МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ВОЛОГОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра автоматики и вычислительной техники

РАСПРЕДЕЛЕННЫЕ ФАЙЛОВЫЕ СИСТЕМЫ, БАЗЫ И БАНКИ ДАННЫХ

Методические указания к лабораторным работам. Часть 1

Факультет электроэнергетический

Направления подготовки магистров:
09.04.01 – Информатика и вычислительная техника
09.04.04 – Программная инженерия

Вологда
2014

УДК 621.374.1

Распределенные файловые системы, базы и банки данных: методические указания к лабораторным работам. Часть 1. - Вологда: ВоГУ, 2014. - 40 с.

Приводятся подробные пояснения к первым четырем работам лабораторного практикума, содержащие теоретические сведения и указания, необходимые для успешного выполнения лабораторных работ, формирования и развития профессиональных компетенций в области распределенного хранения и обработки данных.

Утверждено редакционно-издательским советом ВоГУ

Составители: С.Ю. Ржеуцкая, канд. техн. наук, доцент
И.Н. Вахрушев, ассистент

Рецензент А.М. Водовозов, канд. техн. наук, профессор
зав. кафедрой УВС

Подписано в печать 30.10.2014. Усл. печ. л. 2,5. Тираж экз.
Печать офсетная. Бумага офисная. Заказ № _____

Отпечатано: РИО ВоГУ, г. Вологда, ул. С. Орлова, 6

Введение

Лабораторный практикум предназначен для ознакомления магистрантов с наиболее важными практическими аспектами создания и администрирования распределенных файловых систем и баз данных, организации эффективного доступа к данным. Данные методические указания содержат подробные пояснения по выполнению четырех лабораторных работ, каждая из которых позволяет освоить определенные темы лекционного курса и сформировать (развить) профессиональные компетенции в области распределенного хранения и обработки данных.

В дополнение к данным методическим указаниям следует использовать рабочую программу дисциплины, учебное пособие «Базы данных. Язык SQL» и дистанционный практикум по языкам SQL и PL/SQL, установленный на сервере кафедры АВТ и доступный по адресу http://atpp.vstu.edu.ru/cgi-bin/arh_problems.pl.

1. Лабораторная работа №1. Знакомство с основными возможностями распределенной файловой системы DFS

Цель работы: *получение базовых знаний, умений и навыков работы с распределенной файловой системой DFS (Distributed File System).*

1.1 Пояснения к выполнению лабораторной работы

Файловый сервер и файловый сервис. Ключевым компонентом любой распределенной системы является файловая система. Как и в централизованных системах, в распределенной системе функцией файловой системы является хранение программ и данных и предоставление доступа к ним по мере необходимости. Файловая система поддерживается одной или более машинами, называемыми файл-серверами. Файл-серверы перехватывают запросы на чтение или запись файлов, поступающие от других машин (не серверов). Эти машины называются клиентами. Каждый посланный запрос проверяется и выполняется, а ответ отсылается обратно. Файл-серверы обычно содержат иерархические файловые системы, каждая из которых имеет корневой каталог и каталоги более низких уровней. Рабочая станция может подсоединять и монтировать эти файловые системы к

своим локальным файловым системам. При этом монтируемые файловые системы остаются на серверах.

Файловый сервис - это описание функций, которые файловая система предлагает своим пользователям. Оно включает имеющиеся примитивы, их параметры и функции, которые они выполняют, т.е. определяет интерфейс файловой системы с клиентами.

Файловый сервер - это процесс, который выполняется на отдельной машине и помогает реализовывать файловый сервис. В системе может быть один файловый сервер или несколько, но в хорошо организованной распределенной системе пользователи не знают, как реализована файловая система. В частности, они не знают количество файловых серверов, их месторасположение и функции. Более того, пользователи даже не должны знать, что файловый сервис является распределенным.

Так как обычно файловый сервер - это просто пользовательский процесс (или иногда процесс ядра), выполняющийся на некоторой машине, в системе может быть несколько файловых серверов, каждый из которых предлагает различный файловый сервис.

Файловый сервис в распределенных файловых системах (как и в централизованных) имеет две функционально различные части: собственно файловый сервис и сервис каталогов. Первый имеет дело с операциями над отдельными файлами, такими, как чтение, запись или добавление, а второй - с созданием каталогов и управлением ими, добавлением и удалением файлов из каталогов и т.п.

Две главные цели, которые преследуют все распределенные файловые системы:

- Сетевая прозрачность - обеспечить те же самые возможности доступа к файлам, распределенным по сети, которые имеются в системах разделения времени на централизованных ЭВМ.
- Высокая доступность - ошибки систем или осуществление операций копирования и сопровождения не должны приводить к недоступности файлов.

Distributed File System (DFS). Это компонент Microsoft Windows, использующийся для упрощения доступа и управления файлами, физически распределёнными по сети. При её использовании файлы, распределённые по серверам, представляются пользователям находящимися в одном месте.

DFS может быть реализована двумя способами:

- распределенная файловая система с изолированным корнем;
- доменная распределенная файловая система.

Основные понятия DFS

Узловой сервер – сервер домена, на котором располагается корень DFS. Можно реплицировать корень DFS, создав корневые целевые папки на других серверах домена. Это обеспечит доступ к файлам при отключении узлового сервера.

Пространство имен (или корень DFS) – особое имя, используемое пользователями для доступа к файловым ресурсам.

Целевая папка – общедоступная папка на файловом сервере, к которой происходит доступ.

Папка – ссылка в пространстве имён, указывающая на целевую папку.

Преимущества использования DFS

Простой доступ к файлам. Пользователи используют единый сетевой ресурс для доступа к файлам, даже если эти файлы физически находятся на разных серверах.

Доступность. Доменные DFS обеспечивают пользователям доступ к файлам двумя способами:

- во-первых, операционная система Windows Server 2008 автоматически публикует топологию DFS в Active Directory. Благодаря этому, пространство имен DFS всегда видимо для пользователей всех серверов в домене;
- во-вторых, администратор может реплицировать корень DFS и целевые папки. Репликация означает дублирование корней и целевых папок DFS на нескольких серверах домена. При этом пользователи всегда могут получить доступ к своим файлам даже в случае недоступности физического сервера, на котором они находятся.

Эффективная загрузка сервера. Корень DFS может поддерживать несколько целевых папок DFS, которые физически распределены по сети. Например, это полезно, если есть файл, который активно используется пользователями. Пользователи будут обращаться к этому файлу не на одном сервере, сильно загружая его, а к файлу, распределенному системой DFS по разным серверам.

Безопасность файлов и папок. Поскольку общие ресурсы, управляемые DFS, используют стандартные разрешения NTFS и разрешения общего доступа к файлам, можно использовать существующие группы безопасности и учетные записи пользователей.

1.2 Порядок выполнения лабораторной работы

Для выполнения лабораторной работы на компьютерах вашей лаборатории были предварительно сформированы две виртуальные машины с установленной операционной системой Windows Server 2008. Одна из них сконфигурирована как контроллер домена с именем Dcsrvl, другая – рабочий компьютер сети (член домена) с именем Boston. Вся работа выполняется в данной виртуальной среде. Основные задачи работы –- развернуть файловую систему DFS (Distributed File System), познакомиться с её ключевыми возможностями, научиться добавлять общие папки.

Задание 1. *Установка DFS в операционной системе Windows Server 2008*

Систему DFS можно установить при добавлении роли сервера Файловые службы (File Services) в Мастере добавления ролей (Add Roles Wizard) или позже, в Диспетчере сервера (Server Manager), щелкнув правой кнопкой мыши узел Роли\Файловые службы (Roles\File Services) и применив команду Добавить службы ролей (Add Role Services). Какой бы способ вы ни выбрали, выполните на страницах мастера следующее.

1. На странице Пространство имен DFS (DFS Namespaces) укажите, следует ли создать именное пространство. Щелкните кнопку Далее (Next).

2. Если откроется страница Тип пространства имен (Namespace Type), выберите, следует ли использовать именное пространство домена (для сред Active Directory) или выбрать автономное пространство имен (для рабочих групп). Если на всех DFS-серверах пространства имен используется система Windows Server 2008, включите режим Windows Server 2008. Щелкните кнопку Далее (Next).

3. Если откроется страница Конфигурация пространства имен (Namespace Configuration), можно щелкнуть кнопку Добавить (Add), чтобы добавить папки. Это можно сделать и потом, с помощью оснастки Управление DFS (DFS Management). Щелкните кнопку Далее (Next). Если вы не создаете пространство имен DFS и не добавляете папки, можете сделать это позже с помощью консоли Управление DFS (DFS Management) в Диспетчере сервера (Server Manager).

Задание 2. *Создание прост ранст ва имен DFS*

Именное пространство DFS формирует корень общих папок в организации. Чтобы создать именное пространство DFS, выполните следующие действия.

1. В Диспетчере сервера (Server Manager) откройте и щелкните правой кнопкой мыши узел Роли\Файловые службы\Управление DFS\npocTpaHCTBa имен (Roles\File Services\DFS Management\ Namespaces) и примените команду Создать пространство имен (New Namespace). Запустится Мастер создания нового пространства имен (New Namespace Wizard).

2. На странице Сервер пространства имен (Namespace Server) введите имя сервера, который будет управлять именным пространством. С целью обеспечения избыточности позже можно добавить серверы для управления именным пространством. Пользователи не будут ссылаться на имя сервера при получении доступа к именному пространству DFS. Щелкните кнопку Далее (Next).

3. На странице Имя и параметры пространства имен (Namespace Name And Settings) введите имя. Оно будет применяться при получении пользователями доступа к именному пространству DFS, например, так: \имя_домена\имя_пространства\. Щелкните кнопку Изменить (Edit Settings), чтобы настроить разрешения именного пространства. Щелкните кнопку Далее (Next).

4. На странице Тип пространства имен (Namespace Type) выберите создание доменного пространства имен или автономного именного пространства. Доменные пространства имен используют в качестве корня имя домена Active Directory, а автономные именные пространства – имя сервера. Щелкните кнопку Далее (Next).

5. На странице Просмотр настроек и создание пространства имен (Review Settings And Create Namespace) щелкните кнопку Создать (Create).

6. На странице Подтверждение (Confirmation) щелкните кнопку Закрыть.

Параметры созданного пространства имен можно изменить, щелкнув по пространству правой кнопкой мыши и применив команду Свойства (Properties).

Диалоговое окно свойств именного пространства содержит три вкладки.

Общие (General) На этой вкладке можно ввести описание именного пространства.

Ссылки (Referrals) При доступе к корню именного пространства или папкам клиент получает ссылку с контроллера домена. Клиенты всегда пытаются получить доступ к первому целевому компьютеру в списке ссылок, а если компьютер не отвечает, переходят к следующему в списке. На этой вкладке можно управлять сортировкой конечных ресурсов в списке. В раскрывающемся списке Метод сортировки (Ordering Method) выберите Случайный порядок (Random Order), чтобы распределять ссылки среди всех конечных объектов (конечные объекты на том же узле отображаются первыми в списке). Выберите метод Минимальные затраты (Lowest Cost), чтобы направлять клиентов на ближайший конечный компьютер с помощью ссылок сайта (которые можно настроить в консоли Active Directory — Пользователи и компьютеры (Active Directory Sites And Services)). Чтобы клиенты не могли получать доступ к конечному объекту на другом сайте Active Directory, выберите метод Исключить конечные объекты вне сайта клиента (Exclude Targets Outside Of The Client's Site). По умолчанию папки наследуют метод сортировки корня именного пространства, но вы можете редактировать также свойства отдельных папок. Параметр Продолжительность кэширования (Cache Duration) определяет время ожидания клиентов перед запросом новой ссылки.

Дополнительно (Advanced) Выберите один из двух способов оптимизации опроса: Оптимизировать для согласованности (Optimize For, Conelite псу) или Оптимизировать для масштабируемости (Optimize For Scalability). Первый способ конфигурирует серверы именного пространства для запросов основного контроллера домена PDC (Primary Domain Controller) при каждом изменении именного пространства, чтобы ускорить отображение этих изменений для пользователей. Второй способ, при котором регулярно опрашивается ближайший контроллер домена, снижает количество запросов, повышая таким образом производительность и уменьшая нагрузку на основной контроллер домена.

Задание 3. *Добавление папок в именное прост ранст во DFS*

Прежде чем использовать именное пространство, в него нужно добавить папки. Папки могут быть чисто организационными (то есть они существуют только и именном пространстве DFS) или связанными с общей папкой на сервере. При подключении к именному пространству DFS эти папки отображаются аналогично папкам в стандартной файловой системе.

Для того чтобы добавить папки в именное пространство DFS, выполните следующее.

1. В Диспетчере сервера (Server Manager) выберите узел Роли\Файловые службы\Управленис DFS\npocrpaiicTBa имен (Rolc9\Fllc Servlces\DPS Managemcnt\Namcipaccs).

2. В панели сведений щелкните правой кнопкой мыши именное пространство DFS и примените команду Создать папку (New Folder). Откроется диалоговое окно Создание папки (New Folder).

3. Введите имя для папки. Если она будет использоваться лишь в организационных целях (например, только содержать другие папки), щелкните ОК. Если папка должна содержать файлы, щелкните кнопку Добавить (Add), чтобы связать ее с общей папкой. Если добавить множество папок с конечными объектами, для этих папок можно конфигурировать автоматическую репликацию. Щелкните ОК.

Задание 4. *Работа с автономными файлами*

Мобильным пользователям может понадобиться доступ к общим папкам даже в случае отключения их компьютеров от внутренней сети организации. Автономные файлы (Offline Files) обеспечивают эту возможность, разрешая клиентским компьютерам автоматически копировать копии файлов в общих папках, а также предоставляя прозрачный доступ к файлам при отключении пользователя от сети. При следующем подключении к сети компонент Автономные файлы синхронизирует все обновления.

Администраторы сервера могут конфигурировать автономные файлы в общей папке, а пользователи клиентских компьютеров могут настраивать автономные файлы при подключении к общей папке. Чтобы сконфигурировать кэширование автономных файлов в общей папке, выполните такие шаги.

1. В Диспетчере сервера (Server Manager) выберите узел Роли\Файлоаые службы\Управление общими ресурсами и хранилищами (Rolcs\File Serviced Share And Storage Management).

2. В панели сведений щелкните правой кнопкой мыши общий ресурс, который хотите настроить, и примените команду Свойства (Properties).

3. На вкладке Доступ (Sharing) щелкните кнопку Дополнительно (Advanced),

4. В диалоговом окне Дополнительно (Advanced) перейдите на вкладку Кэширование (Caching), представленную на рисунке 1.1.

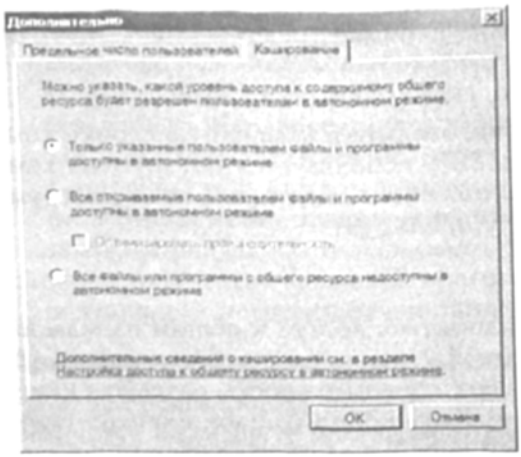

Рисунок 1.1. Наст ройка парамет ров авт ономных файлов для общей папки

Выберите одну из следующих опций и дважды щелкните ОК.

■ Только указанные пользователем файлы и программы доступны в автономном режиме (Only The Files And Programs That Users Specify Arc Available Offline) Пользователи должны вручную выбирать файлы, к которым хотят получать доступ в автономном режиме.

■ Все открываемые пользователем файлы и программы доступны и в автономном режиме (Only The Files And Programs That User» Open From The Share Are Automatically Available Offline). Файлы, к которым пользователи подключаются в сети, автоматически копируются в течение ограниченного интервала времени.

■ Все файлы или программы с общего ресурса недоступны и автономном режиме (No Files Or Programs From The Shure Arc Avuilable Offline). Запрещает пользователям получать доступ к автономным файлам. Эту опцию лучше всего задействовать при работе с конфиденциальными документами, которые не должны храниться на мобильных компьютерах.

Доступ к этим параметрам можно также получить в Проводнике Windows (Windows Explorer), щелкнув кнопку Дополнительный Доступ (Advanced Sharing) на вкладке Доступ (Sharing) диалогового окна свойств общей папки, а затем – кнопку Кэширование (Caching).

Если выбрать опцию Только указанные пользователем файлы и программы доступны в автономном режиме (Only The Files And

Programs That Users Specify Are Available Offline), пользователи должны отконфигурировать подключенные диски для включения автономных файлов. В Windows Vista настройка автономных файлов на подключенном диске выполняется следующим образом.

1. В Проводнике Windows (Windows Explorer) щелкните правой кнопкой мыши сетевую папку или файл и примените команду Свойства (Properties).

2. На вкладке Автономные файлы (Offline Files) установите флажок Всегда доступны в автономном режиме (Always Available Offline) и щелкните OK.

Система Windows автоматически синхронизирует файл или папку. Пользователи могут позже вернуться на вкладку Автономные файлы (Offline Files) и скопировать последнюю версию файла.

Задание 5. *Уст ановка роли DFS на Windows Server 2008*

1. Откройте оснастку Server Manager (рисунок 1.2).

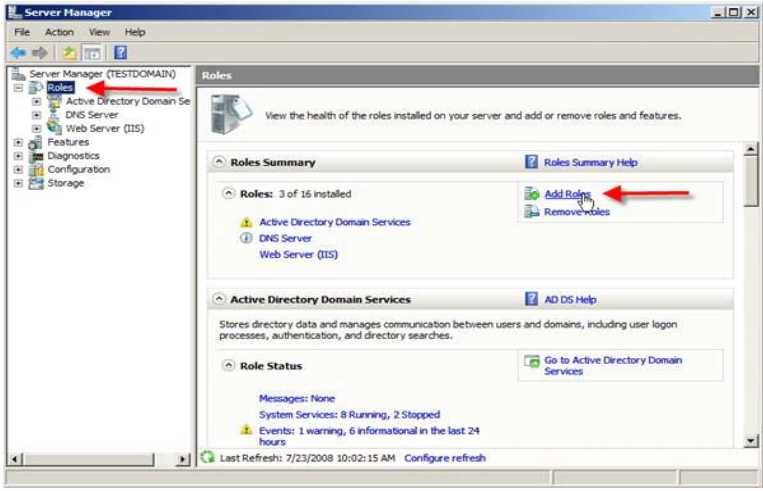

Рисунок 1.2. Окно Server Manager

2. Перейдите в раздел **Roles** и выберите **Add Roles**.

3. Из списка ролей выберите **File Services** (рисунок 1.3).

4. Появится информационное окно (**File Services),** перейдите далее, нажав Next (рисунок 1.4).

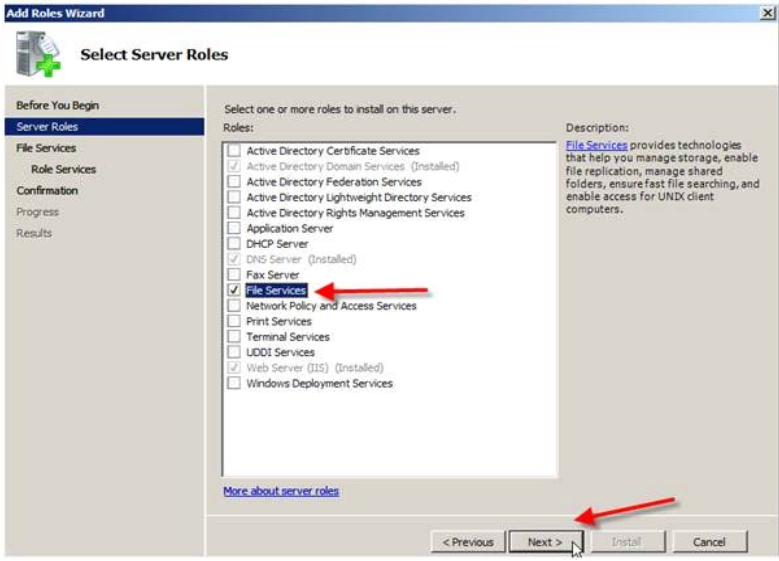

Рисунок 1.3. Выбор роли сервера

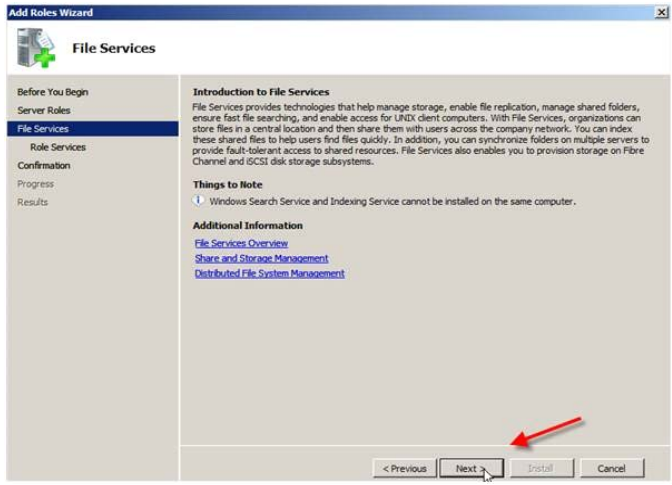

Рисунок 1.4. Информационное окно

5. В списке ролей выберите **Distributed File System,** а также **DFS Namespaces и DFS Replication**; после чего нажмите **Next** (рисунок 1.5).

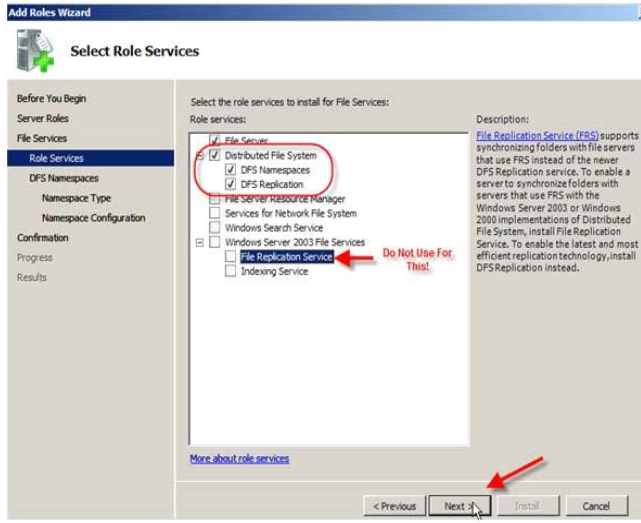

Рисунок 1.5. Выбор ролей

6. На экране «Create a DFS Namespace», вы можете указать, хотите ли вы создать пространство имен немедленно, или позднее.

В данном случае не будем создавать корень пространства имен. Поэтому выберите «**Create a namespace later using the DFS Management snap-in Server Manager»** и нажмите Next (рисунок 1.6)**.**

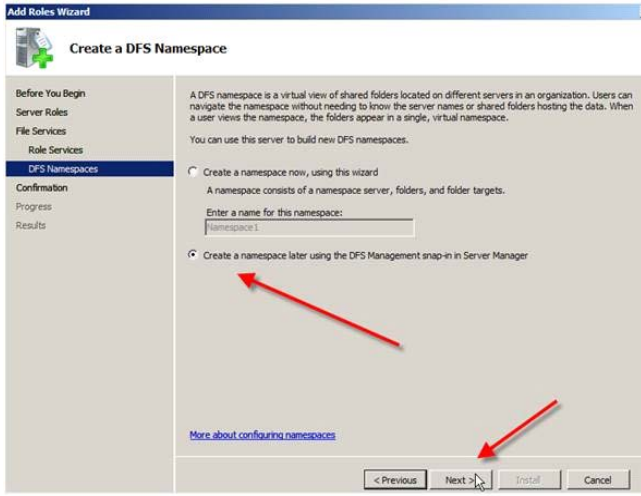

Рисунок 1.6. Окно создания пространства имен

7. На следующем экране, нажав **Install**, запустите процесс установки службы DFS (рисунок 1.7).

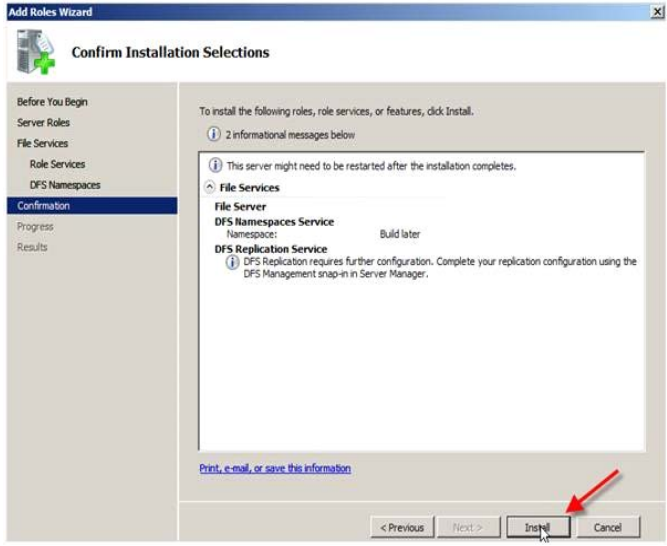

Рисунок 1.7. Окно запуска процесса установки службы DFS

8. После установки DFS, в консоли Server Manager появится новая роль **File Services** со следующим списком установленных компонентов:

Distributed File System
DFS Namespaces
DFS Replication

Задание 6. *Добавление служ бы ролей Распределенная файловая сист ема DFS*

В этом задании требуется добавить роль сервера Файловые службы (Files Services) и службу ролей Распределенная файловая система DFS (Distributed File System) на компьютеры Dcrsvl и Boston. Затем нужно создать именное пространство DFS на обоих компьютерах, а также общие папки, которые станут частью этого именного пространства. Общие папки будут автоматически выполнять репликацию файлов, обеспечивая избыточность для клиентов, которым требуется доступ к файлам.

1. На машине Dcsrvl откройте Диспетчер сервера (Server Manager), щелкните правой кнопкой мыши узел Роли (Roles) и примените

команду Добавить роли (Add Roles). Запустится Мастер добавления ролей (Add Roles Wizard).

2. На странице Перед началом работы (Before You Begin) щелкните кнопку Далее (Next).

3. На странице Роли сервера (Server Roles) выберите Файловые службы (File Services) и щелкните кнопку Далее (Next).

4. На странице Файловые службы (File Services) щелкните кнопку Далее (Next).

5. На странице Выбор служб ролей (Select Role Services) установите флажки для служб ролей Файловый сервер (File Server). Распределенная файловая система DFS (Distributed File System) и Диспетчер ресурсов файлового сервера (File Server Resource Manager). Щелкните кнопку Далее (Next).

6. На странице Создание пространства имен DFS (Create A DFS Namespace) введите имя пространства Общий. Щелкните кнопку Далее (Next).

7. На странице Тип пространства имен (Namespace Type) оставьте параметры по умолчанию. Щелкните кнопку Далее (Next).

8. На странице Конфигурация пространства имен (Namespace Configuration) щелкните кнопку Далее (Next).

9. На странице Наблюдение за хранилищем (Storage Usage Monitoring) установите флажки для всех локальных дисков и щелкните кнопку Далее (Next).

10. На странице Параметры отчета (Report Options) щелкните Далее (Next).

11.На странице Подтверждение (Confirmation) щелкните кнопку Установить (Install).

12.На странице Результаты (Results) щелкните кнопку Закрыть (Close).

Повторите эти шаги на машине Boston, но не создавайте именное пространство на странице Создание пространства имен DFS (Create A DFS Namespace).

Задание 6. *Добавление сервера в простр анст во имен DFS*

Теперь добавьте в пространство имен DFS реплицированную папку, выполнив следующие действия.

1. На машине Dcsrvl откройте Диспетчер сервера (Server Manager), щелкните правой кнопкой мыши узел Роли\Файловые службы\Управление DFS\Пространства имен\ (Roles\File Services\DFS

Management NameSpaces\Общий) и примените команду Добавить сервер пространства имен (Add Namespace Server).

2. Щелкните кнопку Обзор (Browse). В диалоговом окне Выберите компьютер (Select Computer) введите имя Boston и щелкните кнопку ОК. Если будет предложено запустить на компьютере Boston службу Пространство имен DFS (DFS Namespace), щелкните кнопку Да (Yes). Вновь щелкните (Ж, чтобы закрыть диалоговое окно Добавление сервера пространства имей (Add Namespace Server).

3. В панели сведений перейдите на вкладку Серверы пространства имен (Namespace Servers). На ней будут представлены оба сервера. Если один из них отключен, клиенты смогут подключаться ко второму. Так обеспечивается избыточность для именных пространств DFS.

Задание 7. *Добавление реплицированной папки в пространство имен DFS*

Создадим общую папку с именем Files на Dcsrvl и Boston, добавим ее в пространство имен DFS и отконфигурируем для репликации.

1. На машине Dcsrvl откройте Диспетчер сервера (Server Manager), щелкните правой кнопкой мыши узел Роли\Файловые службы\Управление общими ресурсами и хранилищами (Roles\File Services\Share And Storage Management) и выполните команду Подготовить общий ресурс (Provision Share). Запустится Мастер подготовки общих папок (Provision A Shared Folder Wizard).

2. На странице Размещение общей папки (Shared Folder Location) введите путь C:\Files. Щелкните кнопку Далее (Next). Если будет предложено, щелкните Да (Yes), чтобы создать папку.

3. На странице Разрешения NTFS (NTFS Permissions) выберите опцию Да, изменить разрешения NTFS (Yes, Change NTFS Permissions). Щелкните кнопку Изменить разрешения (Edit Permissions) и предоставьте группе Пользователи (Users) разрешение Изменить (Modify). Щелкните ОК. Затем щелкните кнопку Далее (Next).

4. На странице Протоколы общего доступа (Share Protocols) введите имя общего ресурса Files. Щелкните кнопку Далее (Next).

5. На странице Параметры SMB (SMB Settings) щелкните кнопку Дополнительно (Advanced). На вкладке Кэширование (Caching) выберите опцию Все файлы или программы из общего ресурса недоступны в автономном режиме (No Files Or Programs From The Share Are Available Offline). Таким образом, мобильные компьютеры не будут хранить локально кэшируемую копию файлов. Щелкните ОК, а затем щелкните кнопку Далее (Next).

6. На странице Разрешения SMB (SMB Permissions) выберите опцию Администраторы имеют полный доступ; все остальные пользователи и группы имеют доступ только на чтение (Administrators Have Full Control; All Other Users And Groups Have Only Read Access). Щелкните кнопку Далее (Next).

7. На странице Политика квот (Quota Policy) установите флажок Применить квоту (Apply Quota), Выберите опцию Автоматически применять шаблон для создания квот на существующих и новых вложенных пайках (Auto Apply Template To Create Quotas On Existing And New Subfolders). Затем в раскрывающемся списке Наследовать свойства из следующего шаблоны квоты (Derive Properties From This Quota Template) выберите шаблон Предел 200 МБ с расширением 50 МБ (200 MB Limit With 50 MB Extension). Щелкните кнопку Далее (Next).

8. На странице Политика фильтра блокировки файлов (File Screen Policy) установите флажок Применить фильтры блокировки файлов (Apply Pile Screen). В раскрывающемся списке Наследовать свойства из следующего шаблона фильтра блокировки файлов (Derive Properties From This File Screen Template) выберите шаблон Блокировать исполняемые файлы (Block Executable Files). Щелкните кнопку Далее (Next).

9. На странице Публикация в пространстве имей DPS (DPS Namespace Publishing) установите флажок Публиковать общий ресурс SMB в пространстве имен DPS (Publish The SMB Share To A DFS Namespace). В поле Родительская папка в пространстве имен (Parent Folder In Namespace) введите путь к папке \\nwtraders.msfi\Public (или укажите имя своего домена). В поле Имя новой папки (New Folder Name) введите имя Files. Щелкните кнопку Далее (Next).

10.На странице Проверить параметры и создать общий ресурс (Review Settings And Create Share) щелкните кнопку Создать (Create).

11. Щелкните кнопку Закрыть (Close).

12. На машине Boston откройте окно командной строки с полномочиями администратора и запустите следующие команды для создания папки, предоставления разрешения NTFS Изменение (Modify) для группы Пользователи (Users) и назначения общего доступа к папке. Так вы продублируете общую папку, созданную па машине Dcsrvl с помощью Мастера подготовки общих папок (Provision A Shared Folder Wizard),

```
mkdir C:\Files
icacls C:\Files\ /grant users:M
net share Files=C:\Files /GRANT:Users, READ /GRANT:Administrators, FULL /CACHE:None
```

Теперь на машине Dcsrvl добавьте общую папку \\Boston\Fihs в качестве конечной папки для \nwtmders.msft\Public\Files.

1. На машине Dcsrvl откройте Диспетчер сервера (Server Manager), щелкните правой кнопкой мыши папку \\nwlraders.msft\Public и примените команду Обновить (Refresh).

2. В диспетчере сервера щелкните правой кнопкой мыши папку \\nwtmders.msft\Public\Files и примените команду Добавить конечный объект папки (Add Folder Target).

3. В диалоговом окне Создание конечного объекта папки (New Folder Target) введите имя папки \\Boston\Files. Щелкните OK.

4. В диалоговом окне Репликация (Replication) щелкните кнопку Да (Yes), чтобы создать группу репликации между серверами Dcsrvl и Boston. Запустится Мастер репликации папок (Replicate Folder Wizard).

5. На странице Группа репликации и имя папки репликации (Group And Replicated Folder Name) щелкните кнопку Далее (Next).

6. На странице Replication Eligibility щелкните кнопку /Далее (Next).

7. На странице Основной член (Primary Member) выберите компьютер Dcsrvl. Щелкните кнопку Далее (Next).

8. На странице Выбор топологии (Topology Selection) выберите опцию Полное слияние (Full Mesh). Щелкните кнопку Далее (Next). Отметим, что в случае с несколькими партнерами репликации (более двух-трех) о постоянном обновлении одного сервера более эффективно использовать топологию Hub And Spoke.

9. На странице График и полоса пропускания группы репликации (Replication Group Schedule And Bandwidth) щелкните кнопку Далее (Next). Отметим, что вы можете ограничить пропускную способность (чтобы снизить влияние репликации на работу других приложений) или выполнять репликацию только во время простоя.

10.На странице Проверить параметры и создать группу репликации (Review Settings And Create Replication Group) щелкните кнопку Создать (Create).

11. Затем на странице Подтверждение (Confirmation) щелкните кнопку Закрыть (Close).

12.В диалоговом окне Задержка репликации (Replication Delay) щелкните кнопку OK.

13. В диспетчере сервера, в узле Управление DFS\Пространства имен (DFS Management\Namespaces), выберите папку \\nwtraders.msft\Public\Files и в панели сведений перейдите на вкладку Репликация (Replication). Оба компьютера, Dcsrvl и Boston, будут указаны в качестве участников репликации.

14. В диспетчере сервера откройте узел Управление DFS\Репликация (DFS Management\Replication) и выберите папку nwtraders.msft\Public\Files. В панели сведений просмотрите все четыре вкладки с информацией о группе репликации, созданной мастером репликации папок.

Задание 8. *Проверка репликации DFS*

В этом упражнении вы подключитесь к именному пространству DFS и создадите файл для проверки автоматической репликации.

1. Войдите на компьютер Dcsrvl как обычный пользователь, щелкните кнопку Пуск (Start) и откройте Компьютер (Computer).

2. В панели инструментов окна Компьютер (Computer) щелкните опцию Подключить сетевой диск (Map Network Drive).

3. В окне Подключить сетевой диск (Map Network Drive) введите имя папки \\nwtraders.msft\Public\Files. Затем щелкните кнопку Готово (Finish). Проводник Windows (Windows Explorer) подключит диск Z к общей папке.

4. На новом подключенном диске создайте текстовый файл: в панели сведений, на пустом месте, щелкните правой кнопкой мыши, примените команду Создать (New) и выберите опцию Текстовый документ (Text Document). Ваши права ограничены компонентом UAC (User Account Control) до уровня обычного пользователя, а группа Пользователи (Users) имеет лишь разрешение на чтение (Read), поэтому, даже несмотря на предоставление группе Пользователи разрешения NTFS Изменение (Modify), создать файл вы не сможете.

5. В проводнике Windows выберите папку C:\Files. Затем щелкните правой кнопкой мыши на пустом месте в панели сведений, примените команду Создать (New) и выберите опцию Текстовый документ (Text Document). Назначьте документу имя Текстовый файл. Затем откройте файл и введите фразу «Hello, world». Сохраните и закройте файл.

6. На машине Boston откройте проводник Windows и просмотрите папку C:\Files. Обратите внимание на выполнение репликации документа Текстовый файл (на это может потребоваться несколько минут). Откройте файл и убедитесь, что он содержит введенный вами текст.

2. Лабораторная работа №2. Создание учебной распределенной базы данных

Цель работы: *подготовка к выполнению следующих лабораторных работ, закрепление навыков создания и наполнения базы данных.*

2.1 Пояснения к выполнению лабораторной работы

При выполнении этой и следующих работ потребуются команды DDL (создание, удаление и изменение структуры таблиц) и команды DML (вставка, удаление, обновление и извлечение данных).

Общий вид команды для *создания таблицы*:

```
CREATE TABLE имя_таблицы
(список_определений_столбцов [,список ограничений_на_таблицу]
)
```

Определение каждого столбца в списке имеет вид:

имя_столбца тип_столбца [DEFAULT значение_по_умолчанию] [ограничения_на_столбец]

Типы данных

Числовые типы. В Oracle внутреннее представление всех числовых данных одинаково – все числа хранятся в двоично-десятичном представлении. Поэтому система хранения Oracle поддерживает единственный числовой тип NUMBER[(p[,s])], где p,s – количество десятичных знаков числа и его дробной части.

Oracle, однако, принимает и все стандартные числовые типы и автоматически переводит их во внутреннее представление NUMBER. Например, стандартный тип INTEGER она автоматически приводит к типу NUMBER(38).

Символьные типы:

CHAR [(размер)] строка фиксированной длины (по умолчанию размер типа – 1 байт), максимальный размер в Oracle 2000 байт.

VARCHAR(размер) (в Oracle дополнительно поддерживается тип VARCHAR2, идентичный VARCHAR, но гарантированно неизменный вне зависимости от возможного изменения типа VARCHAR в стандарте) – строка переменной длины, в Oracle его размер не более 4000 байт.

Данные типов CHAR и VARCHAR хранятся по-разному. Под CHAR всегда отводится столько байт, сколько указано в размере, а под

данные типа VARCHAR память на диске выделяется в соответствии с реальными размерами текста плюс четыре байта для хранения размера.

Константы символьного типа берутся в апострофы. Апостроф внутри символьной константы кодируется двумя апострофами.

Типы даты и времени. В Oracle реализован тип DATE, которые служит для хранения даты и времени суток, и тип TIMESTAMP, который позволяет хранить момент времени с высокой точностью.

Константы типа даты заключаются в апострофы, как и текстовые константы, их формат должен соответствовать настройкам сервера.

Типы для хранения больших объектов. Стандарт поддерживает типы BLOB (Binary Large Object) и CLOB (Character Large Object).

Oracle дополнительно поддерживает тип LONG для хранения больших текстов. Работа с типами LONG и CLOB выполняется значительно медленнее, чем с типом VARCHAR, поэтому они используются только для действительно больших текстов.

Ограничения

Ограничения (constraints) позволяют задавать дополнительные условия проверки вводимых данных, которые СУБД проверяет автоматически. Любое ограничение может быть поименовано, в противном случае, имя для ограничения СУБД создает автоматически. Для явного задания имени к описанию ограничения следует добавить слева фразу CONSTRAINT имя_ограничения.

Можно задать ограничения для одного столбца или для всей таблицы.

Ограничения столбца:

NOT NULL/NULL – допустимы ли пустые (неопределенные) значения в столбце, по умолчанию используется значение NULL

PRIMARY KEY – ограничение первичного ключа, при этом автоматически накладывается ограничение NOT NULL.

UNIQUE – ограничение уникальности (альтернативный ключ), ограничение NOT NULL также накладывается автоматически.

REFERENCES имя_главной_таблицы — внешний ключ, который задается для столбца подчиненной таблицы. Для значений внешнего ключа автоматически выполняется проверка на существование равного значения первичного ключа главной таблицы. При определении внешнего ключа могут быть дополнительно определены правила обеспечения ссылочной целостности. Например, правилами для удаления являются:

ON DELETE RESTRICT – запретить удаление строки главной таблицы, если в подчиненной есть строки, которые на нее ссылаются;

ON DELETE CASCADE – вместе со строкой главной таблицы автоматически удалить все ссылающиеся на нее строки подчиненной таблицы;
ON DELETE SET NULL – при удалении строки главной таблицы установить во внешнем ключе ссылающихся строк значение NULL (это правило может быть определено только в том случае, если на внешний ключ не наложено ограничение NOT NULL);
ON DELETE SET DEFAULT – при удалении строки главной таблицы установить во внешнем ключе значение по умолчанию (это правило может быть определено только в том случае, если при определении внешнего ключа задано DEFAULT значение).

По умолчанию принят запрет удаления строк при наличии ссылок на них (ON DELETE RESTRICT).

Аналогичные правила можно определить для обновления (ON UPDATE), однако необходимости в них обычно не возникает, т.к. первичные ключи обновлять не принято.

CHECK – проверка логических выражений при изменении данных в столбце, например, CHECK (имя_столбца BETWEEN 1 AND 100).

Например:

```
CREATE TABLE t
(c1 NUMBER(8) DEFAULT 0 NOT NULL CONSTRAINT un UNIQUE,
  c2 VARCHAR(100) NOT NULL
)
```

Создается таблица с именем t, содержащая числовой столбец c1, обязательный для заполнения и принимающий значение 0 по умолчанию, все значения этого столбца уникальны (ограничение уникальности имеет имя un) и текстовый столбец c2, обязательный для заполнения текстом, его размер не более 100 символов.

Ограничения таблицы

Данные ограничения используются в том случае, если они затрагивают сразу несколько столбцов:

PRIMARY KEY (список столбцов) — составной первичный ключ;

UNIQUE (список столбцов) — составной альтернативный ключ;

FOREIGN KEY имя_внешнего_ключа (список столбцов) REFERENCES имя_главной_таблицы [правила поддержки ссылочной целостности];

CHECK (логическое выражение, затрагивающее несколько столбцов).

Например:

```
CREATE TABLE t1
( c1 NUMBER(3) NOT NULL,
  c2 DATE NOT NULL,
  c3 NUMBER(3) NOT NULL,
  CONSTRAINT pk_t1 PRIMARY KEY(c1,c2),
  CONSTRAINT ck_t1 CHECK(c1+c3<=200)
)
```

Команда удаления таблицы имеет вид:

DROP имя_таблицы

Нельзя удалить таблицу, если существуют внешние ключи, ссылающиеся на эту таблицу.

Команда изменения структуры таблицы выглядит так:

ALTER TABLE имя_таблицы указания_по_изменению_структуры

Например:

ALTER TABLE t ADD n NUMBER(4) NOT NULL (t— имя существующей таблицы, n — имя нового столбца)

ALTER TABLE t ADD PRIMARY KEY(n)

или

ALTER TABLE t ADD CONSTRAINT pk_t PRIMARY KEY(n)

В последнем примере ограничение первичного ключа получит имя pk_t

ALTER TABLE t DROP PRIMARY KEY(n)

или

ALTER TABLE t DROP CONSTRAINT pk_t

ALTER TABLE t DROP COLUMN n

ALTER TABLE t MODIFY c2 VARCHAR(200)

В последнем примере изменили размер столбца c2

2.2. Схема базы данных

В качестве обучающего примера в некоторых дальнейших работах используется база данных для предварительной продажи билетов на автовокзале, эта же база данных используется в дистанционном практикуме по языку SQL, поэтому кратко поясним ее схему. Билеты на рейсы продаются не более чем за неделю вперед, без фиксации номера места в базе данных. При отправке рейса количество проданных билетов обнуляется и начинается продажа на этот же рейс, но отправляющийся через неделю. Каждый рейс имеет остановки только в тех пунктах, которые соответствуют маршруту этого рейса. Для вычисления стоимости билета используется показатель «цена за километр», он зависит от класса автобуса, назначенного на тот рейс, на который покупается билет.

Логическая схема базы данных в нотации IDEF1X имеет вид:

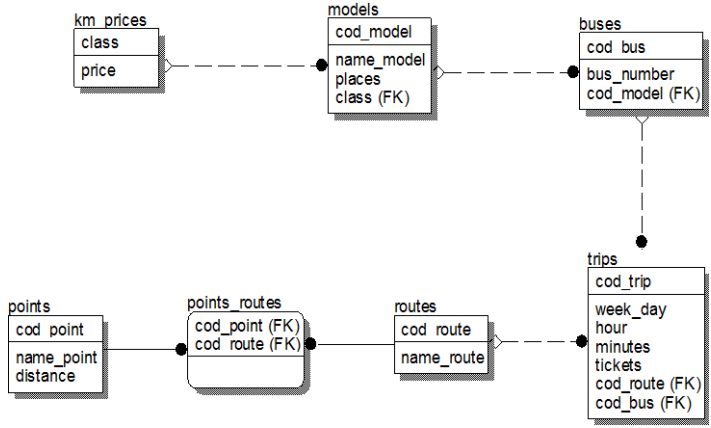

Рисунок 2.1. Схема базы данных Автовокзала

Пояснения к таблицам и их атрибутам:

1. **km_prices** (расценки за километр)
   - class (класс автобуса)
   - price (цена за км для данного класса)
2. **models** (модели автобусов)
   - cod_model (код, суррогатный ключ)
   - name_model (название, атрибут носит справочный характер)
   - places (количество мест в автобусах данной марки)
   - class (класс комфортности)
3. **buses** (автобусы)
   - cod_bus (код автобуса, возможно его инвентарный номер)
   - bus_number (номер ГИБДД)
   - cod_model (марка автобуса, внешний ключ)
4. **points** (населенные пункты)
   - cod_point (код, суррогатный ключ)
   - name_point (название)
   - distance (расстояние от пункта отправления)
5. **routes** (маршруты)
   - cod_route (код, суррогатный ключ)
   - name_route (название маршрута)

6. **trips** (рейсы) связывает сущности Маршруты и Автобусы, дополняя их такими важными атрибутами как время отправления. Здесь же

можно фиксировать и количество проданных билетов, обнуляя этот атрибут при отправлении каждого рейса.

cod_trip (код рейса, суррогатный ключ)
week_day (день недели)
hour (часы)
minute (минуты)
cod_route (код маршрута)
cod_bus (код автобуса, назначенного на данный рейс)
tickets (количество проданных билетов)

2.3 Порядок выполнения лабораторной работы

При выполнении этой и следующих лабораторных работ будет использоваться учебная распределенная система, включающая две базы данных под управлением СУБД Oracle на двух различных узлах локальной вычислительной сети кафедры АВТ.

1. Создадим учетные записи на двух серверах Oracle, используя консольную утилиту SQL Plus. Запустите программу SQL Plus, на ее запрос введите имя пользователя SYSTEM, пароль MANAGER и строку подключения (host string) ORCL (это имя первого сервера, на нем установлена СУБД Oracle 9i).

2. Создайте для себя учетную запись пользователя:

```
CREATE USER имя_пользователя IDENTIFIED BY пароль
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
```

3. Присвойте себе привилегию dba

```
GRANT DBA TO имя_пользователя
```

4. То же самое проделаем для второго сервера – его имя AVTEDUDB, установлена СУБД Oracle 11g и надстройка Oracle Application Express (Apex), благодаря которой возможно удаленное подключение к данному серверу через Интернет (для реализации этой возможности требуется создание учетной записи в Apex, это сделает преподаватель).

Внимание! Имя и пароль пользователя в команде CREATE USER должен совпадать с именем и паролем на первом сервере, это позволит полноценно использовать все возможности Oracle при создании распределенной базы данных. Выйдите из SQL Plus, а затем войдите под своим именем с разными строками подключения, чтобы убедиться в том, что обе учетные записи созданы правильно.

5. Можно переключаться между схемами и базами данных, не выходя из SQL Plus, с помощью команды:
CONNECT имя_пользователя/пароль@строка_подключения

Далее создадим базу данных Автовокзал на сервере AVTEDUDB и заполним таблицы случайными данными. Для этой цели заранее были подготовлены два сценария, они находятся в файлах create_buses.sql и insert_buses.sql. Изучите и выполните эти сценарии.

Таким образом, к концу данной работы сформирована локальная база данных на одном сервере. Далее эта схема будет расширена удаленными таблицами на другом сервере.

3. Лабораторная работа №3. Закрепление навыков разработки запросов и хранимого кода

Цель работы: *закрепление (доведение до автоматизма) навыков разработки SQL-запросов и хранимого кода на языке PL/SQL.*

3.1 Пояснения к выполнению работы

При выполнении данной работы потребуется уверенное знание конструкций языков SQL (DML) и PL/SQL

Команды манипулирования данными:

Вставка одной строки:
INSERT INTO имя_таблицы [(список_имен_столбцов)]
VALUES (список значений)

Например:
INSERT INTO students (cod_st, name_st, born) VALUES (3, ‘Петров’,’12.03.1990’)

Вставка множества строк на основе запроса по другим таблицам:
INSERT INTO имя таблицы [(список столбцов)]
SELECT … (запрос на выборку из других таблиц)

Например:
INSERT INTO marks (cod_st, cod_sub)
SELECT students.cod_st, subject.cod_sub
FROM students, subject WHERE cod_sub=5

В таблицу оценок будут добавлены все студенты, в качестве предмета будет помещен предмет с кодом 5. Теперь преподавателю этого предмета достаточно проставить оценки в уже созданных строках.

Команда удаления строк:

DELETE FROM имя_таблицы
[WHERE условие_отбора_строк_для_удаления]

`Например: удалить всех студентов с фамилией Иванов

DELETE FROM students WHERE name_st='Иванов'

Полностью очистить таблицу students: DELETE FROM students

Нельзя удалить строку родительской таблицы, если есть связанные строки в подчиненной таблице и действует правило обеспечения ссылочной целостности ON DELETE RESTRICT.

Надо заметить, что для полной очистки таблицы существует более эффективная команда

TRUNCATE TABLE имя_таблицы

`Команда обновления данных:

UPDATE имя_таблицы SET столбец1=значение [, столбец2=значение …] [WHERE условие_отбора_строк]

Например: прибавим по одному баллу к каждой оценке:

UPDATE marks SET mark= mark+1

Запрос на выборку данных в общем виде выглядит так:

SELECT [DISTINCT] список_выражений(* - все столбцы таблицы)
FROM список_таблиц, представлений, запросов
(возможно, с операцией соединения JOIN)
[WHERE условия_отбора_строк]
[GROUP BY список_выражений_для_группировки]
[HAVING условия_отбора_групп_с_агрегатными_функциями]
[ORDER BY список_выражений_для_сортировки]

Основные конструкции языка PL/SQL, используемого в Oracle

Блок PL/SQL. Типы данных. Блок – совокупность операторов, заключенная в операторные скобки `BEGIN ... END.` Объявление переменных предшествует блоку и образует секцию объявления, которая начинается словом `DECLARE`. Поддерживаются те же типы, что и для столбцов таблиц. Можно указать тип переменной, явно ссылаясь на определенный столбец или таблицу. Например:

```
DECLARE x INTEGER;
  fio students.name_st%TYPE;
  s subjects%TYPE;
BEGIN
  . . .
END
```

Операторы языка PL/SQL

1. *Оператор присваивания:* переменная:= выражение;
2. *Условный оператор:*

```
IF условие THEN оператор1; оператор2;  ….
[ELSIF условие THEN оператор3; оператор4; ...]
[ELSE оператор5; оператор6; ...]
END IF;
```

3. *Операторы цикла*

А. Бесконечный цикл, условие выхода задается в теле цикла:

```
LOOP
  оператор1; оператор2; ...
  EXIT WHEN условие выхода из цикла;
END LOOP;
```

Б. Цикл с предусловием

```
WHILE условие LOOP
  оператор1; оператор2;  ...
END LOOP;
```

С. Цикл с параметром:

```
FOR параметр IN (множество значений) LOOP
  оператор1; оператор2;  ...
END LOOP;
```

Множество значений параметра обычно задается в виде диапазона (начальное_значение .. конечное_значение)

4.*Оператор возврата из процедуры/функции:*

```
RETURN;
RETURN выражение (только функции).
```

Средства для обработки исключительных ситуаций. Общий подход к обработке исключительных ситуаций состоит в том, что для каждой ситуации определяется ее обработчик. Например:

```
BEGIN
  INSERT INTO … VALUES … ;
  SELECT ... INTO …;
EXCEPTION
  WHEN DUPLICATE_KEYS THEN...
  WHEN TOO_MANY_ROWS THEN ...
  WHEN OTHERS THEN      ...
END;
```

Использование команд SQL. Курсоры. Команды INSERT, DELETE и UPDATE используются в программе на PL/SQL в качестве отдельных

операторов наряду с другими операторами языка. В данных командах разрешено использовать переменные везде, где по правилам SQL используются константы.

Результатом оператора SELECT является множество строк, для обработки которых в стандарт SQL введен механизм курсора.

Различают неявный и явный курсоры. Неявный курсор можно использовать только в том случае, если запрос на выборку возвращает ровно одну строку. Тогда этот результат можно поместить в обычные переменные, используя расширенный синтаксис команды SELECT:

SELECT список_выражений INTO список_переменных ... остальная часть оператора SELECT

Количество переменных в списке и их типы должны в точности соответствовать списку выражений оператора SELECT.

При выполнении команды SELECT ... INTO … в различных случаях ее применения могут возникнуть две разные исключительные ситуации:

- TOO_MANY_ROWS, если запрос SELECT возвращает несколько строк
- NO_DATA_FOUND, если запрос SELECT вообще не возвращает данных.

Явный курсор является более универсальным средством обработки произвольной выборки из базы данных. Он должен быть явно объявлен в разделе DECLARE.

DECLARE CURSOR имя_курсора IS SELECT ... (запрос на выборку)

Например:

```
DECLARE CURSOR cur IS
  SELECT name_st FROM students WHERE name_st LIKE ‘A%’
```

Объявление курсора не является выполнимым оператором. Выборка, заданная в объявлении курсора, выполняется только при его открытии.

Например:

```
OPEN CURSOR cur
```

После открытия курсора можно последовательно выбирать строки курсора, используя оператор FETCH. Например:

```
FETCH cur INTO fio
```

Переменная fio должна быть предварительно объявлена.

Каждое следующее выполнение FETCH выбирает значение столбцов из следующей строки выборки в переменные заданного списка. Оператор FETCH, как правило, применяется в цикле.

Например:

```
LOOP
  FETCH cur INTO fio;
  ...
  EXIT WHEN NOT cur%FOUND;
END LOOP;
```

После того как выбраны все нужные строки, курсор должен быть закрыт. Например: CLOSE cur

Цикл по курсору

Некоторые СУБД, в том числе Oracle, поддерживают цикл по курсору:

```
  FOR параметр IN имя_курсора LOOP
    ...
  END LOOP;
```

Использование такого цикла не требует операций открытия и закрытия курсора – они выполняются неявно. Параметр цикла не требуется объявлять в секции DECLARE, его тип определяется автоматически как RECORD, а имена полей записи соответствуют именам в объявлении курсора. Например:

```
FOR cur_rec IN cur LOOP
   ... cur_rec.name_st…
  END LOOP;
```

Хранимые процедуры и функции

Хранимая процедура создается оператором SQL:

```
CREATE [OR REPLACE] PROCEDURE имя[(список_параметров)]
AS
блок PL/SQL
```

Аналогично создается хранимая функция:

```
CREATE [OR REPLACE] FUNCTION имя[(список_параметров)]
RETURN тип_результата, возвращаемого функцией
AS
блок PL/SQL, обязательно содержащий оператор
RETURN выражение
```

В списке параметров должен быть описан режим использования каждого параметра: IN (только входной – используется по умолчанию), OUT (только выходной), INOUT (и входной, и выходной). Режим использования указывается после имени параметра. Типы параметров, как и типы переменных, можно указывать явно или с помощью ссылки на соответствующий столбец или таблицу.

Триггеры

Триггеры – особый вид хранимых процедур, которые запускаются автоматически при наступлении определенных событий в базе данных. В рамках работы рассмотрим только триггеры уровня таблицы.

Триггер создается при помощи команды SQL:

```
CREATE [OR REPLACE] TRIGGER имя_триггера
время_активизации активизирующая_команда ON имя_таблицы
[FOR EACH ROW]
[WHEN дополнительное условие запуска триггера]
AS
Блок PL/SQL
```

В теле триггера можно использовать любые операторы PL/SQL, кроме операторов SQL, которые изменяют ту таблицу, для которой был создан данный триггер. Любые другие таблицы изменять можно.

В теле триггера в Oracle можно использовать две предопределенные переменные, которые обозначают модифицируемую строку таблицы:

:NEW – новое значение строки, применяется для команд INSERT и UPDATE

:OLD – старое значение строки (до модификации), применяется для команд DELETE и UPDATE.

3.2 Порядок выполнения лабораторной работы

Следующие задания выполняются в созданной базе данных и/или проверяющей системе кафедры АВТ. Их можно выполнять в любом порядке, в том числе, удаленно через Интернет.

Задание 1.. Разработайте и проверьте 7 запросов на выборку. Для автоматической проверки запросов будет использована проверяющая система кафедры АВТ (http://atpp.vstu.edu.ru), в которой имеются тексты всех заданий – из имеющихся 70 заданий выберите по одному из каждой десятки.

Задание 2. Придумайте 2 новых запроса к базе данных Автовокзал с решениями для пополнения проверяющей системы.

Задание 3. Следующая задача – создать серверные объекты (представления, хранимые процедуры, триггеры) для работы приложений КАССИРОВ и ДИСПЕТЧЕРА Автовокзала.

Функции кассира:

1. Продажа билетов, при этом должно увеличиться количество проданных билетов в таблице trips, также должна быть рассчитана стоимость купленных билетов.

2. Возврат билетов – количество проданных билетов должно уменьшиться на величину, равную количеству возвращаемых билетов.

Функция диспетчера: обнуление количества проданных билетов в таблице trips при отправке рейса, при этом информация об отправленном рейсе сохраняется в хранилище данных.

В качестве обучающего примера создадим серверные объекты для реализации продажи билетов – представление кассира и хранимую процедуру. Представление **points_trips**, содержащее всю необходимую для кассира справочную информацию о пунктах и рейсах, включая и количество свободных мест на каждый рейс (в таблице trips хранится количество проданных билетов), создается при помощи команды:

```
create view points_trips
as
select points_routes.cod_point  cod_point,
        trips.cod_trip  cod_trip,
        routes.name_route  name_route,
        trips.week_day  week_day,
       trips.hour||':'|| trips.minute  time,
        models.class class,
        models.places- trips.tickets  free_places
from points_routes, routes, trips, buses, models
where points_routes.cod_route=routes.cod_route  and
         routes.cod_route= trips.cod_route             and
         trips.cod_bus=buses.cod_bus                   and
         buses.cod_model=models.cod_model
```

Обратите внимание на то, как в Oracle обозначается операция конкатенации — ||. Преобразование из числового типа в текстовый для столбцов **hour** и **minute** при этом будет выполнено автоматически.

Если представление создано без ошибок, проверьте выдаваемую им виртуальную таблицу при помощи команды

```
select * from points_trips
```

Представление на основе нескольких таблиц в Oracle всегда является необновляемым. Для изменения одного единственного поля,

доступного кассирам, напишем хранимую процедуру с параметрами, которая будет запускаться из приложения для кассиров.

Назовем процедуру **Sale**, ее входными параметрами будут **Код рейса** (cod_r), **Код пункта** (cod_p) и необходимое **количество билетов** (n_ticket), выходными — **признак**, успешно ли выполнена продажа (err), а также **стоимость** всех купленных билетов (s). Примерный текст этой процедуры может иметь вид:

```
create or replace procedure sale (cod_r trips.cod_trip%type,
                                  cod_p points.cod_point%type,
                                        n_ticket number,
                                        err out number, s out number
                                      )
as
free_pl number;
begin
        select free_places into free_pl from points_trips
        where cod_point=cod_p and cod_trip=cod_r;
        if n_ticket>free_pl then
                err:=1;
        else
                err:=0;
                update trips set tickets=tickets+n_ticket
                where cod_trip=cod_r;
                select km_prices.price*points.distance*n_ticket into s
                from km_prices, points_trips, points
                where km_prices.class=points_trips.class and
              points_trips.cod_point=points.cod_point and
                points_trips.cod_point=cod_p and
points_trips.cod_trip=cod_r;
        end if;
exception
        when TOO_MANY_ROWS then
                err:=2;
        when NO_DATA_FOUND then
                err:=3;
        when OTHERS then
                err:=4;
end;
```

Теоретически при правильном функционировании системы исключительные ситуации произойти не должны. Но все-таки раздел exception принято включать в текст хранимой процедуры.

Исключение TOO_MANY_ROWS могло бы возникнуть при исполнении запроса на выборку, если бы он возвратил более одной строки, исключение NO_DATA_FOUND — если бы select вообще не возвратил строк, OTHERS — любые другие сбойные ситуации.

Если процедура создана без ошибок компиляции, то ее можно проверить непосредственно в окне команд, запустив на выполнение. Для этого рекомендуем написать небольшой отладочный блок PL/SQL.

Введите следующий текст и запустите его на выполнение:

```
declare err number;
          s number;
begin
  sale(1,1,3,err,s);
  dbms_output.put_line('err=' || err);
  dbms_output.put_line('s=' || s);
end;
```

Чтобы эта процедура работала в SQL Plus, необходимо установить для параметра среды serveroutput значение Вкл (в меню Параметры).

Входные параметры обозначают, что мы продаем 3 билета на рейс с кодом 1 до пункта тоже с кодом 1. Если рейс 1 проходит через пункт 1 и на него имеется не менее трех свободных мест, то параметр err должен получиться равным 0, а параметр s должен иметь значение стоимости 3 билетов.

Протестируйте процедуру на разных значениях входных параметров.

Самостоятельные задания

Задание 3.1. По аналогии с процедурой Продажа билетов напишите процедуру Возврат билетов, в которой нужно обязательно проверить, что количество возвращаемых билетов не больше, чем количество проданных билетов на этот рейс (иначе в таблице trips из-за ошибки кассира может оказаться отрицательное количество проданных билетов). Стоимость возвращаемых билетов должна вычисляться, как и в процедуре Продажа билетов.

Задание 3.2. Основная обязанность диспетчера - отправка рейсов в соответствии с расписанием. Создайте представление для диспетчера, отображающего только те рейсы, которые отправляются *сегодня, в порядке возрастания времени* с указанием названия маршрута. Для

получения дня недели, соответствующего текущей дате, используйте выражение **to_number(to_char(sysdate,'d'))**. Функция **sysdate** возвращает текущую дату, параметр ‘d’ функции **to_char** преобразует дату в номер дня недели.

Задание 3.3. Напишите хранимую процедуру «Отправка рейса» (параметр «код рейса»), которая проверяет, продан ли на рейс хотя бы один билет и в этом случае обнуляет количество проданных билетов в таблице **trips**.

Задание 3.4. Создайте таблицу **sent_trips (отправленные рейсы),** которая будет использоваться руководством для анализа деятельности автовокзала**.** Она содержит поля:
id_trip **–** уникальный идентификатор отправленного рейса (PK)
cod_trip – идентификатор рейса (уникален только в пределах недели, но полезен для выполнения группировки)
cod_bus – код автобуса, назначенного на этот рейс
name_route – название маршрута
hour – часы отправления
minute – минуты отправления
week_day – день недели
sent_day – дата отправки рейса (при занесении рейса в хранилище это текущая дата и фактическое время отправки рейса)
fill_bus – наполняемость автобуса в процентах (этот атрибут полезно вычислять при занесении рейса в хранилище)
tickets – количество проданных билетов (может понадобиться при формировании отчетов).

Создайте последовательность:

```
create sequence sent_trips_seq;
```

Для обращения к значениям последовательности используются псевдостолбцы **currval** и **nextval**. **Currval** возвращает текущее значение, **NextVal** инкрементирует текущее значение и возвращает результат, при этом он становится текущим значением. Создайте триггер для автоматического заполнения столбца id_trip:

```
create trigger sent_trip_key
before insert on sent_trips
for each row
begin
select sent_trips_seq.nextval into :new.id_trip from dual;
end;
```

Создайте триггер на обновление таблицы **trips,** который перед обнулением количества проданных билетов (при работе процедуры «отправка рейса») заполнит строку таблицы **sent_trips** данными об отправленном рейсе. Если в триггере вам понадобится переменная, нельзя опускать DECLARE перед ее определением.

Дополнительные задания

Задание 1д. Напишите хранимую процедуру, которая удаляет из таблицы trips те рейсы, для которых средняя наполняемость автобуса меньше заданного значения (параметр процедуры). Посмотрите реальную наполняемость по вашей базе и подберите значение параметра, чтобы при запуске процедуры удалилось несколько рейсов.

Задание 2д. Напишите хранимую функцию (входной параметр «код рейса»), которая возвращает строку, содержащую название маршрута, по которому идет этот рейс, и все остановки этого маршрута. Например: Маршрут: название Остановки: список остановок через пробел или запятую. Эту функцию можно использовать в запросе типа

Select cod_trip, route_points(cod_trip) from trips

Для решения задачи потребуется явный курсор, содержащий названия остановок заданного рейса.

Лабораторная работа №4. Создание связей между базами данных. Фрагментация и репликация

Цель работы: *освоение механизмов СУБД Oracle для создания связей между серверами, разработка распределенных запросов, знакомство с возможностями репликации на основе объекта snapshot*

4.1 Пояснения к выполнению работы

Создание связей (линков) между удаленными базами данных. Связь между серверами баз данных Oracle создается командой:

CREATE PUBLIC DATABASE LINK имя
CONNECT TO “имя пользователя” IDENTIFIED BY “пароль”
USING 'строка подключения';
где строка подключения определяет сетевой адрес удаленного сервера.

Созданная связь является односторонней, т.е. тот сервер, на котором выполнена эта команда, сможет манипулировать данными сервера, указанного в строке подключения, как своими собственными (с небольшим снижением производительности). При выполнении работы мы создадим две связи (на обоих серверах), чтобы обеспечить возможности двустороннего обмена данными.

Для обеспечения «прозрачности» при обращении к удаленным объектам (например, к таблицам на прилинкованном сервере) будем использовать механизм синонимов. Создать синоним можно с помощью команды:
CREATE [PUBLIC] SYNONYM имя_синонима FOR имя_удаленного объекта.

После этого можно использовать два способа обмена данными с удаленным сервером:

1. Фрагментация
2. Репликация

Способ фрагментации данных предполагает использование распределенных запросов и транзакций, извлекающих (изменяющих) удаленные данные. Способ репликации использует синхронизируемые копии данных на различных серверах, что позволяет существенно повысить производительность распределенной информационной системы. При выполнении работы будут освоены оба способа распределения данных.

4.2 Порядок выполнения работы

Для удобства можно открыть два консольных окошка с SQL*Plus:
первое для работы с БД cOracle 9i - ORCL,
второе - с Oracle 11g AVTEDUDB. Можно и в одном окне переключаться между базами данных, используя команду:
CONNECT логин/пароль@имя базы данных (orcl или avtedudb)

После первой лабораторной работы имеются объекты базы данных на сервере ORCL. В схемах AVTEDUDB пока ничего нет.

Задание 1. *Создание связей между серверами Oracle*

В окне **AVTEDUDB** создаем связь для манипулирования данными на **ORCL:**

```
CREATE PUBLIC DATABASE LINK имя CONNECT TO
"ваш_логин_на_orcl" iDENTIFIED BY "ваш_пароль"  USING
'(DESCRIPTION=(ADDRESS=
(PROTOCOL=tcp)(HOST=sur.avt.vstu.edu.ru)(PORT=1521))(CONNECT_
DATA=(SERVICE_NAME=ORCL)))'
```

Внимание! Логин и пароль обязательно заключить в кавычки, строка подключения заключается в апострофы.

Проверьте созданную связь, выполнив, например, запрос:
SELECT * FROM models@имя_только_что_созданной_связи
или запрос к любой другой таблице на **ORCL**.

Убедитесь, что и команды insert, delete, update выполняются для удаленных таблиц, например:
INSERT INTO points@имя_связи VALUES (101,'sokol',45); COMMIT;

Внимание! При этом команды DDL для удаленной БД не поддерживаются – можете в этом убедиться.

Аналогично, в окне **ORCL** создаем связь с **AVTEDUDB:**
CREATE PUBLIC DATABASE LINK имя CONNECT TO "ваш_логин_на_avtedudb" IDENTIFIED BY "ваш_пароль" USING '(DESCRIPTION=(ADDRESS=
(PROTOCOL=tcp)(HOST=avtedu.avt.vstu.edu.ru)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=AVTEDUDB)))'

Задание 2. *Обеспечение прозрачности доступа к удаленным данным.*

Создайте синонимы для нескольких таблиц вашей схемы.

Например, в окне **AVTEDUDB** можно создать синоним для таблицы models, созданной в **ORCL:**
CREATE [PUBLIC] SYNONYM models FOR models@имя_связи
SELECT * FROM models – теперь этот запрос можно выполнить с одинаковым результатом с обоих серверов.

Задание 3. *Использование механизма фрагментации*

Перенесем архив отправленных рейсов на другой сервер (**AVTEDUDB**). Это можно сделать, например, при помощи команд:
CREATE TABLE SENT_TRIPS
AS
SELECT * FROM SENT_TRIPS@ИМЯ_СВЯЗИ

Эта команда успешно выполнится, т.к. обращение к удаленной БД идет в запросе select.

Создать последовательность и триггер удаленно нельзя, также как удалить таблицу sent_trips на orcl – это нужно сделать только в «родных» окнах.

Создайте необходимые синонимы, чтобы процедура отправки рейса (точнее, триггер, который к ней привязан) сохраняла данные об отправленном рейсе на удаленном сервере – добейтесь этого.

Задание 4. *Использование механизма репликации*

В oracle этот механизм реализуется при помощи объекта snapshot (допустимо употреблять слово «снапшот» вместо дословного перевода «мгновенный снимок»). Имеется и более общее понятие – материализованное представление, оно будет появляться в некоторых сообщениях сервера. По техническим причинам удобнее создать снапшот из **AVTEDUDB** в **ORCL.** С этой целью создайте на сервере **AVTEDUDB** еще какую-нибудь дополнительную таблицу, для которой стоит создать реплику на **ORCL.** Допустим, список автобусов, которые находятся в ремонте **buses_repair** (можно всего из одного столбца cod_bus). Создайте для нее синоним в **ORCL.**

Будем создавать snapshot для таблицы **buses_repair** из **в ORCL (**можете создать snapshot и для любой другой таблицы или на основе запроса к таблице**).**

Внимание! Для автоматического обновления снапшота необходимо, чтобы значение init-параметра job_queue_processes на сервере ORCL имело значение 1 (для этого перед выполнением работы был отредактирован файл INIT.ORA, после чего сервер перезапущен).

Теперь самое главное – создаем снапшот.

1 способ – снапшот с полным обновлением

В окне **ORCL** выполняем команду:

```
CREATE SNAPSHOT имя_снапшота BUILD IMMEDIATE
REFRESH COMPLETE
START WITH SYSDATE NEXT SYSDATE + 1/1440
AS SELECT * FROM имя_синонима_удаленной_таблицы
```

Если выдано сообщение, что материализованное представление создано, то все хорошо – реплика уже появилась (в команде есть опция BUILD IMMEDIATE). Можно писать запрос:

```
SELECT * FROM имя_снапшота
```

Теперь проверим, что реплика обновляется каждую минуту. Перейдем в окно **AVTEDUDB** и добавим строку в таблицу **buses_repair** или удалим строку (не забудем про команду COMMIT).

Вернемся в окно **ORCL**. Подождем минуту, и выдадим:

```
SELECT * FROM имя_снапшота;
```

Если вы увидите изменения в реплике, значит, не зря трудились – мы получили обновляемый снапшот. Жаль его удалять, но придется - много снапшотов, обновляемых каждую минуту, создадут лишнюю нагрузку на сеть. УДАЛИТЕ СОЗДАННЫЙ СНАПШОТ.

2 способ – снапшот с быстрым обновлением

Если реплика создается для небольшой таблицы, то возможен вариант полного обновления снапшота. Но для больших объемов данных в команде create snapshot лучше использовать опцию refresh fast (быстрое обновление снапшота – в реплике обновляются только измененные данные).

Для этого сначала убедимся, что в таблице, для которой создается реплика, определен первичный ключ (если нет, то определим). Далее создаем журнал изменений таблицы для быстрого обновления снапшота (в окне **AVTEDUDB).**

CREATE SNAPSHOT LOG ON имя_таблицы WITH PRIMARY KEY

Команда для создания нового, быстро обновляемого снапшота (выполняется в окне **ORCL**)

CREATE SNAPSHOT имя_снапшота BUILD IMMEDIATE
REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 1/1440
WITH PRIMARY KEY
AS SELECT * FROM синоним_удаленной_таблицы

Проверьте факт автоматического обновления созданного вами снапшота.

Библиографический список

1. Нортроп, Т. Проектирование сетевой инфраструктуры Windows Server® 2008: учебный курс Microsoft /Т. Нортроп, Дж.К. Макин — М.: «Русская редакция», 2011. - 592 с.
2. Дейт, К. Введение в системы баз данных: пер. с англ. /К.Дж. Дейт. 8-е издание. – М.: Вильямс, 2006. – 1326 с.
3. Ульман, Д. Введение в системы баз данных: пер. с англ. /Д.Ульман, Д.Уидом. – М.: Лори, 2000. – 512 с.
4. Грибер, М. Введение в SQL / М. Грибер. М:, Лори, 1996. – 379 с.
5. Базы данных: учебник для ВУЗов / Под ред.А.Д.Хомоненко — СПб: Корона принт, 2000. – 416 с.
6. Полякова, Л. Основы SQL. Курс лекций: учебное пособие / Л.Н. Полякова – М.: ИНТУИТ.РУ, 2004. – 368 с.
7. Ржеуцкая, С. Базы данных. Язык SQL: учеб. пособие / С.Ю. Ржеуцкая – Вологда: ВоГТУ, 2010. – 160 с.