Содержание

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).

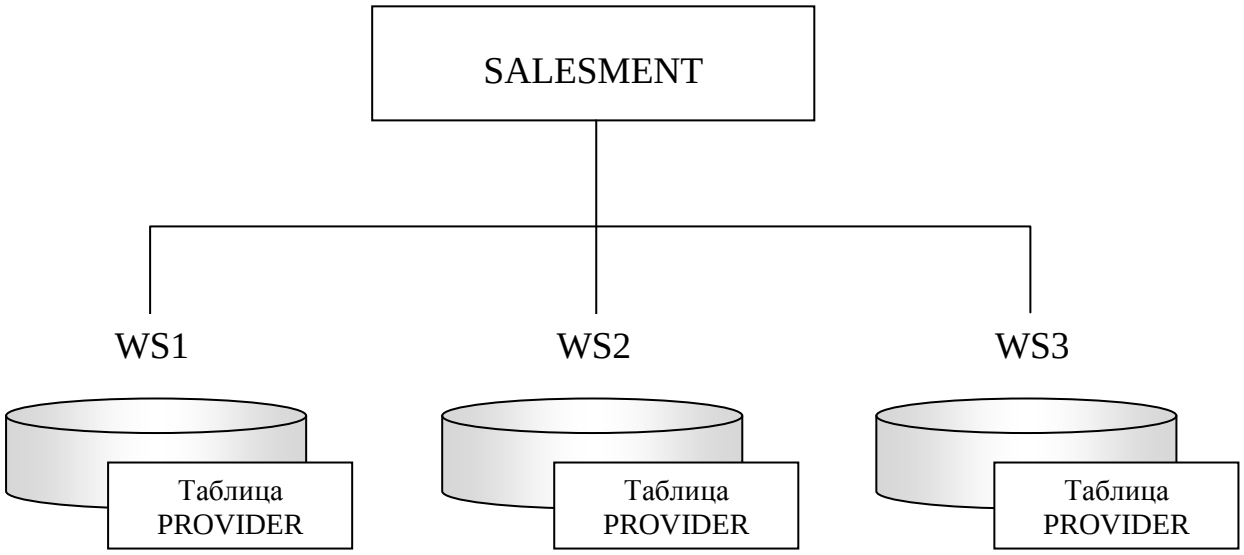

Рисунок 2 – Структура сети баз данных SALESMENT

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).

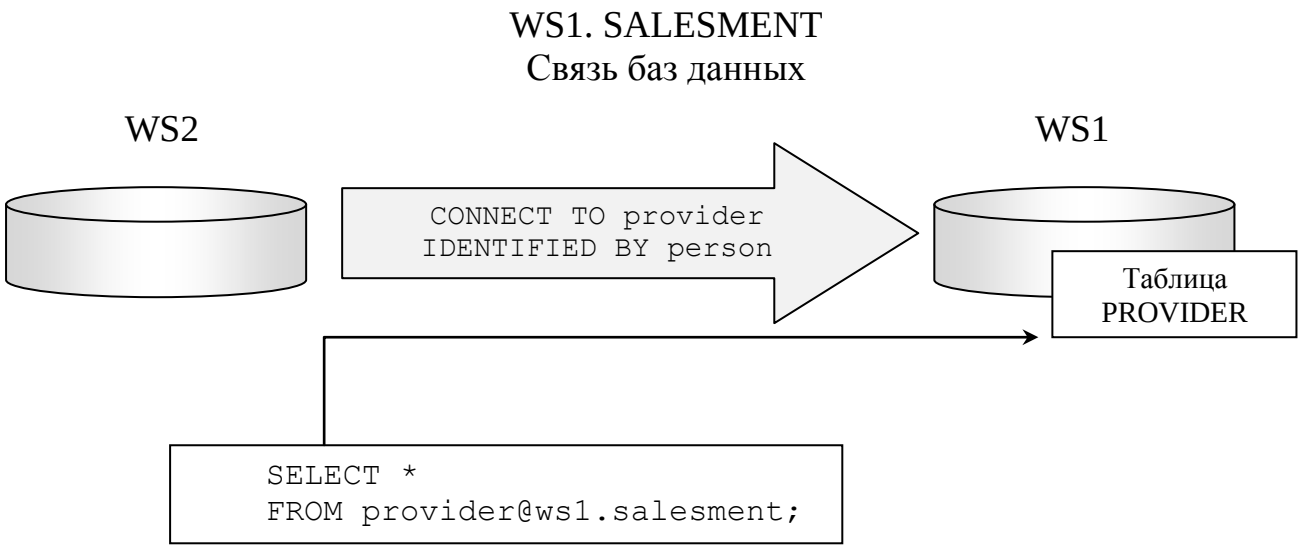

Рисунок 3 – Пример связи баз данных Oracle

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.

Список дополнительной литературы

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.