фраз GROUP BY или CONNECT BY, соединений, подзапросов и операторов множеств.

Запрос, определяющий снимок, может создавать структуру, отличную от структуры главной таблицы. Например, следующее предложение CREATE SNAPSHOT создает локальный снимок с именем EMP_LOC_1 по главной таблице на BASE, который содержит лишь столбцы EMPNO, ENAME и MGR главной таблицы, и лишь те строки, которые содержат сотрудников отдела 10:

```
CREATE SNAPSHOT emp_loc_1
 AS SELECT empno, ename, mgr
```

FROM emp@base
WHERE deptno = 10;

Условия, необходимые для создания материализованных представлений

Для создания снимков необходимы следующие группы привилегий:

- Чтобы создать снимок в своей схеме, вы должны иметь системные привилегии CREATE SNAPSHOT, CREATE TABLE, CREATE VIEW и CREATE INDEX - только для простых снимков, а также привилегию SELECT по главным таблицам.

- Чтобы создать снимок в схеме другого пользователя, вы должны иметь системные привилегии CREATE ANY SNAPSHOT, CREATE ANY TABLE, CREATE ANY VIEW и CREATE ANY INDEX - только для простых снимков. Кроме того, владелец снимка должен иметь соответствующие привилегии SELECT по главным таблицам.

В любом случае, владелец снимка должен также иметь достаточную квоту на табличное пространство, в котором создается снимок.

Столь большой набор привилегий, требуемый для создания снимка, объясняется объектами, которые также должны быть созданы от имени этого снимка.

Снимки могут быть предназначены только для чтения или быть обновляемыми.

Различают четыре вида обновлений: compete - полное, fast - частичное или быстрое, force - принудительное и never - отсутствие обновлений. При полном обновлении данные представления полностью копируются при каждом обновлении. Поскольку частичное обновление затрагивает только изменения, произошедшие с момента предыдущего обновления, оно обычно проходит быстрее. Обычно быстрое обновление превосходит по скорости полное, если

изменилось менее четверти строк таблицы. Для фиксации изменений используются журналы материализованных представлений. Если журнал не создан, допустимо выполнение только полного варианта обновления.

Применение консоли OEM для создания моментальных снимков

После установления соединения с консолью OEM щелкните по знаку + рядом с пунктом Schema, чтобы раскрыть его. Выберите схему, в которой создается снимок.

Для выбора пункта создания – Create возможны варианты:

- воспользоваться пунктом Create меню Object.

- через контекстное меню (правая клавиша мыши), затем выбрать пункт Create.

- выбрать раздел Tables и щелкнуть мышью на пиктограмме в виде зеленого кубика – третья сверху в левой колонке.

На появившемся экране выберите объект Materialized View(Snapshot), затем нажмите кнопку Create в нижней части экрана.

После завершения этих операций на экране появится окно Create Materialized View с пятью вкладками:

- General для ввода имени и описания нового снимка, а также имен схемы и табличного пространства, которые предполагается использовать. Также можно указать, будет ли снимок обновляемым updatable.

- Refresh можно задавать тип обновления, метод обновления: первичный ключ или RowID, когда проводить обновление: по требованию, после каждой фиксации, автоматически с заданным периодом в днях или никогда, имя сегмента отката, который будет использоваться на главном - Master или локальном – Local сайте.

- Storage позволяет модифицировать параметры памяти для снимка.

- Index Storage при использовании в определяющем снимок запросе объединения или агрегата для него создается индекс. Здесь можно менять параметры индекса.

- Options позволяет указать, чтобы снимок загружался параллельно.

Завершается создание щелчком мыши на кнопке Create внизу окна.

Создание журнала снимков для простых снимков

Создание журнала снимков уменьшает объем обработки и время, необходимое для обновления простого снимка. Журналы снимков не могут использоваться со сложными снимками. Журнал снимков ассоциируется с единственной главной таблицей; аналогично, главная таблица может иметь лишь один журнал снимков. Если на одной и той же главной таблице базируются несколько простых снимков, все они используют тот же самый журнал снимков.

Создавайте журнал снимков в одной базе данных с главной таблицей, используя команду SQL CREATE SNAPSHOT LOG. Вы можете установить опции памяти для блоков данных, размеров и распределения экстентов, а также табличное пространство для журнала снимков. Например, следующее предложение создает журнал снимков, ассоциированный с таблицей EMP:

```
CREATE SNAPSHOT LOG ON emp
TABLESPACE users
STORAGE (INITIAL 10K NEXT 10K PCTINCREASE 50);
```

Для создания журнала снимков можно использовать консоль OEM.

Именование журналов снимков

ORACLE автоматически создает журнал снимков в той схеме, которая содержит главную таблицу. Вы не можете специфицировать имя для журнала снимков (это имя неявно дается ORACLE); поэтому вопросы уникальности имени не возникают.

Условия, необходимые для создания журналов снимков

Если вы владеете главной таблицей, то для создания журнала снимков вы должны иметь системные привилегии CREATE TABLE и CREATE TRIGGER. Если вы создаете журнал снимков для главной таблицы другого пользователя, то вы должны иметь системные привилегии CREATE ANY TABLE и CREATE ANY TRIGGER. В любом случае владелец журнала снимков должен иметь достаточную квоту в табличном пространстве, в котором создается этот журнал.

Условия, необходимые для создания журнала снимков, непосредственно определяются привилегиями, необходимыми для создания объектов, реализующих этот журнал снимков.

Использование снимков

Снимки опрашиваются точно так же, как таблица или представление. Например, следующее предложение опрашивает снимок с именем EMP:

```
SELECT * FROM emp;
```

Так как снимки можно только читать, вы не можете выдавать по снимкам предложений INSERT, UPDATE или DELETE; если вы сделаете это, будет возвращена ошибка. Хотя предложения INSERT, UPDATE и DELETE могли бы быть выданы по базовой таблице снимка, такие предложения привели бы к нарушению снимка. Никогда не манипулируйте данными в базовой таблице снимка.

Создание представлений и синонимов по снимкам

На базе снимков можно определять представления и синонимы. Следующее предложение создает представление по снимку EMP:

```
CREATE VIEW sales_dept AS
SELECT ename, empno
FROM emp
WHERE deptno = 10;
```

Условия, необходимые для использования снимка

Чтобы опрашивать снимок, вы должны иметь для этого снимка объектную привилегию SELECT, полученную либо явно, либо через роль.

Индексирование снимков

Чтобы увеличить производительность запросов по снимку, вы можете создавать индексы по этому снимку. Чтобы индексировать столбец снимка, вы должны создать индекс по нижележащей таблице "SNAP$", которая была создана для хранения строк этого снимка.

Удаление снимков

Вы можете удалить снимок независимо от его главной таблицы или журнала снимков. Чтобы удалить локальный снимок, используйте команду SQL DROP SNAPSHOT.

Например:

```
DROP SNAPSHOT emp;
```

Если вы удаляете снимок по главной таблице, который был единственным, то вы должны также удалить журнал снимков этой главной таблицы, если он существует.

Условия, необходимые для удаления снимков

Чтобы удалить снимок, необходимо быть его владельцем или иметь системные привилегии DROP ANY SNAPSHOT, DROP ANY TABLE и DROP ANY VIEW.

Удаление журналов снимков

Вы можете удалить журнал снимков независимо от его главной таблицы и от существующих снимков. Вы можете решить удалить журнал снимков по одной из следующих причин:

- Все простые снимки главной таблицы были удалены.

- Все простые снимки главной таблицы должны быть полностью (а не быстро) обновлены.

Чтобы удалить локальный журнал снимков, используйте команду SQL DROP SNAPSHOT LOG, например:

```
DROP SNAPSHOT LOG emp_log;
```

Условия, необходимые для удаления журнала снимков

Чтобы удалить журнал снимков, необходимо быть владельцем его главной таблицы или иметь системную привилегию DROP ANY TABLE.

Перед созданием материализованного представления необходимо создать связь базы данных с БД источника. Связь может быть публичной public – доступной всем учетным записям БД, или частной – доступной только создающему ее пользователю. При создании связи БД необходимо определить имя и пароль учетной записи для соединения с ней, а также название службы, связанной с удаленной базой данных. По умолчанию, будут использоваться ваше имя и пароль в локальной БД.

Пример создания публичной связи:

```
create public database link NEW_LINK
connect to USER_1 identified by EX#OF#PASS
using 'BASE'
```

открывается сеанс связи в БД - BASE , где будет зарегистрирован пользователь с учетной записью USER_1 и паролем EX#OF#PASS.

При обращении к таблице имя связи добавляется к имени таблицы после знака @. Например:

```
select * from table_name@new_link.
```

Задание на лабораторную работу

1. В БД источнике создать пользователя, наделив его правами администрирования снимков:
2. По одной из таблиц создать снимок и журнал снимков.
3. Реализовать материализованное представление.
4. В локальной БД создать пользователя с правом просмотра снимка
5. Проверить результат репликации.

Лабораторная работа № 12

Тема «Реализация приложений»

Трехуровневая архитектура – это расширение модели клиент сервер. Функции уровней зависят от реализации. MIDAS - multi-tired distributed application service suite. Это технология Borland для создания трехуровневых приложений баз данных. Применение данной технологии позволяет быстро разрабатывать надежные и простые в сопровождении распределенные БД, которые практически не зависят от формата хранения данных на сервере. Эти трехуровневые приложения состоят из следующих компонентов, или слоев:

1. Первый (самый нижний) слой – Сервер базы данных. Это может быть любая из имеющихся на рынке систем управления базами данных (Oracle, Interbase, MS SQL, MySQL, Sybase, Paradox и пр.). Назовем его уровнем БД.

2. Средний слой – Сервер приложений , используемый для реализации деловой логики приложения, и транспортный слой. Посредством этого компонента осуществляется доступ клиентов к данным, хранимым в слое БД: передача запросов, результатов, проверка на правильность вносимых данных, обновление данных. Транспортный уровень.

3. Самый верхний слой - клиентское приложение. Используя механизмы транспортного уровня, применяется для презентации приложений.

Клиент

Применение данной технологии позволяет создать клиентское, приложение, практически не требующее настройки. Оно может даже и не знать о способе физического хранения данных на сервере. Технология MIDAS как раз и предназначена для реализации связи между транспортным уровнем и клиентским приложением.

Доступ

MIDAS предоставляет несколько протоколов для связи транспортного и клиентского уровня. Это: DCOM; Socket; Web; Corba;

DCOM Этот режим связи основан на использовании встроенных возможностей Windows. Из числа достоинств данного метода выделим следующие:

- наличие встроенной поддержки в Windows98/2000/XP/NT. DCOM может быть установлен и на Win95 как отдельное приложение;

- отсутствие необходимости в запуске дополнительных приложений на стороне сервера для управления подключением клиентов;

- реализация возможности автоматического запуска сервера при обращении к нему клиента и его выгрузки при отсутствии обращений.

Недостатки метода:

- наличие проблем с работой DCOM в сетях с контролером домена не NT. Это значит, что полноценное функционирование DCOM-приложений возможно только на базе серверных версий Windows NT/2000…;

- отсутствие возможности создания прозрачности положения сервера. То есть клиент должен явно указать, на какой машине в сети установлен сервер;

- использовать DCOM можно только на Windows-платформах.

Socket Один из простейших случаев реализации связи. В его основе лежит использование сокетов и протокола TCP/IP.

Достоинства:

- может функционировать на любой платформе, использующей TCP/IP;

- требует минимум установленных компонентов в системе.

Недостатки:

- требует дополнительного приложения, установленного на сервере для постоянного отслеживания запросов клиентов;

- отсутствие механизма обеспечения безопасности. Другими словами, все серверы приложений на этой машине могут быть использованы любым клиентом, имеющим доступ к TCP/IP.

Web Основан на протоколе HTTP. Незаменимая вещь при необходимости обеспечения доступа к данным извне.

Достоинства:

-позволяет организовать возов объекта с машины находящейся за пределами сегмента сети.

Недостатки:

-требует установки на стороне клиента wininet.dll;

-требует веб-сервера IIS 4.0 (или выше) - или других серверов, поддерживающих эту технологию.

CORBA Полностью независимый от ОС стандарт функционирования приложений в распределенной сети.

Достоинства:

- независимость от ОС;
- позволяет полностью реализовать механизм прозрачности положения сервера;
- возможность выбора между автоматическим и ручным запуском сервера приложений.

Недостатки:

- требуется установка дополнительного программного обеспечения (брокера объектных запросов);
- более сложная установка и настройка сервера приложений для автоматического запуска;
- использование в Delphi COM для реализации CORBA.

Для демонстрации функционирования технологии MIDAS мы воспользуемся любым средством разработки Borland (Delphi 5, 6, 7, C++Builder 5, 6…), которое поддерживает разработку данного типа приложений. Кроме того, нам понадобятся базы данных, к которым мы будем предоставлять доступ нашим клиентам (для нашего первого приложения воспользуемся набором данных, входящим в состав продуктов Borland).

Так как мы создаем демонстрационное приложение, при помощи которого потом будем расширять наши познания MIDAS, создадим для начала простенький сервер, который будет предоставлять нам доступ к одной конкретной таблице. Кроме того, мы создадим клиент для отображения данных из таблицы заданной на сервере.

Создание простенького MIDAS-сервера средствами разработки Borland не предоставляет никакой сложности (как раз тут в полной мере реализована методика разработки приложений без ручного написания кода).

Для начала откроем одно из средств разработки Borland, к примеру Delphi. Далее выберем пункт создания нового приложения и создадим для нашего сервера библиотеку ActiveX. Для этих целей можно также создать приложение (New =>Application).

После этого нам нужно будет еще раз заглянуть в пункт меню New - теперь уже для того, чтоб создать удаленный модуль данных (Remote Data Module).

После того как в нашем проекте появился модуль, перетаскиваем на него главный связующий объект DataSetProvider из палитры Data Access, а также компонент TTable - кроме него могут использоваться любые другие компоненты для работы с БД, TQuery, TStoredProc и т.п. В нашем же случае достаточно будет и одного Ttable.

Теперь настроим наш TTable на работу с некоторой базой данных - посредством задания (или выбора из списка) значений свойств DatabaseName и TableName.

Далее свойство Table1.Active назначаем равным "true" - и переходим к настройке провайдера. Здесь, также прилагая минимум усилий, заполняем свойство DataSet компонента DataSetProvider1 - и регистрируем наш сервер (Run =>Register ActiveX Server).

Всё - на этом разработка сервера приложений закончена. И, как видим, мы вполне обошлись без ввода единой строки кода - все было сделано средой разработки автоматически.

В этот раз для создания клиента выберем при создании приложения пункт Application - и после появления на экране формы примемся перетаскивать на нее необходимые компоненты. Итак, для непосредственной связи с сервером приложений на стороне клиента могут быть использованы следующие компоненты:

- DCOM Connection;
- Socket Connection;
- Web Connection;
- Corba Connection.

Для взаимодействия с сервером все они используют почти одноименные протоколы.

Так как мы пока тренируемся и клиент и сервер у нас находятся на одном компьютере, то для связи воспользуемся первым из перечисленных компонентов. Здесь при настройке DCOMConnection нам понадобится написать от руки только одно свойство - ComputerName (в нашем случае - localhost).

Далее опять беремся за мышь и переходим к заполнению свойств ServerName и ServerGUID.

Сначала из ниспадающего меню при ServerName выбираем наш вариант (если вы еще не занимались разработкой подобных приложений, то в списке будет один только наш проект с названием, подобным тому, что дано на рис.,- оно назначается объекту по шаблону <имя библиотеки ActiveX. имя сервера>). После этого выбора среда разработки сама вставит в поле ServerGUID, необходимый идентификатор. Последнее приготовление компонента DCOMConnection к работе осуществляется его активизацией (DCOMConnection1.Active = true). Теперь, если никаких ошибок не возникло, мы с уверенностью можем сказать, что сервер настроен нормально.

Для дальнейшей работы перетащим с палитры компонентов DataAccess компонент TClientDataSet, который и будет представлением удаленного набора данных на стороне клиента. Для задействования компонента ClientDataSet мы должны установить его свойства RemoteServer и ProviderName (порядок установки значений этих свойств оказывает существенное влияние на дальнейшее поведение ClientDataSet). Так, из ниспадающего меню при свойстве RemoteServer выбираем значение того связующего компонента (если их несколько), который настроен на необходимый сервер. В нашем случае в этом списке будет только один компонент DCOMConnection1, который мы и выбираем.

После этого принимаемся за установку значения свойства ProviderName. Как вы помните, при разработке сервера приложений в RemoteDataModule мы поместили компонент TDataSetProvider. Теперь же - при разработке клиента - выбираем его из ниспадающего списка и заносим в свойство ProviderName.

Теперь нам остается перетащить на форму компонент, который будет отображать полученные данные (например, TDBGrid), и компонент TDataSource. После настройки всех необходимых параметров (DataSource.DataSet, DBGrid.DataSource) мы готовы вывести наши данные на форму. Для этого установим значение свойства Active компонента ClientDataSet в True. Результат - на рис. При этом заметьте: мы еще даже не запустили наше приложение-клиент - но уже удостоверились в его работоспособности.

По большому счету, работа с удаленным набором данных в таких приложениях осуществляется так же, как и у стандартных двухуровневых приложениях БД,- за исключением того, что при обработке данных более правильным будет использование так называемой модели "портфеля".

Модель портфеля основывается на двух методах компонента ClientDataSet:

```
procedure SaveToFile (const FileName: string = ''; Format
TDataPacketFormat=dfBinary);
procedure LoadFromFile (const FileName: string = '');
```

Первый из представленных методов сохраняет полученные данные в локальном файле - в соответствующем формате. Второй, соответственно, загружает сохраненные данные. Если надо работать с двумя таблицами, их необходимо сохранять в двух отдельных файлах (или, чтоб не засорять диск большим количеством файлов, воспользоваться COM-хранилищами и методами ClientDataSet.SaveToStream и ClientDataSet.LoadFromStream).

Сам процесс записи и считывания данных с локального диска чрезвычайно прост, однако для правильной работы с "портфелем" необходимо принять во внимание некоторые моменты

Свойство PacketRecords - одно из важнейших свойств компонента ClientDataSet. Основное его предназначение - установка количества записей, которые могут быть переданы сервером в одном пакете. При создании экземпляра класса TClientDataSet это свойство автоматически принимает значение -1, что означает передачу сразу всех данных в одном пакете. Установка значения свойства PackerRecords в ноль информирует сервер о том, что клиенту необходимы только метаданные базы данных (информация о самой БД, описания таблиц, названия, типы используемых полей и т.п.).

Если же значение свойства больше нуля, то информация передается сервером по запросу несколькими пакетами, в каждом из которых размещается не больше записей. Это удобно во многих случаях - особенно при низкой пропускной способности сети (или в случае связи через модем).

Но установка значения 1 и больше не подходит для работы по "портфельному" методу. Вместо этого при использовании модели "портфеля" мы оставим значение свойства PacketRecords по умолчанию (-1). При этом все останется по-прежнему - кроме того, полный набор данных на сервере будет доступен все время.

С другой стороны, этот подход не очень практичен при использовании больших объемов данных.

Для сохранения одной или более отредактированных записей при использовании удаленного набора данных надо вызвать метод ApplyUpdates:

function ApplyUpdates (MaxErrors: Integer): Integer; virtual;

Присвоение параметру MaxErrors значения проинформирует компонент о том, что сервер не должен останавливать процесс обновления данных при количестве ошибок, меньшем чем MaxErrors. Установка значения -1 говорит о том, что вы не хотите

останавливать процесс обновления при возникновении ошибок. Но в случае возникновения ошибок метод ApplyUpdates возвращает значение, равное их количеству.

После того как вы внесли свои изменения в удаленную БД, вы, возможно, захотите обновить свой набор данных - с тем чтобы посмотреть изменения, внесенные в базу другими пользователями. Для этого используется метод Refresh, входящий в TDataSet. Типичный пример обновления:

```
if ClientDataSet1.ApplyUpdates (-1) = 0 then ClientDataSet1.Refresh;
```

Итак, теперь вы уже знаете, как организовать в своей сети (или даже на одном компьютере) полнофункциональное трехуровневое приложение Клиент-Сервер - и облегчить тем самым последующую настройку и сопровождение своего приложения, а также задействование новых клиентских машин с минимумом затрат рабочего времени и ресурсов аппаратуры. Под конец хотелось бы, с вашего позволения, дать несколько советов для тех, кто сразу же бросился писать код. Не спешите - определитесь сначала с тем, что вы хотите видеть в вашем приложении, и, в зависимости от поставленных задач, выберите приемлемый способ связи и представления данных. Вот несколько советов по выбору протокола передачи данных, если:

- вы в первый раз занимаетесь разработкой приложения такого уровня, ТО можно посоветовать использовать DCOM, установив сервер приложений на и клиента на одной машине;

- ваша БД будет использоваться только с ОС Windows с контроллером домена NT - ТО смело используйте DCOM;

- вы не уверены в платформе или в том, что в сети есть NT-домен, ТО можете воспользоваться протоколом CORBA или сокетами;

- необходимо обеспечить распределение нагрузки на несколько компьютеров и их имена и адреса не известны заранее (или могут изменятся), ТО воспользуйтесь технологией CORBA;

- вы желаете предоставить доступ к данным извне, ТО вам не обойтись без WebConnection. Но здесь следует также позаботится и о веб-сервере, поддерживающем данную технологию;

- вам необходимо в кратчайшие строки организовать многослойную БД и вы не хотите забивать голову внушительными перечнями настроек и правами доступа Corba и DCOM, ТО SocketConnection - как раз то, что вам нужно.

Задание на лабораторную работу

1. Создать сервер приложения, используя компонент DCOM.
2. Реализовать клиента.
3. Осуществить доступ к данным.

Список использованных источников.

1. Адколи А. Велпури Р. ORACLE 9i для Windows.- М.: Лори, 2006. – 498 с.
2. Терьо О. 101 ORACLE 9i. Администрирование баз данных. - М.: Лори, 2005. – 512 с.
3. Луни К. ORACLE 9i Настольная книга администратора. - М.: Лори, 2005. – 748 с.
4. Кайт Т. ORACLE для профессионалов. –СПб.: - Диа Софт Ю П, 2003. – 672 с.
5. Ален К. 101 Oracle PL\SQL. - М.: Лори, 2001. – 350 с.
6. Карпова Т.С. Базы данных: модели, разработка, реализация. СПб.: Питер, 2002. – 304с.
7. Дейт, К. Введение в системы баз данных.- М. :Издательский дом «Вильямс», 2002. – 1072с.

Приложение

Используемые термины и понятия

Экземпляр – это совокупность процессов, разделяющих определенную область памяти и управляющих базой данных.

Табличное пространство – логические части базы данных. Используются для логической группировки данных между собой. Каждое табличное пространство состоит из одного или более файлов.

Блоки данных – наименьшая единица хранения данных в базе данных. Размер блока по умолчанию зависит от операционной системы.

Экстенты – непрерывный набор блоков базы данных.

Сегменты – один или несколько экстентов. Сегмент данных хранит пользовательские данные; индексный сегмент содержит индексы; сегмент отката – хранит информацию отката для возможности возврата к предыдущему состоянию базы данных.

Связь базы данных позволяет определить путь доступа к объекту из удаленной базы данных.

Схема – совокупность объектов, принадлежащих учетной записи пользователя.

Разделы – меньшие таблицы, полученные в результате разбиения больших таблиц, в соответствии с некоторыми диапазонами.

Кластеры содержит таблицы, часто используемые совместно.

Хеш-кластеры используют функции хеширования кластерного ключа строки для определения места ее хранения.

Синонимы указывают на объекты в локальной или удаленной базе данных.

Триггеры – это процедуры, выполняемые при наступлении указанного события.

Учебное издание

ORACLE - СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Методические указания

Составители: ***Логанова Лилия Владимировна,***
Колчин Юрий Владимирович

Технический редактор *Е.А. Симановский*
Редакторская обработка *Т.К. Кретинина*
Корректорская обработка *О.Ю. Дьяченко*
Доверстка *А.В. Ярославцева*

Подписано в печать 11.01.07. Формат 60х84 1/16.
Бумага офсетная. Печать офсетная.
Усл. печ. л. 4,8 . Усл. кр.-отт. 4,9. Печ. л. 5,0 .
Тираж 50 экз. Заказ . ИП-85/2006

Самарский государственный
аэрокосмический университет.
443086 Самара, Московское шоссе, 34.

Изд-во Самарского государственного
аэрокосмического университета.
443086 Самара, Московское шоссе, 34.