МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА
(национальный исследовательский университет)»

Распределенная обработка данных в современных СУБД

Электронные методические указания
к лабораторным работам

САМАРА
2010

Составители: Додонов Михаил Витальевич,
Сопченко Елена Вильевна

Рецензент: Авсиевич А.В.

В пособии приведены методические указания к лабораторным работам по разделу "Распределенная обработка данных в современных СУБД ".

Учебное пособие предназначено для студентов, обучающихся по магистерской программе 010300.68 "Фундаментальная информатика и информационные технологии" и изучающих дисциплину "Распределенная обработка данных в современных СУБД".

Разработано на кафедре программных систем.

Содержание

Введение

Совокупность данных, представленных предметной областью и предназначенных для совместного применения, называется ***базой данных*** (БД).

Распределенная БД - это множество физических баз данных, которые выглядят для пользователя как одна логическая БД.

В большинстве случаев распределенная СУБД состоит из ядра СУБД и набора дополнительных продуктов, покупаемых отдельно, которые обеспечивают работу с распределенной БД. Некоторые фирмы-разработчики СУБД встраивают средства работы с распределенной БД в ядро СУБД. Кроме того, различные фирмы вкладывают разные понятия в термин "распределенная СУБД" и по-разному определяют набор необходимых для такой СУБД функций. Поэтому потенциальным покупателям распределенных СУБД очень непросто сравнивать эти СУБД между собой и делать правильный выбор.

Однако существует некоторый набор функций, которые должны быть присущи каждой распределенной СУБД. Наиболее распространенным описанием этих функций является следующее:

"Распределенная СУБД обеспечивает пользователям доступ к информации независимо от того, какое оборудование и какое прикладное программное обеспечение используется в узлах сети. Пользователи при этом не обязаны знать, где физически размещаются данные и как надо выполнять физический доступ к ним. Распределенная СУБД позволяет выполнять горизонтальное и вертикальное "расщепление" таблиц и помещать данные одной таблицы в различных узлах сети. Запросы к данным распределенной БД формулируются так, как будто база данных локальна. При обработке транзакций и выполнении операций

копирования/восстановления распределенной БД обеспечивается целостность всей БД.

Протоколы управления БД, например, протокол двухфазной фиксации изменений, реализуют механизмы блокировки, обеспечивающие непротиворечивость данных. Средства глобальной оптимизации запросов автоматически выбирают наилучший способ выполнения в локальной или глобальной сети сложных транзакций, а специальные мониторы позволяют получать информацию о параметрах выполняющихся в разных узлах процессов и на основе этой информации выполнять настройку системы с целью обеспечения максимальной производительности работы."

Кроме понятия распределенной базы данных широко используется понятие ***распределенной обработки***.

Распределенный способ обработки данных основан на распределении функций обработки между различными компьютерами, включенными в сеть. Этот способ может быть реализован двумя путями: первый предполагает установку ЭВМ в каждом узле сети (или на каждом уровне системы), при этом обработка данных осуществляется одной или несколькими ЭВМ в зависимости от реальных возможностей системы и ее потребностей на текущий момент времени. Второй путь - размещение большого числа различных процессоров внутри одной системы. Такой путь применяется в системах обработки банковской и финансовой информации, там, где необходима сеть обработки данных (филиалы, отделения и т.д.).

Преимущества распределенного способа: возможность обрабатывать в заданные сроки любой объем данных; высокая степень надежности; сокращение времени и затрат на передачу данных; повышение гибкости систем, упрощение разработки и эксплуатации программного обеспечения и т.д. Высокая эффективность этого способа может быть достигнута за счет разделения функций между компьютерами, входящими в сеть и

специализирующимися на хранении и обработке информации определенного вида.

Примером такого разделения могут служить системы, построенные по архитектуре ***«клиент-сервер»***.

Система разбивается на две части - клиентскую и серверную, которые могут выполняться как в разных узлах сети, так и на одном компьютере («клиент-сервер» без выхода в сеть). Прикладная программа или конечный пользователь взаимодействуют с клиентской частью системы, которая в простейшем случае обеспечивает просто надсетевой интерфейс. Клиентская часть системы при потребности обращается по сети к серверной части. Интерфейс серверной части определен и фиксирован. Поэтому новые клиентские части легко интегрируются в существующую систему.

Основной проблемой систем, основанных на архитектуре "клиент-сервер", является то, что в соответствии с концепцией открытых систем от них требуется мобильность в как можно более широком классе аппаратно-программных решений открытых систем с учетом того, что в разных сетях применяется разная аппаратура и протоколы связи. Попытки создания систем, поддерживающих все возможные протоколы, приводит к их перегрузке сетевыми деталями в ущерб функциональности.

Другая проблема связана с возможностью использования разных представлений данных в разных узлах сети (различная адресация, представление чисел, кодировка символов и т.д). Это особенно существенно для серверов высокого уровня: телекоммуникационных, вычислительных, баз данных.

Общим решением проблемы мобильности систем, основанных на архитектуре "клиент-сервер" является опора на программные пакеты, реализующие протоколы удаленного вызова процедур (RPC - Remote Procedure Call). При использовании таких средств обращение к сервису в удаленном узле выглядит как обычный вызов процедуры. Средства RPC, в

которых, содержится вся информация о специфике аппаратуры локальной сети и сетевых протоколов, переводит вызов в последовательность сетевых взаимодействий. Тем самым, специфика сетевой среды и протоколов скрыта от прикладного программиста.

При вызове удаленной процедуры программы RPC производят преобразование форматов данных клиента в промежуточные машинно-независимые форматы и затем преобразование в форматы данных сервера. При передаче ответных параметров производятся аналогичные преобразования. Если система реализована на основе стандартного пакета RPC, она может быть легко перенесена в любую открытую среду.

Лабораторная работа № 1
Проектирование распределенной базы данных

Цель задания

Изучение методики проектирования распределенных баз данных

Краткие теоретические сведения

Проектирование распределенной базы данных, расположенной в разных узлах сети начинается с разработки общего проекта системы, т.е. построения модели предметной области.

Проектирование модели предметной области

• ***Модель данных*** - используемая при проектировании знаковая система (способ абстрагирования предметной области).

Описание предметной области в терминах выбранной модели данных называют концептуальной схемой предметной области. Концептуальные схемы различаются по уровню абстракции. Наиболее часто применяют модель «сущность-связь», тремя основными конструктивными элементами которой являются *сущность*, *атрибут* и *связь*.

• ***Сущность*** - это обобщённое понятие для обозначения множества однородных объектов предметной области, информацию о которых необходимо собирать и хранить в информационной системе. Сущность определяется своим уникальным именем и перечнем атрибутов, характеризующих свойства сущности.

• ***Атрибут*** - это поименованная характеристика сущности, которая принимает значения из некоторого множества допустимых значений. Атрибуты моделируют свойства сущности.

• ***Связь*** - это обобщённое понятие, предназначенное для обозначения выделенного в предметной области отношения между двумя сущностями.

В любой связи выделяются два конца (в соответствии с существующей парой связываемых сущностей), на каждом из которых указываются имя конца связи, степень конца связи (сколько экземпляров данного типа сущности должно присутствовать в каждом экземпляре данного типа связи), обязательность связи (т. е. любой ли экземпляр данного типа сущности должен участвовать в некотором экземпляре данного типа связи).

Связь представляется в виде ненаправленной линии, соединяющей две сущности или ведущей от сущности к ней же самой. При этом в месте «стыковки» связи с сущностью используются:

- трехточечный вход в прямоугольник сущности, если для этой сущности в связи могут (или должны) использоваться много (*many*) экземпляров сущности;
- одноточечный вход, если в связи может (или должен) участвовать только один экземпляр сущности.

Обязательный конец связи изображается сплошной линией, а необязательный – прерывистой линией.

Связь между сущностями ПРОИЗВОДИТЕЛЬ и ТОВАР, показанная на рисунке 1, связывает производителей разных товаров и товары. Конец связи с именем «изготовлен» позволяет связывать с одним производителем более одного товара, причем каждый товар должен быть связан с каким-либо производителем. Конец связи с именем «производит» показывает, что каждый товар может быть произведен только одним производителем, причем производитель не обязательно должен производить какой-либо товар (например, в текущий момент времени производство может быть временно приостановлено).

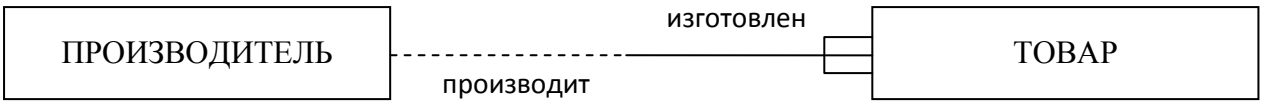

Рисунок 1 - Пример типа связи

Лаконичная устная трактовка изображенной диаграммы состоит в следующем:

- каждый ТОВАР произведен только одним ПРОИЗВОДИТЕЛЕМ;
- каждый ПРОИЗВОДИТЕЛЬ может производить один или более ТОВАРОВ.

Предметная область БД определена, если известны существующие в ней объекты, их свойства и отношения (связи).

При описании той или иной предметной области желательно, чтобы соблюдались следующие требования:

- полнота охвата объектов (сущностей) рассматриваемой области;

- однозначность атрибутов;

- возможность включения новых объектов (сущностей).

Таким образом, проектирование БД начинается с предварительной структуризации предметной области: фиксации объектов (сущностей), свойств этих объектов и виды отношений между объектами.

Информацию о проекте суммируют с использованием графических диаграмм.

Проектирование логической структуры базы данных

В повседневной практике, логическая модель нормализуется до третьей нормальной формы. Нормализация является операцией перемещения атрибутов в подходящие сущности в соответствие с требованиями нормальных форм. Нормализация данных означает проектирование структур данных таким образом, чтобы удалить

избыточность и ограничить несвязанные структуры. Широко используются пять нормальных форм, но на практике многие логические модели приводят только к третьей нормальной форме (НФ).

Пусть задано отношение *R (A1, ... An),* где *(A1, ... An)* – множество атрибутов отношения *R.*

- Схема отношений *R* находится в *1НФ*, если в БД нет одинаковых кортежей, а также значение, определяемое доменом (множеством возможных значений) каждого атрибута является атомарным, то есть значения не являются ни списками, ни множествами простых или сложных значений, и их нельзя использовать по частям.

- Для каждого отношения *R* с приписанным ему некоторым набором функциональных зависимостей (ФЗ) существует вполне определенное множество функциональных зависимостей F, называемое полным.

- Полное множество всех ФЗ, полученное на основе множества F называется замыканием множества F и обозначается F+, причем $F \subseteq F+$.

- Для данного множества функциональных зависимостей *F* и данной функциональной зависимости $X \rightarrow Y$, множество Y называется частично зависимым от X относительно F, если существует $X' \subset X$ такое, что существует зависимость из $X' \rightarrow Y \subset F+$ (также принадлежит F*). В противном случае Y называется полностью зависимым от X.

- Схема отношений R находится в 2 НФ, если она находится в 1 НФ и если в этом отношении каждая позиция, не входящая в ключ (непервичный атрибут), функционально полно зависит от ключа.

- Для данной схемы отношения R подмножества X ($X \subset R$), атрибута $A \in R$ и множества ФЗ F, атрибут A называется транзитивно зависимым от X в R, если существует подмножество $Y \subseteq R$ такое, что $X \rightarrow Y$ (функционально связано), $Y \not\rightarrow X$ (функционально не связано), и $Y \rightarrow A$ относительно F, при этом $A \notin XY$ (не принадлежит ни X, ни Y).

• Схема отношений R находится в 3-ей НФ относительно множества ФЗ F, если она находится в 1-ой НФ и ни один из первичных атрибутов не является транзитивно зависимым от ключей в R.

Ниже приведены простые правила нормализации:

1. Каждый факт в модели должен быть представлен только один раз.
2. Связи многие ко многим должны быть устранены.
3. Атрибуты, независящие от первичного ключа, необходимо размещать в зависимых сущностях.
4. Повторяющиеся атрибуты необходимо размещать в зависимых сущностях.

Проектирование физической структуры базы данных

Классическая реляционная модель данных требует, чтобы данные хранились в так называемых плоских таблицах, у которых каждая ячейка может быть однозначно идентифицирована указанием строки и столбца таблицы. Кроме того, в одном столбце все ячейки должны содержать данные одного простого типа.

Варианты

1. Организация занимается поставками молочной продукции. Организация характеризуется ИНН, наименованием, телефоном, адресом, фамилией директора, фамилией менеджера, ответственного за поставки. Молочная продукция характеризуется названием, % жирности, стоимостью. Потребителями продукции являются другие организации, характеризующиеся различным видом собственности (государственные учреждения, муниципальные учреждения, ООО, ОАО, ИП).

2. Организация занимается поставками мучной продукции. Организация характеризуется ИНН, наименованием, телефоном, адресом, фамилией директора, фамилией менеджера, ответственного за поставки. Мучная продукция характеризуется названием, сортом, стоимостью.

Потребителями продукции являются другие организации, характеризующиеся различным видом собственности (государственные учреждения, муниципальные учреждения, ООО, ОАО, ИП).

3. Организация занимается поставками плодоовощной продукции. Организация характеризуется ИНН, наименованием, телефоном, адресом, фамилией директора, фамилией менеджера, ответственного за поставки. Плодоовощная продукция характеризуется названием, видом, сортом, стоимостью. Потребителями продукции являются другие организации, характеризующиеся различным видом собственности (государственные учреждения, муниципальные учреждения, ООО, ОАО, ИП).

Задание

1. Для заданной предметной области построить ER-модель, выделить сущности, описать атрибуты каждой сущности, установить связи между сущностями.
2. Разработать схему базы данных. При необходимости произвести нормализацию отношений.
3. При построении физической структуры базы данных согласовать другими вариантами название полей, их идентификаторов и типов данных для всех сущностей (общие во всех вариантах поля должны иметь одинаковые идентификаторы).

   Например, обозначить таблицу поставщиков – PROVIDER с атрибутами prov_id, prov_name и т.д., покупателей (потребителей) – CUSTOMER, продукция – PRODUCT и пр.
4. Реализовать полученную схему в среде Oracle на своей рабочей станции (далее рабочие станции будем обозначать WS1, WS2, …, WSn). При создании дать имя базе данных по имени соответствующей рабочей станции (например, WS1, WS2 и WS3).

5. Заполнить таблицы данными (не менее 10 записей в каждой балице базы данных. При этом данные о поставщиках и потребителях не должны совпадать на разных рабочих станциях).
6. Оформить отчет о выполнении лабораторной работы.

Содержание отчета

1. Цель работы.
2. ER-модель предметной области с указанием имен, типов и обязательности каждой связи.
3. Схема отношений базы данных в 3НФ с указанием первичных атрибутов.
4. Структура физических таблиц Oracle.

Контрольные вопросы по лабораторной работе № 1:

1. Дайте определения следующим понятиям: предметная область, модель данных.
2. Этапы проектирования: концептуальное, логическое, физическое проектирование.
3. Что понимается в моделях «сущность-связь» под сущностью, атрибутом, доменом, экземпляром сущности, связью, уникальным идентификатором сущности?
4. Объяснить необходимость нормализации отношений. Отличия 1НФ, 2НФ, 3НФ.
5. Средства Oracle для построения баз данных.

Лабораторная работа № 2
Манипуляция данными в системах распределенных баз данных

Цель задания

Изучение методов связывания объектов и манипуляций данными базы данных, распределенной на нескольких компьютерах сети.

Краткие теоретические сведения

Большинство систем РД состоит из нескольких баз данных, управляемых разными серверами, расположенными в различных местах. Все серверы и клиенты Oracle должны использовать Net8 - сетевое программное средство Oracle для взаимодействия друг с другом по сети. Каждый сервер базы данных в РБД управляет доступом к своей локальной базе данных – за управление системой в целом не отвечает ни один сервер.

Именование объектов

Все сервисы, доступные в сети, должны иметь уникальные имена, чтобы пользователи и приложения знали, как с ним обращаться. Глобальное имя базы данных состоит из двух частей:

- основное имя базы данных, назначаемое ей при создании. Имя базы данных не должно содержать больше восьми символов.
- сетевой домен базы данных, который показывает логическое местонахождение базы данных в сети.

На рисунке 2 представлена сеть баз данных гипотетической компании SALESMENT (продажи). Сеть SALESMENT состоит из трех баз данных WS1, WS2 и WS3. Им соответствуют глобальные имена (имена сервисов) WS1. SALESMENT, WS2. SALESMENT, WS3. SALESMENT.

Для того, чтобы ссылаться на конкретные имена объектов схемы базы данных, не являющейся локальной, нужно дополнить имя объекта глобальным именем базы данных. На рисунке 2 показано, что в каждой из

баз данных WS1, WS2 и WS3 содержится таблица PROVIDER (производитель/поставщик).

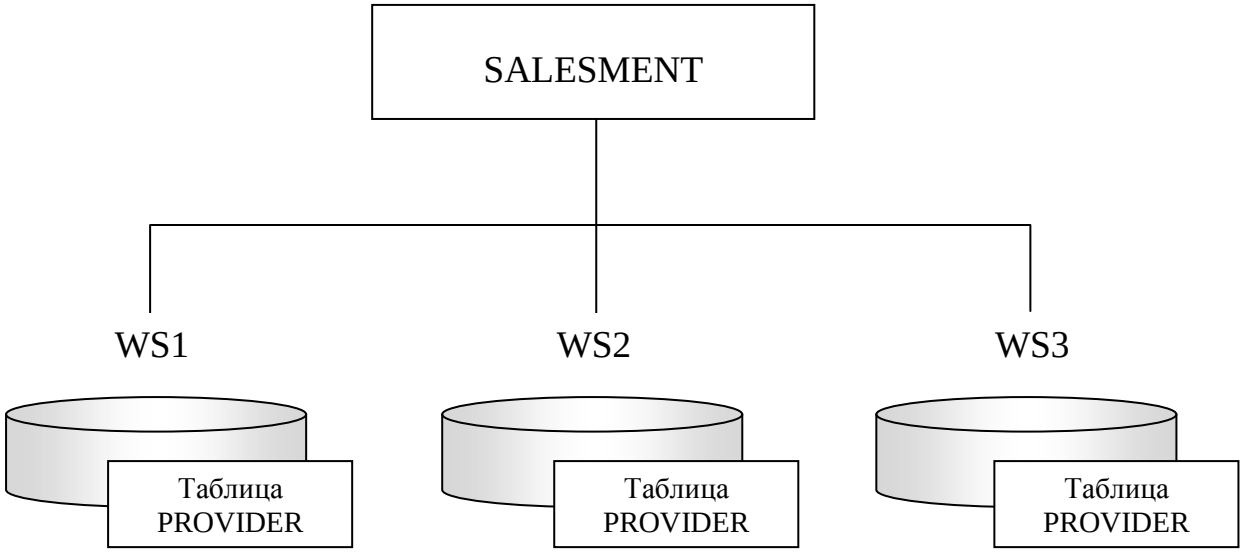

Рисунок 2 – Структура сети баз данных SALESMENT

Если запустить приложение (например, SQL*Plus) и соединиться с базой данных (например, WS2), то можно обратиться как к таблице PROVIDER базы данных WS2, так и к таблице PROVIDER базы данных WS1, идентифицировав этот объект с помощью его составного имени в распределенной базе данных:

```
SELECT * FROM provider@ws1.salesment;
```

Выполняя этот запрос, сервер локальной базы WS2 неявно использует связь баз данных, соединяющую базы данных WS1 и WS2.

Прозрачность распределенной базы данных

Пользоваться указанной системой именования возможно при разработке очень небольших баз данных, т.к. каждый разработчик должен знать текущее местоположение объектов в системе распределенной базы данных.

В Oracle имеется средство, позволяющее сделать работу с этими объектами прозрачной. Это механизм создания синонима, который скрывает физическое место хранения объекта в системе распределенной базы данных:

```
CREATE PUBLIC SYNONIM prov1
 FOR provider@ws1.salesment;
```

После создания общего синонима пользователи локальной базы данных могут ссылаться на удаленную таблицу PROVIDER рабочей станции WS1 как на локальную. Oracle автоматически превращает локальный псевдоним в имя удаленной таблицы и использует для обращения к ней связь базы данных.

```
SELECT * FROM prov1;
```

Другим средством прозрачности могут служить представления. Например, локальное представление PRODUCT указывает на данные, содержащиеся в удаленной таблице PRODUCT рабочей станции WS1.

```
CREATE VIEW product AS
 SELECT * FROM product@ws1.salesment;
```

Установление связей баз данных

Oracle предлагает такую поддержку распределенных баз данных, которая позволяет производить удаленные и распределенные запросы по каналам связи баз данных.

Для того чтобы предоставить доступ к удаленным базам данных в распределенной системе, необходимо установить в локальной баз данных связи баз данных (*database link*). Связь двух баз данных обозначает однонаправленную линию связи между двумя базами данных Oracle (рисунок 3).

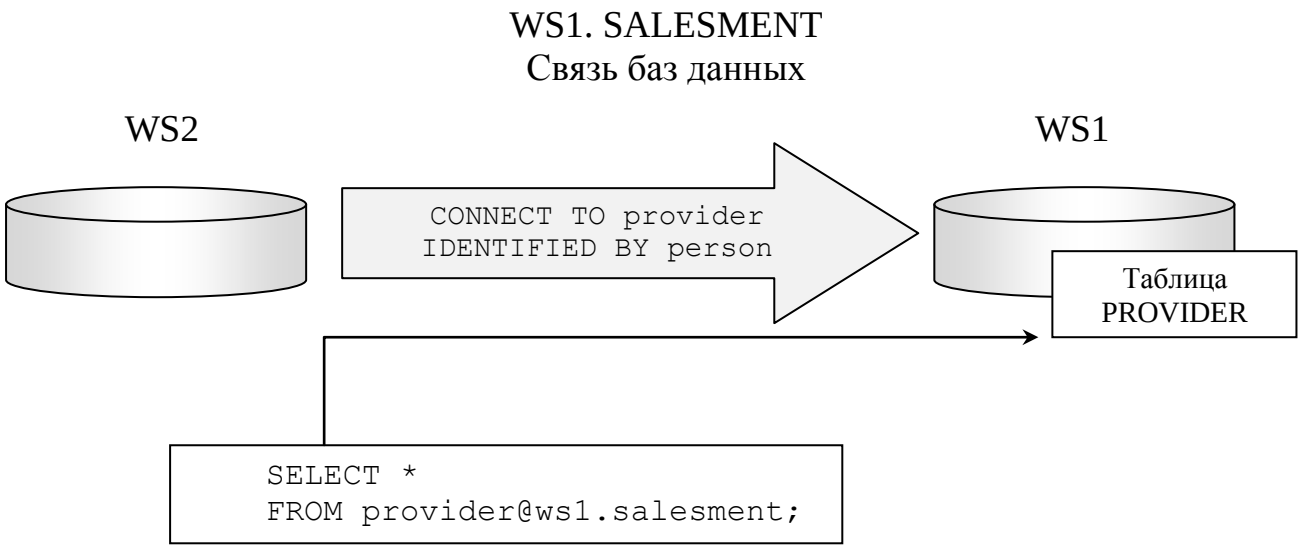

Рисунок 3 – Пример связи баз данных Oracle

Установление связей можно использовать для достижения двоякой цели — обеспечения прозрачности расположения и независимости расположения. Например, с помощью следующего оператора в локальной базе WS2 можно создать связь, описывающую путь к удаленной базе данных WS1. SALESMENT:

```
CREATE DATABASE LINK ws1.salesment;
```

При разработке приложений для работы в распределенной базе данных необходимо выполнять различные операции манипуляций с данными: удаленные запросы, обновления и т.д. Рассмотрим примеры операторов SQL и PL/SQL, позволяющие выполнить эти операции.

Установив связи между базами данных, программа может манипулировать данными на любом из узлов сети.

Удаленные запросы

Удаленный запрос (*remote query*) – это оператор SELECT, считывающий информацию в нескольких удаленных таблицах, находящихся в одном и том же удаленном узле. Например, с помощью следующего удаленного запроса информация считывается в удаленных таблицах PROVIDER и PRODUCT базы данных WS3.

```
SELECT prov.name, p.name, p.price
 FROM provider@ws3.salesment prov, product@ws3.salesment p;
```

Распределенные запросы

Распределенный запрос (*distributed query*) – это оператор SELECT, считывающий информацию из двух или более баз данных. Например, с помощью следующего распределенного запроса информация считывается в локальной таблице PROVIDER и удаленной таблице PRODUCT базы данных WS3.

```
SELECT prov.name, p.name, p.price
 FROM provider prov, product@ws3.salesment p;
```

Удаленное обновление

Удаленное обновление (*remote update*) – это операция обновления, с помощью которой модифицируются данные удаленной таблицы. Например, с помощью следующего оператора UPDATE обновляется строка в таблице PRODUCT удаленной базы данных WS1.

```
UPDATE product@ws1.salesment
    SET prod_price = 200
    WHERE prod_id = 1;
```

Распределенное обновление

Распределенное обновление (*distributed update*) модифицирует данных двух и более серверов. Единственный способ распределенного обновления – создать хранимую процедуру, метод объектных типов и т.д. и включить их в состав две или более операции обновления, каждая из которых обновляет данные в различных базах данных. Например, следующий анонимный блок PL/SQL можно считать распределенным обновлением:

```
BEGIN
UPDATE product
    SET prod_price = 200
    WHERE prod_id = 1;
UPDATE product@ws2.salesment
    SET prod_price = 200
    WHERE prod_id = 1;
UPDATE product@ws3.salesment
    SET prod_price = 200
    WHERE prod_id = 1;
END;
```

Удаленные транзакции

Удаленная транзакция (*remote transaction*) – это транзакция, содержащая один или более удаленных операторов, каждый из которых ссылается на одну и ту же удаленную базу данных. Например, следующая удаленная транзакция обновляет информацию только в базе данных WS1.

```
UPDATE product@ws1.salesment
    SET prod_price = 200
    WHERE prod_id = 1;
UPDATE product@ws1.salesment
    SET prod_price = 300
    WHERE prod_id = 2;
UPDATE product@ws1.salesment
    SET prod_price = 150
    WHERE prod_id = 3;
COMMIT;
```

Распределенные транзакции

Распределенная транзакция (*distributed transaction*) – это транзакция, включающая один или более операторов, обновляющих информацию в двух и более разных базах данных. Например, следующая распределенная транзакция обновляет информацию в нескольких базах данных.

```
UPDATE product
    SET prod_price = 200
    WHERE prod_id = 1;
UPDATE product@ws1.salesment
    SET prod_price = 200
    WHERE prod_id = 1;
UPDATE product@ws3.salesment
    SET prod_price = 150
    WHERE prod_id = 3;
COMMIT;
```

Двухфазная фиксация

При установлении соединений программа может обновлять данные где угодно. Однако она также должна выдать явную команду фиксации в каждый из экземпляров, в который она выдала DML-команды, иначе в какой-то момент времени разные пользователи могут видеть рассогласованное состояние базы данных.

Для транзакции, как единого целого, должна быть выполнена либо полная фиксация (операция завершения), либо полный откат. Чтобы обеспечить соблюдение этого правила для распределенных транзакций, в Oracle применяется специальный алгоритм двухфазного завершения (*two-phase commit mechanism*), который координирует управление транзакциями в сети. Двухфазное завершение необходимо при сетевых и системных сбоях, которые могут прерывать завершение распределенных транзакций.

В общих чертах действие этого механизма выглядит следующим образом: когда выдается команда фиксации, один из экземпляров базы данных, участвующий в транзакции, принимает на себя роль координатора. На этапе первой фазы этот экземпляр выбирает один из

экземпляров в качестве точки фиксации и дает всем остальным задействованным экземплярам (среди которых может быть и он сам, если он не является точкой фиксации) указание приготовиться. После того как получено подтверждение на фиксацию от других экземпляров, точке фиксации предлагается выполнить обычную фиксацию. В зависимости от результата этой фиксации координатор просит все остальные экземпляры либо зафиксировать транзакцию, либо выполнить ее откат.

Конечно, и в этом случае возможны всякого рода сбои. В любой момент могут отказать как отдельные серверы, так и сетевые каналы связи, но, невзирая ни на что, программное обеспечение должно гарантировать целостность данных. В итоге не только возникает значительный трафик сообщений, но и создаются потенциально устойчивые блокировки на уровне блоков.

Механизмы двухфазного завершения являются внутренними процессами серверами баз данных Oracle, участвующих в транзакции. Пользователь должен лишь закончить распределенную транзакцию оператором COMMIT; остальную работу выполняет Oracle.

Закрытие каналов связи

Для "популярных" серверов, т.е. серверов, которые являются объектом множества открытых каналов связи БД, число одновременно открытых Oracle-соединений может стать таким большим, что вызовет чрезмерную подкачку страниц памяти. Последние версии Oracle не только позволяют администратору устанавливать, какое максимальное число каналов связи БД может быть открыто для процесса (через параметр DB_LINKS в файле init.ora), но и разрешают сеансу закрывать канал связи БД, занимающий ресурсы на удаленном сервере. Вот команды для этого примера:

```
ALTER SESSION CLOSE DATABASE LINK ws1.salesment;
ALTER SESSION CLOSE DATABASE LINK ws2.salesment;
ALTER SESSION CLOSE DATABASE LINK ws2.salesment;
```

К сожалению, для установления соединения с Oracle необходимы большие затраты времени центрального процессора на серверной стороне канала, поэтому при сколько-нибудь значительной вероятности того, что этот канал в ближайшем будущем понадобится вновь, вряд ли эффективно его закрывать. "Замораживание" одного-двух мегабайтов памяти на удаленном сервере может оказаться меньшим злом, чем затраты на подключение и отключение, которые придется понести потом. Если же удаленные серверы имеют ограниченные ресурсы и известно, что какие-либо каналы вряд ли еще будут использоваться, то их закрытие позволит увеличить объем доступных ресурсов. Однако при этом также разрушится прозрачность расположения и потребуется более богатое функциональными возможностями приложение, чем то, которое захотят реализовать большинство проектировщиков и программистов.

Варианты увеличения производительности при распределенных соединениях

Распределенные соединения могут создавать проблемы с производительностью все время, пока выполняются операции манипулирования данными в распределенных базах данных.

Если производительность неудовлетворительна, можно рассмотреть следующие варианты:

- На ранних стадиях проекта необходимо постоянное тестирование распределенного соединения на конкретных примерах.
- Создать соединение так, чтобы обеспечить ссылку на одно или несколько представлений и переместить ключевые элементы оптимизации запросов на серверы, где их можно оптимизировать с большей эффективностью.

• Выполнить соединение средствами приложения с использованием удаленного SQL. Эта стратегия может быть эффективной, если удаленные данные можно получить за один запрос или за очень малое число таких запросов.

• Изменить распределение данных, использованных в соединении, перенеся одну или несколько таблиц в другую базу данных (или каким-либо образом реплицировав эти данные). Это решение требует фундаментального изменения в структуре, и для поиска эффективной стратегии распределения, возможно, потребуется перебрать несколько вариантов.

Последнее решение подводит нас к сути проектирования распределенных баз данных:

Стратегия распределения данных должна определяться требованиями к производительности и работоспособности приложения, т.е. данные должны быть распределены по сети таким образом, чтобы максимально соответствовать запросам информационной системы, использующей эти данные.

Задание

1. Установить связи между локальными базами данных WS1, WS2, WS3.
2. Пользователю, выбранному в качестве администратора, создать синонимы для локальных базам данных WS1, WS2, WS3.
3. Создать представления на основе локальных и удаленных таблиц.
4. Проверить работоспособность созданных представлений командой SQL Select.

5. Выполнить удаленный запрос, удаленное обновление, удаленное добавление, удаленное удаление. При выполнении используйте детальный контроль доступа (WHERE).
6. Выполнить распределенный запрос, распределенное обновление, распределенное добавление, распределенное удаление. При выполнении используйте детальный контроль доступа (WHERE).
7. Выполнить удаленную транзакцию.
8. Выполнить распределенную транзакцию.
9. Оформить отчет о выполнении лабораторной работы.

Содержание отчета

1. Цель работы.
2. Операторы SQL, позволяющие создавать синонимы, представления, связи в системе распределенной базы данных.
3. Операторы SQL, позволяющие выполнить удаленные и распределенные манипуляции с данными.
4. Структура физических таблиц Oracle.

Контрольные вопросы по лабораторной работе № 2:

1. Каким образом именуются объекты в распределенной базе данных?
2. Средства Oracle для обеспечения прозрачности имен объектов в распределенной базе данных.
3. Отличие локальной, удаленной и распределенной обработки.
4. Механизм двухфазной фиксации.

Лабораторная работа № 3
Фрагментация базы данных

Цель задания

Изучение методов фрагментации базы данных, распределенной на нескольких компьютерах сети.

Краткие теоретические сведения

Таблицы базы данных могут быть фрагментированы, то есть, разбиты на части, и эти части могут храниться в разных узлах распределенной базы данных. Основанием для фрагментации является повышение производительности. Части таблицы хранятся в тех местах, где к ним происходят наиболее частые обращения - это уменьшает сетевой трафик и сокращает время доступа к данным .

Фрагментация может быть горизонтальной или/и вертикальной.

Горизонтальная фрагментация

Горизонтальная фрагментация - разбиение таблицы по строкам. В этом случае значение в некотором столбце (комбинация значений в столбцах) рассматривается как ключ фрагмента, однозначно определяющий фрагмент, в который входит строка таблицы. Так, например, если таблица PRODUCT представляет список товаров, то данные о каждом товаре могут храниться в узле того отдела, который занимается продажей этого товара. Очевидно, что ключом фрагмента в этом случае должен быть код отдела. Поскольку в базе данных узла фрагмент определяется как таблица, в определение таблицы может (и должно) вводиться ограничение на значение ключа фрагмента.

Вертикальная фрагментация

Вертикальная фрагментация - разбиение таблицы по столбцам. Так, характеристики сортности товара могут храниться в фрагменте на узле

отдела качества, столбцы, определяющие стоимость - в фрагменте на узле отдела продаж и т.д. Вертикальная фрагментация, по сути, представляет собой декомпозицию в схеме данных, когда одна таблица разбивается на две или более таблиц, связанных друг с другом отношением 1:1. Декомпозиция должна быть выполнена без потерь, то есть, набор значений первичных ключей во всех таблицах должен полностью совпадать.

Независимость от фрагментации состоит в том, что ни один из фрагментов не является производным от других фрагментов. Восстановление полной таблицы из фрагментов производится операциями объединения (горизонтальная фрагментация) или/и естественного соединения (вертикальная фрагментация) таблиц-фрагментов. Полная таблица может быть описана как представление, определяемое через указанные операции. При обновлении представления полной таблицы могут возникать проблемы, типичные для обновления представлений: необходимость определения того физического фрагмента, к которому относится обновление. Так, в приведенном выше примере горизонтальной фрагментации перевод товара из одной категории в другую (изменение значения в столбце кода категории) потребует перенесения записи из одного фрагмента в другой.

Фрагментация при распределенных запросах

В распределенных запросах оптимизация еще более важна, чем в локальных. Поскольку выполнение распределенных запросов включает в себя обращение к удаленным узлам и пересылку данных между узлами, минимизация числа таких обращений и объема пересылаемых данных может во много раз уменьшить время выполнения запроса. Выполнение распределенных запросов включает в себя этап глобальной оптимизации, на котором оптимизатор решает, какие данные и с какого узла на какой будут пересылаться, и локальной оптимизации на каждом участвующем в запросе узле.

Например, если на узле A имеется таблица TA, содержащая 100 строк, а на узле B имеется таблица TB, содержащая 106 строк, и запрос требует соединения этих таблиц, то очевидно, что выгоднее переслать таблицу TA на узел B и выполнить соединение на узле B, а не наоборот.

Другой пример. Выше мы рассмотрели пример таблицы PRODUCT (список товаров), фрагментированный горизонтально по продукции определенного вида. Если в описаниях ее фрагментов указано ограничение на код продукта для фрагмента, и оптимизатор "знает" об этом ограничении, то запрос:

```
SELECT * FROM product WHERE prod_cod = 'milk'
```

оптимизатор может локализовать на единственном узле - на том, для которого ограничение ключа фрагмента совпадает с условием, заданном в запросе.

Рассмотрим пример, иллюстрирующий оба типа фрагментации. Имеется таблица PROVIDER (prov_id, prov_name, prov_phone и т.д.), определенная в базе данных на узле WS1. Имеется точно такая же таблица, определенная в базе данных на узле в WS2. Обе таблицы хранят информацию о поставщиках товара. Кроме того, в базе данных на узле в WS3 определена таблица PRODUCT (prod_id, prod_price). Тогда запрос "получить информацию об организациях-поставщиках" может быть сформулирован так:

```
SELECT *
FROM provider@ws1.salesment, provider@ws2.salesment,
ORDER BY prov_id;
```

В то же время запрос "получить информацию о стоимости поставленных продуктов" будет выглядеть следующим образом:

```
SELECT product.prod_id, product.prod_price,
       provider.prov_name
FROM provider@ws1.salesment, provider@ws2.salesment,
     product@ws3.salesment
ORDER BY prod_id;
```

Задание

1. На основе горизонтальной фрагментации:

   а) создать представление, в которое попадут все поставщики продукции всех видов;

   б) создать представление с указанием всех атрибутов, в которое попадут все потребители продукции всех видов;

   b) создать представление, содержащее список всех продуктов всех производителей с указанием кода (артикула), наименования и стоимости;

2. На основе вертикальной фрагментации:

   а) выделить столбцы, отвечающие за характеристику товара (например, сортность) в отдельную таблицу и разместить ее на рабочей станции WS1;

   б) выделить столбцы (код, наименование, стоимость) в отдельную таблицу и разместить ее на рабочей станции WS2;

   в) с рабочей станции WS3 выполнить распределенный запрос и получить представление из таблиц, полученных ранее (см. пункты а) и б) задания 2), содержащее информацию о товарах 1 сорта, стоимость которых отличается от средней не более, чем на 10%;

3. Оформить отчет о выполнении лабораторной работы.

Контрольные вопросы по лабораторной работе №3:

1. Дайте определения следующим понятиям: горизонтальная, вертикальная фрагментация.

2. Какой из видов фрагментации является наиболее эффективным в данной базе данных?
3. Для Вашей базы данных приведите примеры возможной горизонтальной и вертикальной фрагментации.

Лабораторная работа №4
Аутентификация и управление пользователями в СУБД Oracle

Цель задания

Целью выполнения лабораторной работы является обучение методам и средствам аутентификации и управления пользователями в СУБД *Oracle*.

Краткие теоретические сведения

Санкционированный (разрешенный) доступ к ресурсам является одним из важнейших механизмов обеспечения безопасности в распределенных базах данных.

Доступ к ресурсам предусматривает выполнение трех процедур: идентификации, аутентификации и авторизации.

Идентификация - назначение субъектам, объектам, процессам уникальных имен (идентификаторов).

Аутентификация – проверка и подтверждение подлинности идентифицированного субъекта, объекта, процесса.

Авторизация – это определение набора возможных операций с данными, которые может осуществлять пользователь.

В СУБД *Oracle* реализована поддержка принципа безопасности по умолчанию или принципа минимальных привилегий. Суть принципа состоит в том, что пользователь может получить доступ к объекту СУБД

(например, таблице или представлению) или выполнить определенные действия в системе (например, создать новую таблицу или нового пользователя) только если это явно разрешено. Поэтому пользователь, успешно прошедший аутентификацию, по сути ничего не может делать до тех пор, пока уполномоченный администратор не определит перечень возможных для данного пользователя операций. В частности, корректно созданный пользователь после успешной аутентификации (ввода правильного пароля) не сможет даже присоединиться к серверу баз данных, т.е. успешно выполнить команду *CONNECT*.

Каждый пользователь СУБД *Oracle* должен иметь специальный идентификатор: имя или точку входа. Создание нового идентификатора осуществляется уполномоченным пользователем или администратором выполнением предложения *CREATE USER.*

С позиций системы источники, предъявившие идентификатор, неразличимы. То есть, хотя пользователем может быть как реальный человек, сидящий за терминалом, так и прикладной процесс, для системы оба объекта тождественны.

Когда пользователь пытается подключиться к СУБД, сервер СУБД *Oracle* обеспечивает выполнение стандартной процедуры подтверждения подлинности или аутентификации. Обычно для подтверждения подлинности пользователь должен ввести пароль.

СУБД *Oracle* предлагает четыре метода аутентификации – аутентификация на уровне ОС, аутентификация на уровне сети, аутентификация на уровне СУБД и многоуровневая аутентификация.

Система аутентификации пользователей в СУБД *Oracle* может состоять из любого набора предлагаемых СУБД *Oracle* методов аутентификации.

Внешняя аутентификация – аутентификация на уровне ОС (*Authentication by the Operating System*) и аутентификация на уровне сети (*Authentication by the Network*) – предполагают использование средств

операционной системы или сетевых средств аутентификации. Тем самым контроль выносится за пределы средств управления паролями и идентификации пользователя СУБД *Oracle*, хотя пользователь будет по-прежнему опознаваться СУБД. Для этого типа регистрации пароль СУБД не требуется. При вынесении процедуры аутентификации на уровень ОС, сервер определяет идентификатор пользователя из информации сеанса его работы с операционной системой и на основе этой информации разрешает или запрещает пользователю подключение.

Чтобы использовать эту опцию, необходимо установить в файле СУБД *init.ora* параметр *OS_AUTHENT_PREFIX*. Это укажет СУБД *Oracle,* что пользователь, имя которого имеет тот же префикс, должен рассматриваться как подлежащий внешней идентификации. Например, если для параметра *OS_AUTHENT_PREFIX* установлено значение *ops*$ и имеются два пользователя – *ops*$*Ivanov* и *Petrov*, то для системы СУБД *Oracle* не нужен пароль от пользователя *ops*$*Ivanov*, но нужен – от *Petrov.* Данным параметром может быть установлен любой желаемый префикс (включая нулевую строку). В таком случае указывается пустое значение в двойных кавычках. При этом для параметра *REMOTE_OS_AUTHENT* в файле *init*.ora должно быть установлено значение *true* (значение по умолчанию – *false*), чтобы система СУБД *Oracle* могла использовать имя пользователя из незащищенного соединения.

При использовании аутентификации на уровне СУБД (*Authentication by* СУБД *Oracle Database*), имена и пароли этом хранятся не в файлах операционной системы, а в СУБД *Oracle*. Применение этого метода повышает степень защиты данных, так как универсальный сервер данных СУБД *Oracle* обеспечивает более детализированное управление доступом, чем операционная система. Администратор конфигурирует именную область, которая состоит или из строк связей с удаленными базами данных, или из *SID* СУБД *Oracle* (идентификатор локальной СУБД *Oracle*). Кроме того, такая область может содержать роль из СУБД,

которая доступна только пользователям базы, имеющим привилегию присваивать себе аутентификационную роль.

Идентификация в СУБД используется при первичной регистрации (создании учетной записи) пользователя и указании пароля. Этот подход вполне удовлетворителен для небольших групп пользователей и в тех случаях, когда отсутствуют другие средства обеспечения безопасности. Для других типов идентификации вместо пароля необходимо использовать зарезервированное слово *external* (внешний).

Если в СУБД *Oracle* используется идентификация на уровне СУБД, система предоставляет возможность управления паролями.

Многоуровневой (*Multitier Authentication*), или промежуточной, аутентификацией называется регистрация программного обеспечения промежуточного уровня от своего имени для выполнения в СУБД каких-либо действий по поручению пользователя. Это позволяет создавать промежуточные приложения, использующие собственную схему аутентификации, например, с помощью сертификатов *X*509 или другого процесса однократной регистрации и регистрироваться от имени пользователя, не зная его пароля в СУБД. Хотя регистрация выполнена не от имени пользователя, а от имени промежуточного ПО, для СУБД он зарегистрирован.

Многоуровневая аутентификация обеспечивает одноразовую аутентификацию – на сервере приложений – и обеспечивает доступ сервера приложений ко всем необходимым базам данных от имени пользователя, не передавая пароли для каждой СУБД.

Соответствующий оператор *ALTER USER* имеет следующий базовый синтаксис:

ALTER USER <имя пользователя> *GRANT CONNECT THROUGH*

<промежуточный пользователь><, промежуточный пользователь>...

Это дает возможность пользователям, перечисленным в списке промежуточных пользователей, подключаться от имени указанного после

ALTER USER пользователя. По умолчанию для этих пользователей будут устанавливаться все роли данного пользователя. Другая разновидность этого оператора:

ALTER USER <имя пользователя> *GRANT CONNECT THROUGH*

<промежуточный пользователь> *WITH NONE*;

позволяет промежуточной учетной записи подключаться от имени указанного пользователя, но только с ее базовыми привилегиями – роли включаться не будут. Кроме того, можно использовать:

ALTER USER <имя пользователя> *Grant Connect Through*

<промежуточный пользователь> *ROLE* имя_роли,имя_роли,...

или:

ALTER USER <имя пользователя> *Grant Connect Through*

<промежуточный пользователь> *ROLE ALL EXCEPT*
имя_роли,имя_роли,...

Два представленных выше оператора дают промежуточной учетной записи возможность подключаться в качестве пользователя, но при этом включены будут только определенные роли. Необязательно давать учетной записи сервера приложений все привилегии – достаточно предоставить роли, необходимые для выполнения его функций. По умолчанию сервер СУБД *Oracle* пытается включить все стандартные роли пользователя и роли *PUBLIC*. Вполне допустимо разрешить серверу приложений использовать только роль *HR* данного пользователя, и никакие другие прикладные роли этого пользователя.

Разумеется, можно и отобрать соответствующую привилегию:

ALTER USER <имя пользователя> *REVOKE CONNECT THROUGH*
<промежуточный пользователь><,промежуточный пользователь>...

Есть административное представление, *PROXY_USERS*, которое можно использовать для получения информации обо всех промежуточных учетных записях.

Синтаксис оператора *AUDIT* позволяет настроить аудит действий, выполняемых указанными промежуточными пользователями от имени некоторых или всех учетных записей:

AUDIT <действие> *BY* <промежуточный пользователь>,

<промежуточный пользователь> ... *ON BEHALF OF* <клиент>,<клиент>...;

или:

AUDIT <действие> *BY* <промежуточный пользователь>,

<промежуточный пользователь> *ON BEHALF OF ANY*;

Для **администраторов СУБД** требуется более надежная схема опознания, соответствующая привилегированному характеру их задач (например, закрытие и запуск СУБД). Дополнительное опознание может быть выполнено с помощью операционной системы и (или) файла пароля.

Если операционная система предоставляет способ объединения пользователей в группы (это возможно в таких операционных системах, как *Unix* и *NT*), то в документации к СУБД *Oracle* рекомендуется объединить администраторов СУБД в специальную группу. Это позволяет СУБД *Oracle* дополнительно проверять с помощью идентификатора группы, является ли пользователь администратором СУБД.

Файл пароля для администраторов СУБД необязателен и может быть установлен с помощью утилиты *ORAPWD*. Файл пароля ограничивает привилегии администрирования только тех пользователей, которые знают пароль и имеют специальные роли – *SYSOPER* и *SYSDBA.*

Роль *SYSOPER* предоставляет возможность выполнять операторы *STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVE LOG* и *RECOVER*, а также включает привилегию *RESTRICTED SESSION*.

Роль *SYSDBA* включаются все системные привилегии с помощью опции *ADMIN OPTION*, а системная привилегия *SYSOPER* допускает

привилегию *CREATE DATABASE* и привилегию восстановления через определенные промежутки времени.

Задание

1. Изучить возможности аутентификации и управления пользователями с помощью запуска нескольких сеансов соединения с СУБД (например, *SQL*Plus*). Один сеанс работает от имени администратора СУБД (*SYSTEM*), другие – от имени пользователей, созданных администратором (на одном компьютере могут быть запущены несколько сеансов).
2. Создать пользователя *Ivanov* с паролем *SKY*. Обеспечить, чтобы объекты и временные сегменты, создаваемые пользователем *Ivanov*, не принадлежали табличному пространству *SYSTEM*. Обеспечить также пользователю *Ivanov* доступ к табличным пространствам *DATA*01 и *INDX*01 и возможность использования в них пространства размером до одного мегабайта для создания своих объектов. Для этого: назначить пользователю временное табличное пространство, табличное пространство по умолчанию и указать квоты на использование табличных пространств *DATA*01 и *INDX*01. Например:

   SQL> CREATE USER Ivanov
   2> *IDENTIFIED BY SKY*
   3> *DEFAULT TABLESPACE data*01
   4> *TEMPORARY TABLESPACE temp*
   5> *QUOTA* 1M *ON data*01
   6> *QUOTA* 1M *ON indx*01;
   SQL> GRANT create session TO Ivanov;
3. Создать пользователя *Petrov* с паролем *OCEAN*. Обеспечить, чтобы в табличном пространстве *SYSTEM* не было объектов и сегментов сортировки, создаваемых пользователем *Petrov*. Например:

SQL> CREATE USER Petrov

2> *IDENTIFIED BY OCEAN*

3> *DEFAULT TABLESPACE data*01

4> *TEMPORARY TABLESPACE temp*;

4. Скопировать таблицу *ORDERS* из схемы *SYSTEM* в схему пользователя *Petrov*. Прежде чем пользователь *Petrov* сможет создавать объекты в своей схеме, ему необходимо предоставить квоту на его табличное пространство по умолчанию.

   *SQL> ALTER USER Petrov QUOTA UNLIMITED ON data*01,

   SQL> CREATE TABLE Petrov.orders AS

   2> *SELECT * FROM system.orders*;

5. Вывести на экран информацию словаря данных о пользователях *Ivanov* и *Petrov*. Эту информацию можно получить, выполнив запрос к представлению *DBA_USERS*.

   SQL> SELECT username, default_tablespace,

   2> *temporary_tablespace*

   3> *FROM dba_users*

   4> *WHERE username IN* ('*Ivanov*', '*Petrov*');

   USERNAME DEFAULT_TABLESP TEMPORARY_TABLE

   ---------- ---------- ----------

   Petrov *DATA*01 *TEMP*

   Ivanov *DATA*01 *TEMP*

   2 *rows selected*.

6. Выведите на экран информацию словаря данных об объеме пространства в табличных пространствах, которое может использовать *Ivanov*. Эту информацию можно получить, выполнив запрос к представлению *DBA_TS_QUOTAS*.

   *SQL> SELECT * FROM dba_ts_quotas WHERE username = 'Ivanov';*

   TABLESPACE_N USERNAME BYTES MAX_BYT BLOCKS MAX_BLO

   *INDX*01 *Ivanov* 0 1048576 0 512

*DATA*01 *Ivanov* 0 1048576 0 512

2 *rows selected*.

7. Как пользователь *Ivanov* произвести попытку изменить назначенное ему временное табличное пространство.

   SQL> CONNECT Ivanov/*SKY*;

   Connected.

   SQL> ALTER USER Ivanov

   2> *TEMPORARY TABLESPACE data*01;

   ALTER USER Ivanov

   ORA-01031: *insufficient privileges*

8. Как пользователь *Ivanov* изменить свой пароль на *SAM*.

   SQL> CONNECT Ivanov/*SKY*;

   Connected.

   SQL> ALTER USER Ivanov

   2> *IDENTIFIED BY sam*;

9. Как пользователь *SYSTEM* отменить для пользователя *Ivanov* квоту на его табличное пространство по умолчанию.

   SQL> CONNECT system/*manager*

   Connected.

   SQL> ALTER USER Ivanov QUOTA 0 *ON data*01;

10. Удалить пользователя СУБД *Petrov*. Так как пользователь *Petrov* является владельцем таблиц, нужно использовать режим *CASCADE*.

    SQL> DROP USER Petrov CASCADE;

11. Пользователь *Ivanov* забыл свой пароль. Назначить ему пароль *OLINK* и потребовать, чтобы *Ivanov* изменил пароль при следующем входе в систему.

    SQL> ALTER USER Ivanov

    2> *IDENTIFIED BY olink*

    3> *PASSWORD EXPIRE*;

12. Перехватить на рабочей станции *WS*2 данные, переданные при аутентификации пользователя. Попытаться найти в перехвате параметры аутентификации. Результаты перехвата привести в отчете.
13. Оформить отчет о выполнении лабораторной работы.

Содержание отчета о выполненной работе

1. Цель работы.
2. Описание комплекса программно-технических средств, использованного для выполнения лабораторной работы.
3. Результаты выполнения команд языка *SQL*.
4. Выводы по результатам выполнения лабораторной работы.

Контрольные вопросы по лабораторной работе № 4:

1. Дать определения следующим понятиям: идентификация, аутентификация, авторизация.
2. Описать 4 метода аутентификации.
3. Объяснить отличия пользователей и администраторов базы данных при авторизации.

Список использованных источников

1. Технологии и средства консолидации информации: *Учебное пособие. Деревянко А.С., Солощук М.Н. - Харьков: НТУ "ХПИ", 2008. - 432с.*
2. Организация баз данных. 1 часть*: Курс лекций / Е.В. Сопченко, К.А. Кудрин.* Самарский гос. аэрокосмический ун-т. Самара, 2000, 71 с.
3. Сергей Кузнецов. Базы данных. Вводный курс. *www.cityforum.ru*
4. Сергей Кузнецов. Основы современных баз данных. *www.cityforum.ru*
5. Дейт К.Д. Введение в системы баз данных, 6-е издание. -М: Вильямс. 1999 г. -848 с.
6. Бобровски С. Oracle 8. Архитектура. – М: Издательство «Лори», 1998, 210 с.
7. Методические рекомендации к выполнению лабораторных работ по дисциплине «Серверные системы управления базами данных» для студентов специальности 230102 «Автоматизированные системы обработки информации и управления» всех форм обучения /*Сост.: М.В. Додонов, А.Ю. Павлов*. –Самара: СамГУПС, 2007. – 16 стр.