МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Кузбасский государственный технический университет имени Т. Ф. Горбачева»

Кафедра информационных и автоматизированных производственных систем

Составитель
О. Н. Ванеев

УПРАВЛЕНИЕ И АВТОМАТИЗАЦИЯ БАЗ ДАННЫХ

Методические материалы

Рекомендовано цикловой методической комиссией специальности
СПО 09.02.07 Информационные системы и программирование
в качестве электронного издания
для использования в образовательном процессе

Кемерово 2018

Рецензенты

Сыркин И. С. – кандидат технических наук, доцент кафедры информационных и автоматизированных производственных систем ФГБОУ ВО «Кузбасский государственный технический университет имени Т. Ф. Горбачева»

Чичерин И. В. – кандидат технических наук, доцент, зав. кафедрой информационных и автоматизированных производственных систем ФГБОУ ВО «Кузбасский государственный технический университет имени Т. Ф. Горбачева»

Ванеев Олег Николаевич

Управление и автоматизация баз данных: методические материалы [Электронный ресурс] для студентов специальности СПО 09.02.07 Информационные системы и программирование очной формы обучения / сост. О. Н. Ванеев; КузГТУ. – Электрон. издан. – Кемерово, 2018.

Методические указания для дисциплины «Управление и автоматизация баз данных» описывают содержание практических, лабораторных занятий и самостоятельной работы, перечень вопросов на защиту выполненных работ.

РАБОТА №1 ПРАКТИЧЕСКАЯ. ПОСТРОЕНИЕ БАЗЫ ДАННЫХ В СРЕДЕ ОДНОЙ ИЗ СУБД

1. ЦЕЛЬ РАБОТЫ

Получить навыки разработки баз данных в среде MS SQL SERVER Management Studio 2008 (2012).

В связи с этим, задачами работы является изучение архитектуры СУБД MS SQL SERVER:

- знакомство с принципом работы в среде MS SQL SERVER Management Studio 2008 (2012);
- изучение принципов создание модели базы данных на основе анализа и выявления объектов предметной области;
- создание базы данных в соответствии с индивидуальным заданием.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ.

2.1. Общие сведения о базах данных

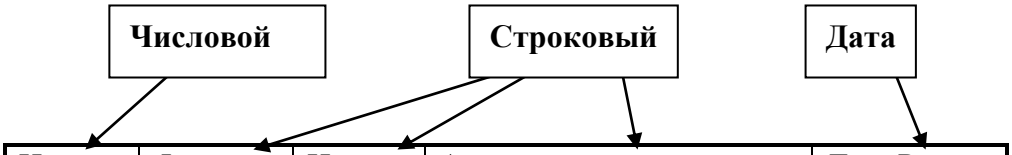
Базовым элементом баз данных, построенных на основе реляционной модели, является отношение. Отношение реализуется в среде различный СУБД как таблица.

Таким образом, таблица это объект, предназначенный для хранения информации в реляционной БД. Информация об единичном экземпляре данных представляется как **запись (кортеж)** или строка в таблице. **Поля (атрибуты)** объекта представляются как – столбцы в табличном виде.

Поля в реляционных базах данных характеризуются следующими свойствами:

1. *Имя поля* – идентификатор поля, по которому организуется программный доступ к нему.
2. *Тип поля* – тип данных, находящихся в этом поле.

Примеры типов представлен на рис. 1.1.



Номер	Фамилия	Имя	Адрес	Дата Рожд.
1025	Иванов	Иван	Пр. Советский 10 – 23	03.02.1978
432	Петров	Петр	Ул. 40 лет октября 20 – 71	18.09.1954
972	Сидоров	Сидор	Ул. Кирова 45 – 67	23.11.1985

Рис.1.1 Таблица – основной элемент базы данных

3. *Размер поля* – величина в байтах, выделяемая для хранения данных в поле. Например: если тип поля СТРОКОВЫЙ, а размер будет равен 10-ти, то это значит, что в ячейку такого поля нельзя будет записать строку более 10 символов. Если задать ЦЕЛЫЙ ЧИСЛОВОЙ тип и установить размер в 4 байта, то числа в ячейке будут принимать значения от 0 до 65535

4. *Инкрементность (счетчик)* – автозаполнение поля в добавленной записи неким значением (как правило числового целого типа).

5. *Ключ* – уникальный идентификатор, характеризующий запись.

6. *Необходимость заполнения* – если поле не обязательно для заполнения, то при добавлении записи (в случае отсутствия данных в поле) оно автоматически заполняется значением по умолчанию, если таковое имеется. Если значения по умолчанию нет, записывается псевдопустое значение «NULL», которое определено в системе специальным идентификатором.

2.2. Системы управления базами данных (СУБД). СУБД MS SQL SERVER 20XX

Система управления базами данных (СУБД). СУБД – вспомогательная система, обеспечивающая работу базы данных.

СУБД обеспечивает:

- логически согласованную работу файлов хранящих данные;
- язык манипулирования данными;
- восстановление информации после сбоев;

• возможность совместной (параллельной работы) нескольких пользователей с данными.

Существуют различные СУБД от разных разработчиков ORACLE, Microsoft SQL Server, MYSQL, PostgreSQL и другие. Каждая СУБД имеет несколько версий. Обычно версия соответствует развитию технологии на некоторый момент времени. Например MS SQL Server 2017.

Microsoft SQL Server (MS SQL Server), – это масштабируемая высокопроизводительная система управления реляционными базами данных для платформ на базе MS Windows. Она разработана с учетом требований к современным распределенным клиент-серверным вычислениям и тесно интегрирована с серверными продуктами семейства Microsoft Office.

Включает в себя библиотеки и службы ядра сервера СУБД. При установке MS SQL SERVER система представляется в виде системной службы MSSQLSERVER. Данная служба все запросы, приходящие на сервер.

Отображение службы MSSQLSERVER в диспетчере задач операционной системы показано на рисунке 1.2. В данном случае Server EXPRESS с именем сервера EXPRESS208R2.

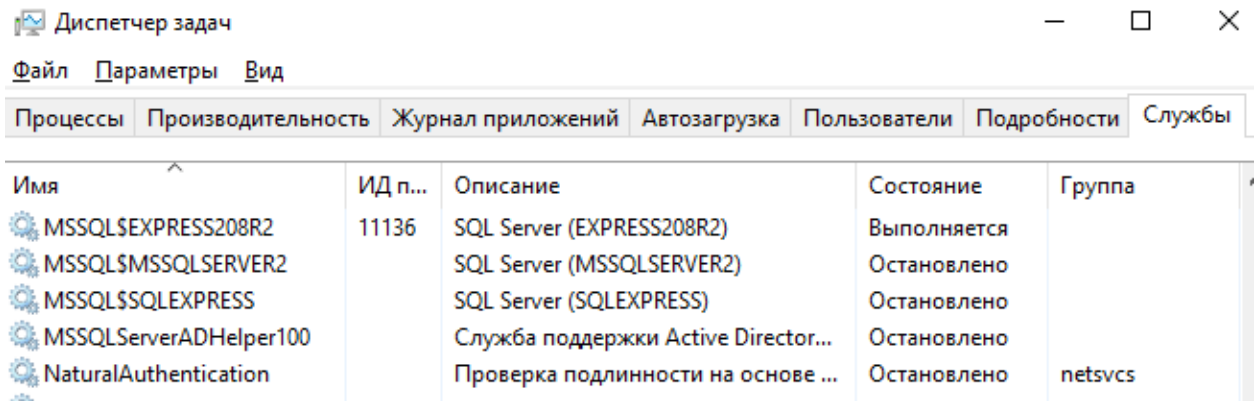


Рис.1.2 Служба MSSQLSERVER. В данном случае SQL Server EXPRESS с именем сервера EXPRESS208R2

В стандартный пакет Microsoft SQL Server входят несколько приложений, служащих для администрирования и разработки клиент-серверных приложений.

Для разработки таблиц и серверных механизмов используется приложение MS SQL SERVER Management Studio (также может быть различных версий).

При запуске приложения открывается окно соединения приложения с сервером. Приложение можно использовать для работы серверами, установленными не зависимо от MS SQL SERVER Management Studio.

CIT-208_01

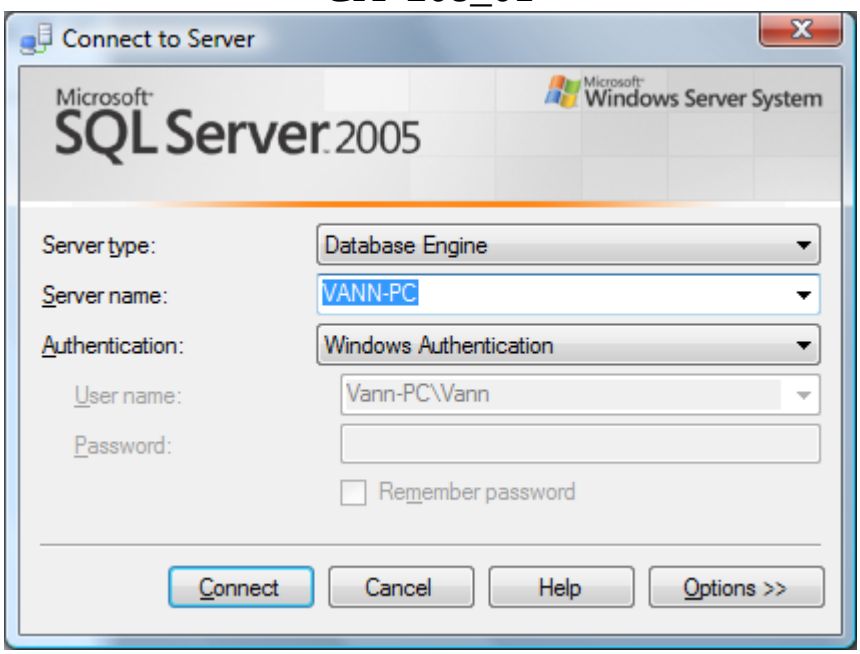


Рис.1.3 Окно соединения с сервером

Для соединения с сервером необходимо знать его имя, имя записи, зарегистрированной на сервере и пароль для этой записи. Если используется авторизация на основе учетной записи Windows, данная учетная запись должна быть зарегистрирована на сервере БД

После соединения с сервером открывается окно приложения MS SQL SERVER Management Studio.

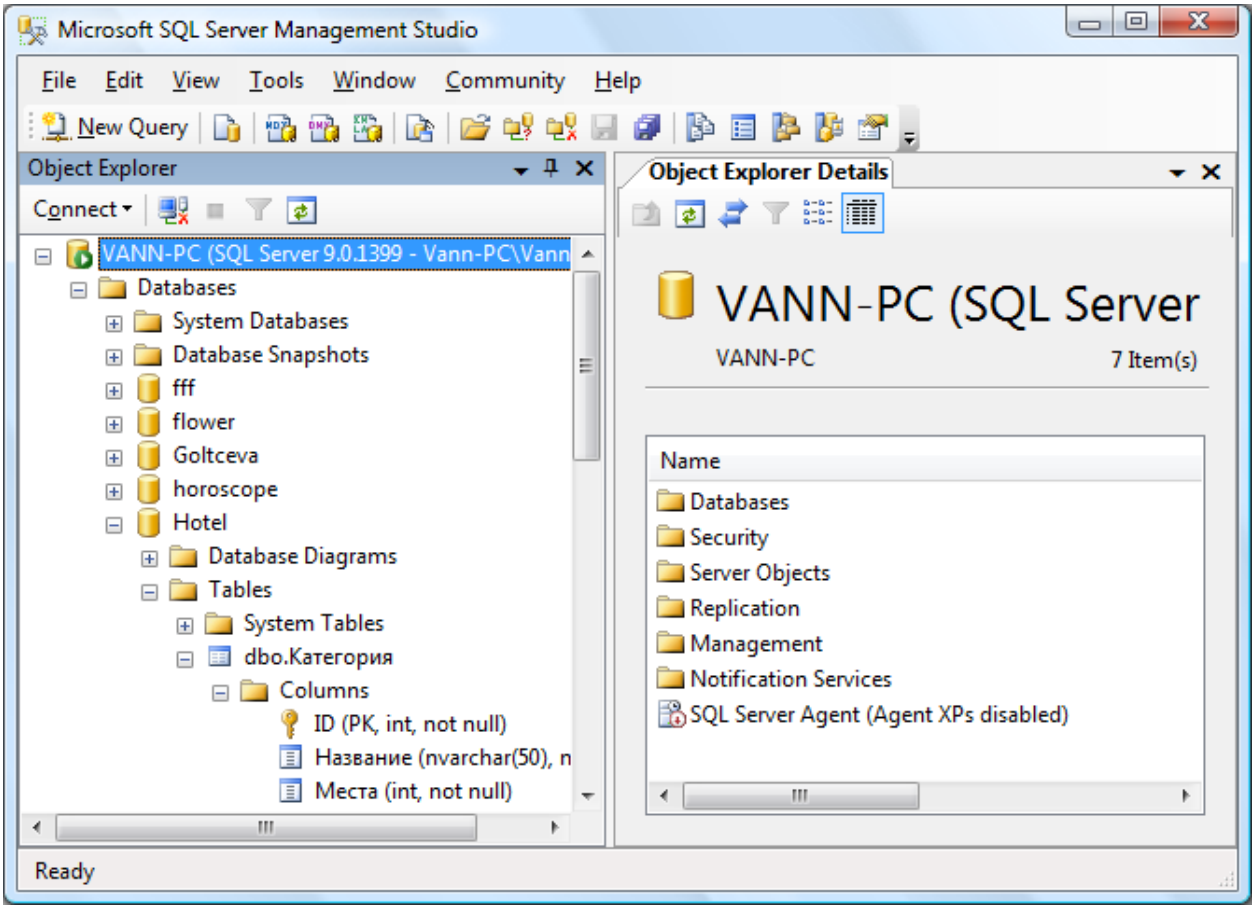


Рис.1.4 Рабочее окно MS SQL SERVER Management Studio

Левую часть окна занимает рабочее окно обозревателя объектов сервера. Объекты сервера представлены в виде древовидной структуры. Корнем дерева является соединение. Management Studio может быть одновременно соединено с несколькими серверами. Работа с любыми объектами сервера может осуществляться через контекстное меню на соответствующем узле дерева. База данных отображается в виде узла Databases. В среде MS SQL Server база данных содержит в себе различные типы объектов (рис. 1.5).

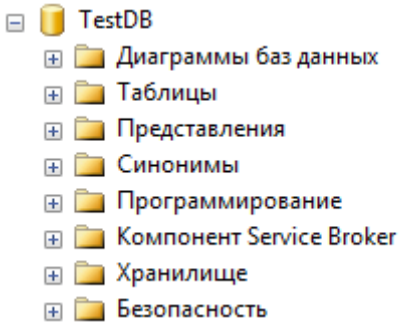


Рис.1.5 Объекты базы данных в среде MS SQL Server

Объекты базы данных в обозревателе объектов сервера сгруппированы в функциональные узлы. Выделяются следующие типы объектов:

- Таблицы – узел «таблицы».
- Представления – узел «Представления».
- Программные объекты (механизмы сервера) – узел «Программирование».
- Объекты обеспечения безопасности – узел «Безопасность».
- Диаграммы баз данных – узел «Диаграммы баз данных».

Все узлы создаются автоматически при создании базы данных.

Согласно рабатам основоположника теории реляционных баз данных Дейту [1] в базе данных выделяются структурная часть, манипуляционная и целостная.

Структурная часть базы данных – таблицы базы данных или реляционные отношения содержится в узле «Таблицы». Создать новую таблицу можно через контекстное меню на данном узле.

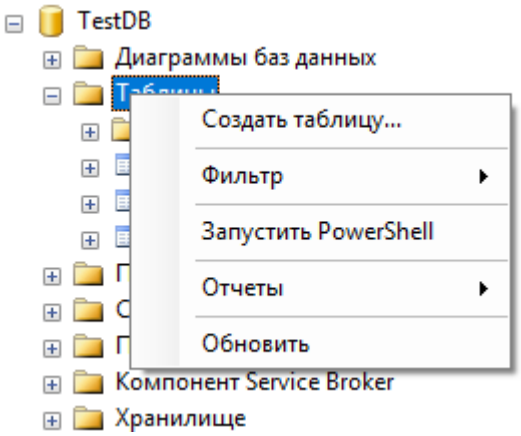


Рис.1.6 Создание новой таблицы в среде MS Management Studio

Создание таблицы подразумевает создание её атрибутов (столбцов) и присвоение имени таблицы.

После вызова команды создания таблицы в левой (рабочей области) Management Studio открывается табличная форма для создания и корректировки атрибутов таблицы.

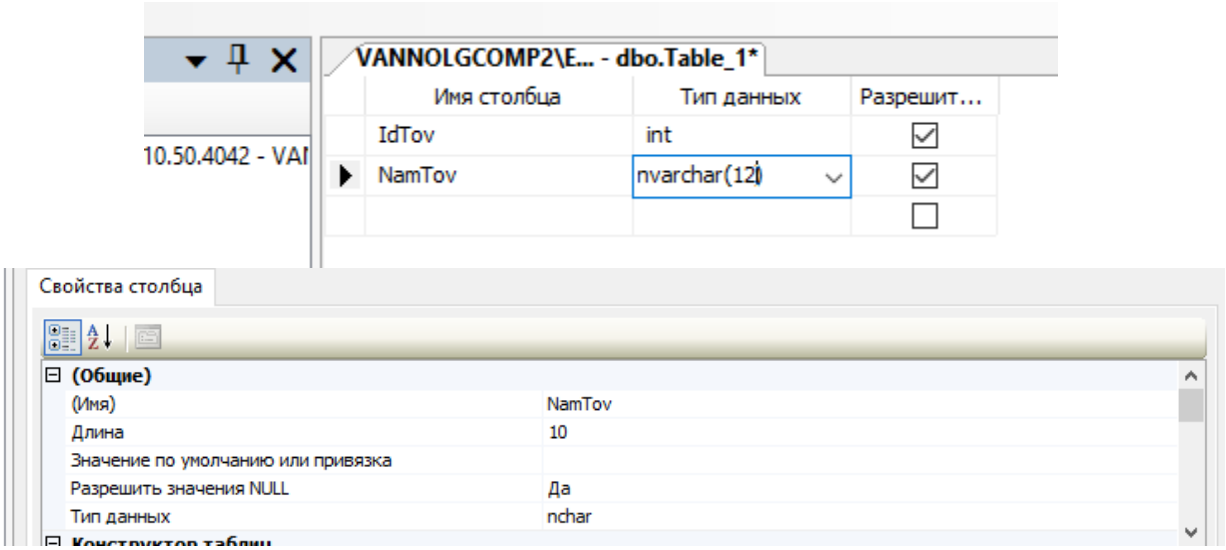


Рис.1.7 Работа с таблицей в режиме её модификации

При создании и модификации таблицы её атрибуты представляются в виде строк таблицы. Каждая строка соответствует отдельному столбцу (атрибуту).

Для каждого столбца необходимо указать его имя и тип данных. Имя можно выбрать любое, но для обеспечения простоты формирования запросов целесообразно для задания имён атрибутов использовать латинский шрифт и не использовать внутри имени пробелы и другие служебные символы. Пример хорошего имени столбца «NameStud» – то есть смысловые части разделяются заглавной буквой. Пример не рекомендуемого имени столбца – «Имя Студента». При использовании такого типа имени при написании запросов их придётся заключать в квадратные скобки. Например «*Select [Имя Студента] from [Студенты]*». Гораздо проще будет выглядеть запись той же команды при использовании рекомендованных именований – *«Select NameStud from Studs»*

Обычно таблицы имеют некоторые идентифицирующий ключевой атрибут и некоторую совокупность описательных атрибутов.

При задании столбцов (атрибутов) таблицы (отношения) могут использоваться различные типы данных, предусмотренных средой конкретного СУБД, в которой производится работа, используются следующие типы данных.

Используемые типы данных представлены на рис. 1.8.

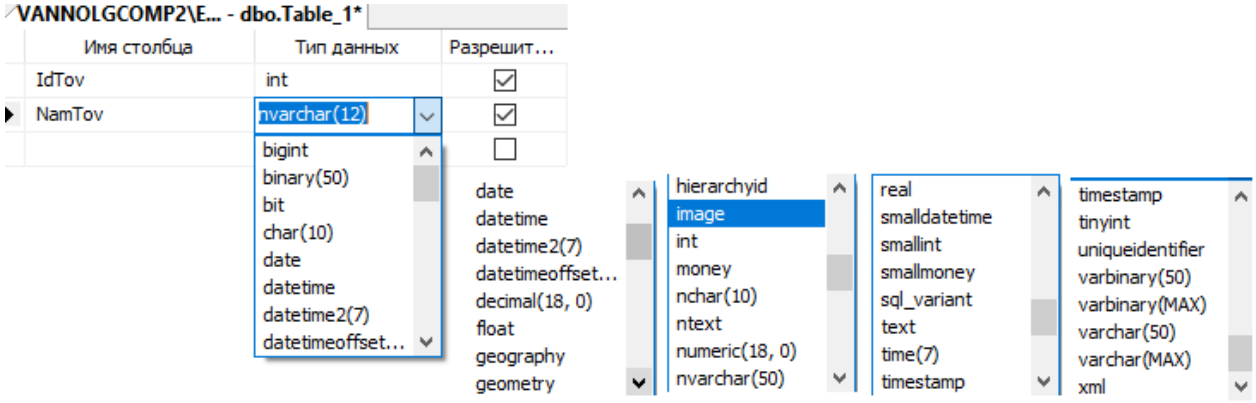


Рис.1.8 Задание типа данных для столбца (атрибута таблицы)

Тип данных выбирается с помощью соответствующего выпадающего списка.

В MS SQL Server 2008R2 объединены в следующие категории:

- Точные числа.
- Приблизительные числа.
- Символьные строки.
- Символьные строки в Юникоде.
- Дата и время.
- Двоичные данные.
- Прочие типы данных.

Точные числа:

- int – целые.
- tinyint – малые целые.
- smallint – малые целые.
- bigint – большие целые.
- numeric, decimal – числа с фиксированной точностью.
- bit – битовые числа.
- smallmoney, money – для работы с денежными величинами.
- float real – приблизительные числа

Типы данных для работы с датой и временем представлены следующими: *date, datetimeoffset, datetime2, smalldatetime, datetime, time.*

Символьные строки:

- char
- varchar
- text
- char [(n)]

- nchar
- nvarchar
- ntext
- nchar [(n)]

Двоичные данные:

- binary
- varbinary
- image

Прочие типы данных:

cursor, timestamp, hierarchyid, uniqueidentifier,
sql_variant, xml, table

Можно также определять собственные типы данных в Transact-SQL или Microsoft.NET Framework. Псевдонимы типов данных основываются на системных типах. Дополнительные сведения о псевдонимах типов данных см. в разделе

Внесение, изменить данных в таблице можно в среде Management Studio через команду «*Изменить первые 200 строк*», вызываемую через контекстное меню на редактируемой таблице.

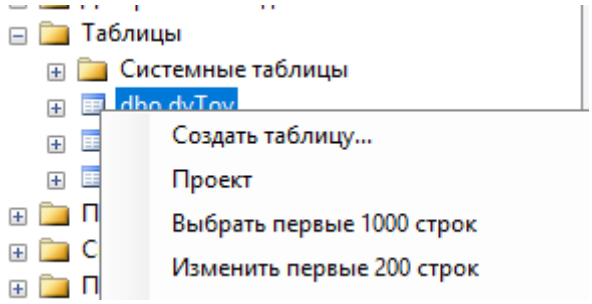


Рис.1.9 Вызов таблицы для изменения и внесения данных

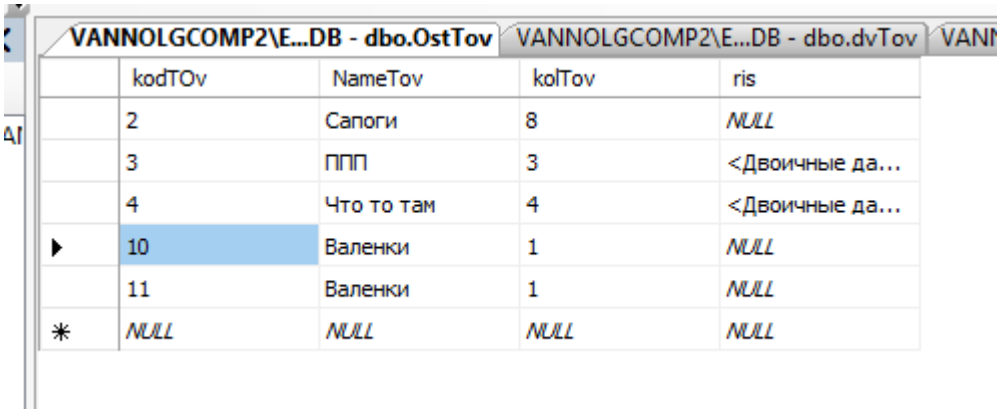


Рис.1.10 Вид таблицы вызванной для внесения данных
и редактирования

3. ЗАДАНИЕ ДЛЯ ВЫПОЛНЕНИЯ

Создать и заполнить данными таблицы в соответствие с вариантом задания.

Вариант 1.

- Студенты (Номер зачётки, Фамилия студента, Имя студента).
- Состав учебных группы (Наименование группы, Номер зачётки студента).

Вариант 2.

- Товары (наименование товара, код товара).
- Состав покупки (номер покупки, код товара, количество).

Вариант 3.

- Учебные предметы (наименование предмета, код предмета).
- Расписание (наименование группы, код предмета, дата начала).
- Кафедры университета (наименование кафедры, код кафедры).
- Учебные аудитории (номер аудитории, код кафедры).

Таблицы заполнить данными в среде MS Management Studio.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое база данных?
2. Базовые свойства реляционных отношений.
3. Что такое ключ реляционного отношения?
4. Как задаются связи между реляционными отношениями?

РАБОТА №2 ПРАКТИЧЕСКАЯ. ПОСТРОЕНИЕ СХЕМЫ И СЛОВАРЯ БАЗЫ ДАННЫХ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является получение практических навыков разработки схемы базы данных.

Задачами работы, обеспечивающими выполнение цели, являются:

- изучение принципов и получение практических навыков;
- выявления отношений в заданной предметной области;
- определение атрибутов отношений;
- выявление связей отношений;
- отображение связей отношений на диаграмме базы данных.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Схема базы данных (от англ. Database schema) – её структура, описанная на формальном языке, поддерживаемом СУБД. В реляционных базах данных схема определяет таблицы, поля в каждой таблице (обычно с указанием их названия, типа, обязательности), и ограничения целостности (первичный, потенциальные и внешние ключи и другие ограничения).

Схемы в общем случае хранятся в словаре данных. Хотя схема определена на языке базы данных в виде текста, термин часто используется для обозначения графического представления структуры базы данных.

Основными объектами графического представления схемы являются таблицы и связи, определяемые внешними ключами.

Выявление отношений в базе данных. В качестве отношений реляционной базы данных отображаются объекты предметной области, обеспечивающие получение информации, определенной в требованиях к системе.

Для выявления сущностей предметной области необходимо её проанализировать и выявить объекты, обладающие свойствами, на основе которых может быть получена информация, определённая в требованиях для базы данных. Состав объектов должен быть доста-

точным, но не избыточным. Обычно выделяются объекты оперативные и справочные.

Оперативные объекты содержат некоторую текущую информацию, они часто обновляются. Это могут быть данные о единичной покупке, например:

таблицаПокупка(покупатель, товар, количествоТовара).

Справочные объекты содержат информацию, которая может использоваться в качестве значений атрибутов для оперативных объектов. Например, данные о товаре:

таблицаТовар(Наименование, цена, производитель).

Атрибуты справочных таблиц могут определяться значениями, других справочных таблиц, например атрибут производитель в таблице *таблицаТовар* может определятся значениями таблицы:

таблицаПроизводитель(Наименование, номерСчёта, юрАдрес).

Для выявленных отношений устанавливаются атрибуты и требования к ним.

Для каждого отношения необходимо сформулировать бизнес – правила соответствующей предметной области.

Бизнес – правила характеризуют поведение объекта в предметной области, значение его атрибутов.

Необходимо проанализировать атрибуты, выявленные для отношений, на предмет их атомарности. Не атомарный атрибут подразумевает некоторое множество составных атрибутов, а, следовательно, его можно представить в виде другого отношения.

Например:

Студент – объект, выполняющий обучение на предметах.

Характеризуется:

фамилией, именем, отчеством (отдельные атрибуты типа строка);

номером зачетной книжки (атрибут целого типа).

Студент обучается на ***учебном курсе*** *(****учебный курс*** *– это отдельное отношение, так как может иметь свои характеристики).*

Для выявленных объектов и их атрибутов необходимо выявить бизнес правила, определяющие требования целостности сущности, то есть обязательность значения данного атрибута, уникальность значения данного атрибута, его допустимые значения.

Например:

Фамилия студента состоит из символов, это обязательный атрибут.

Номер зачетной книжки – число, минимальное значение - 10000,

Максимальное 99999.

Для выявленных отношений необходимо определить бизнес-правила их функционирования в предметной области определяющие их с другими отношениями

В бизнес-правилах, характеризующих связи должна быть дана следующая информация:

- содержания связи;
- множественность связи с одной и другой стороны;
- обязательности и дополнительных ограничений, ограничений накладываемых на связь.

Например, отношение Покупка связано с отношением товар, так как покупка должна всегда содержать товар. Данная связь имеет множественность «один к многим», так как одна покупка может содержать много товаров.

Каждое выявленное бизнес-правило реализуются в виде фрагмента ER диаграммы.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Выделить отношения согласно заданию, описать отношения, и их атрибуты.
2. Описать связи между отношениями с точки зрения их множественности с одной и другой стороны, обязательности, соответствия бизнес правилу предметной области.
3. Построить в среде SQL server Manadgment Sydudio таблицы в соответствии с заданием.
4. Построить диаграмму отношений в среде SQL server Manadgment Sydudio.
5. Произвести заполнение отношений тестовыми данными.

4. ЗАДАНИЯ

В качестве заданий выдается примерная формулировка темы для курса лабораторных работ, в результате которых должна быть создана база данных архитектуры «сервер баз данных», то есть база данных должна быть дополнена серверными механизмами для работы с ней. База данных, созданная на курсе лабораторных работ должна содержать не менее 4 таблиц.

Примеры вариантов заданий.

1. Разработка информационной системы обеспечения хранения, накопления и выборки данных о рейсах междугородних автобусов автовокзала.

2. Разработка информационной системы обеспечения хранения, накопления и выборки данных об охотничьих угодьях Кемеровской области, их ресурсах и выдаче лицензий на охоту.

3. Разработка информационной системы обеспечения хранения, накопления и выборки данных о садовых участках кемеровского района, расположении, владельцах, данные об участке, наименование кооператива, председатель кооператива.

4. Разработка информационной системы обеспечения хранения, накопления и выборки данных об аппаратном обеспечении персональных компьютеров и его поставщиков.

5. Разработка информационной системы обеспечения хранения, накопления и выборки данных о цветах, букетах цветочного магазина.

6. Разработка информационной системы обеспечения хранения, накопления и выборки данных о студентах, учебных группах, успеваемости (база деканат).

7. Разработка информационной системы обеспечения хранения, накопления и выборки данных о состоянии сданной в ремонт компьютерной техники.

8. Разработка информационной системы обеспечения хранения, накопления и выборки данных о тарифах и услугах сотовых операторов.

9. Разработка информационной системы обеспечения хранения, накопления и выборки данных о зоологических особенностях животных.

10. Разработка информационной системы обеспечения хранения, накопления и выборки данных о деятельности гостиницы. Клиенты, номера, проживание клиентов в номерах.

11. Разработка информационной системы обеспечения хранения накопление выборки данных материального обеспечения учебного процесса кафедры «Прикладная механика» КузГТУ.

12. Разработка информационной системы обеспечения хранения накопление выборку данных об индивидуальных прогнозах личностей (гороскоп).

13. Разработка информационной системы обеспечения хранения накопление выборку данных о музыкальных направлениях, и произведениях

14. Разработка информационной системы обеспечения хранения накопление выборку данных о содержании учебного процесса, учебном плане, программах курсов, расписании и их выполнении

15. Разработка информационной системы обеспечения хранения накопление выборки данных об Интернет-провайдерах, их услугах и пользователях.

16. Разработка информационной системы обеспечения хранения накопление выборки данных маршрутах средств общественного транспорта.

17. Разработка информационной системы обеспечения хранения накопление выборки данных о поставке, лицах, осуществляющих поставку и затратах на разгрузку товара в торговый комплекс «Палата».

18. Разработка информационной системы обеспечения хранения накопление выборки данных о соревнованиях Формула 1.

19. Разработка информационной системы обеспечения хранения накопление выборки данных об анкетировании студентов.

20. Разработка информационной системы обеспечения хранения накопление выборки данных о выполнении графика подготовки спортсмена-лыжника к соревнованиям.

21. Разработка информационной системы обеспечения хранения, накопления и выборки данных обеспечивающих работу агентства недвижимости.

22. Разработка информационной системы обеспечения хранения накопление выборки данных об игроках в футбол команд высшей и первой лиги.

23. Разработка информационной системы обеспечения хранения накопление выборки данных о лекарственных средствах, имеющихся в наличии в аптеках города.

24. Разработка информационной системы обеспечения хранения накопление выборки данных обеспечивающих работу автомагазина.

25. Разработка информационной системы обеспечения хранения накопление выборки данных об иероглифах и их сочетаниях китайского языка (китайский словарь)

26. Разработка информационной системы обеспечения хранения накопление выборки данных о музыкальных магазинах г. Кемерово, наличии в них аудио, видео дисках, их содержании и исполнителях.

27. Разработка информационной системы обеспечения хранения накопление выборки данных о странах мира, их основных характеристиках, граничащих странах.

28. Разработка информационной системы обеспечения хранения, накопление и выборки данных о результатах игр сезона по футболу.

29. Разработка информационной системы обеспечения хранения, накопление и выборки данных об игровом компьютерном клубе: игроки, игры, результаты.

30. Разработка информационной системы обеспечения хранения накопление выборки данных о делах, ведомых в ГУВД, фигурантах дел.

31. Разработка информационной системы обеспечения хранения накопление выборки данных о авто-аксессуарах, продаваемы в магазине.

32. Разработка информационной системы обеспечения хранения накопление выборки данных обеспечивающих работу автопредприятия, тип транспортного средства, грузоподъемность, состояние.

33. Разработка информационной системы обеспечения хранения, накопление и выборки данных о начислении зарплаты работникам предприятия. Работник. Дата. Начислено. Необходимо данные об отделах, в которых работают работники.

34. Разработка информационной системы обеспечения хранения, накопление и выборки данных о кулинарных рецептах.

35. Разработка информационной системы обеспечения хранения накопление выборки данных высаженных культурах, исполнителях, проведенных работах, истории посадок на садовом участке.

36. Разработка информационной системы обеспечения хранения накопление выборки данных о нотных записях и текстах музыкальных произведений.

37. Разработка информационной системы обеспечения хранения накопление выборки данных о чемпионате России по баскетболу.

38. Разработка информационной системы обеспечения хранения, накопление и выборки данных о репертуаре театра на сезон.

39. Разработка информационной системы обеспечения хранения, накопление и выборки данных о данных соревнованиях по велоспорту.

40. Разработка информационной системы обеспечения хранения, накопление и выборки данных о товарах ружейного магазина (характеристики оружия, боеприпасы, аксессуары).

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что подразумевается под схемой базы данных?
2. Каким образом выявляются отношения базы данных?
3. Чему соответствует таблица базы данных в предметной области?

РАБОТА №3 ПРАКТИЧЕСКАЯ. ИЗУЧЕНИЕ КОМАНД АДМИНИСТРИРОВАНИЯ ДАННЫХ ДЛЯ СРЕДЫ ОДНОЙ ИЗ СУБД

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Получение практических навыков администрирования в среде MS SQL SERVER.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В общем, *управление пользователями* представляет собой:

- процесс создания и удаления учетных записей пользователей;
- наделение пользователя привилегиями (удаление привилегий);
- наблюдение за действиями пользователей в рамках базы данных.

Стандарт SQL предлагает идентифицировать пользователей базы данных с помощью *идентификаторов разрешения доступа (Authorization Identifier* – authio). «*В соответствии со стандартом SQL, идентификатор разрешения доступа является именем, по которому система распознает пользователя базы данных»*.

В большинстве реализаций языка идентификаторы разрешения доступа называются просто *пользователями.* Могут использоваться термины «пользователь», «пользователь базы данных», «имя пользователя», «учетная запись пользователя».

Обычно выделяются следующие типы пользователей: Клерки, осуществляющие ввод данных (db_denydatawriter), Программисты, Системные инженеры, Администраторы баз данных Системные аналитики, Разработчики, Специалисты по тестированию, Управляющий персонал, Конечные пользователи (public).

Пользователи каждого из указанных типов решают при работе с базой данных свои задачи, и поэтому занимают разные места в иерархии базы данных, имея различные уровни доступа к ней.

Процесс управления пользователями выполняет Администратор базы данных, он выполняет следующие процессы:

- создает учетные записи пользователей;

- наделяет пользователей привилегиями;
- создает пользовательские профили;
- при необходимости удаляет учетные записи.

2.1. Создание учетных записей пользователей

Создание учетных записей пользователей осуществляется с помощью определенных команд SQL в рамках базы данных. Стандартных команд для создания пользователей нет – каждая реализация языка предлагает свои методы. Не зависимо от реализации, базовый подход остается одним и тем же.

Создание учетных записей пользователей в Sybase и Microsoft SQL Server

Последовательность шагов, которые необходимо выполнить при создании учетной записи пользователя в базе данных Sybase или Microsoft SQL Server, должна быть следующей.

1. Создание имени пользователя базы данных SQL Server с указанием пароля и базы данных для доступа.

2. Добавление пользователя в соответствующую базу данных.

Учетная запись пользователя создается оператором следующего вида.

```
SP_ADDLOGIN ИМЯ_ПОЛЬЗОВАТЕЛЯ, ПАРОЛЬ, [, БАЗА_ДАННЫХ ]
```

В базу данных пользователь добавляется с помощью оператора следующего вида.

```
SP_ADDUSER ИМЯ_ПОЛЬЗОВАТЕЛЯ [, ИМЯ_В_БД [, ИМЯ_ГРУППЫ } ]
```

В среде Mamager эти действия обычно выполняются через контекстное меню.

Удаление учетной записи пользователя из базы данных или ликвидация возможности его доступа к данным осуществляется парой простых команд. Однако, следует отметить, для этих команд отсутствует стандарт.

Ликвидировать доступ к базе данных пользователю можно:

- изменением пароля пользователя;
- отменой ранее разрешенных пользователю привилегий доступа к данным.

В некоторых реализациях SQL для удаления учетной записи пользователя из базы данных может использоваться команда DROP

```
DROP USER ИМЯ_ПОЛЬЗОВАТЕЛЯ [ CASCADE ]
```

2.2. Привилегии

Привилегии – это уровни полномочий, предоставленных пользователю при доступе к самой базе данных и ее объектам, при манипуляции данными и при выполнении в рамках базы данных различных административных функций. Привилегии предоставляются посредством команды GRANT и отменяются посредством команды REVOKE. Выделяются следующие типы привилегий:

- привилегии доступа к системе;
- привилегии доступа к данным.

Наделение пользователя необходимыми привилегиями доступа осуществляется с помощью оператора следующего вида.

```
GRANT PRIV1 [ , PRIV2, ... ] TO ИМЯ_ПОЛЬЗОВАТЕЛЯ
```

2.3. Привилегии доступа к системе

Привилегии доступа к системе – это привилегии, дающие возможность пользователю решать в рамках базы данных административные задачи типа создания и удаления баз данных, учетных записей пользователей, изменения и удаления различных объектов базы данных, изменения состояния объектов, изменения состояния базы данных и других подобных операций, несущих в себе при недостаточной внимательности потенциальную опасность для базы данных в целом.

Предлагаемые разными производителями баз данных привилегии доступа к системе сильно отличаются.

Некоторых из привилегий доступа к системе, которые предлагаются в рамках Sybase.

```
CREATE DATABASE
CREATE DEFAULT
CREATE PROCEDURE
CREATE RULE
DUMP DATABASE
DUMP TRANSACTION
EXECUTE
```

2.4. Привилегии доступа к объектам

Привилегии доступа к объектам – это уровни полномочий, предоставленных пользователю при работе с объектами базы данных, и это значит, что для выполнения определенных операций с объектами базы данных пользователю требуется предоставить соответствующие привилегии. Например, чтобы извлечь данные из таблицы другого пользователя, следует сначала получить право доступа к его данным. Привилегии доступа к объектам предоставляются пользователям базы данных владельцами объектов. Напоминаем, что владельца объекта называют также владельцем схемы.

Стандарт ANSI определяет следующие привилегии доступа к объектам.

USAGE. Разрешает использование заданной области.

SELECT. Разрешает доступ к заданной таблице.

INSERT (имя_столбца). Позволяет разместить данные в указанном столбце заданной таблицы.

INSERT. Позволяет поместить данные во все столбцы заданной таблицы.

UPDATE {имя_столбца). Позволяет изменить данные в указанном столбце заданной таблицы.

UPDATE. Позволяет изменить данные во всех столбцах заданной таблицы.

REFERENCES (имя_столбца). Позволяет сослаться в условиях целостности на указанный столбецзаданной таблицы; требуется для всех условий целостности.

REFERENCES, позволяет сослаться в условиях целостности на любой столбец заданной таблицы.

Владелец объекта автоматически наделяется всеми привилегиями относительно этого объекта. Такие привилегии могут быть разрешены также имеющейся в некоторых реализациях языка очень удобной командой GRANT OPTION, которая будет обсуждаться ниже.

Именно привилегии доступа к объектам используются для разрешения или ограничения доступа к объектам данной схемы. Эти привилегии можно использовать для защиты объектов одной схемы от доступа пользователей базы данных, имеющих право доступа к объектам другой схемы той же базы данных.

Обычно право использовать команды GRANT и REVOKE имеет администратор базы данных, но если есть администратор по безопасности, то он тоже может иметь право использовать эти команды. Конкретные инструкции по поводу того, кому и какие именно привилегии следует назначить или отменить, должны исходить от руководства и желательно в письменном виде.

Привилегии доступа к объекту должен распределять владелец этого объекта. Даже администратор базы данных не имеет права давать разрешение на использование не принадлежащего ему объекта, хотя, конечно, администратор всегда имеет реальную возможность это сделать.

2.5. Команда GRANT

Команда GRANT используется для предоставления привилегий как на уровне доступа к системе, так и на уровне доступа к объектам тем пользователям, которые уже имеют учетные записи в базе данных.

Синтаксис оператора следующий.

```
GRANT Привилегия1 [, Привилегия2 ] [ ON Объект ] TO Имя_Пользователя [ WITH GRANT OPTION | ADMIN OPTION ]
```

Одну привилегию пользователю можно предоставить следующим образом.

```
GRANT SELECT ON EMPLOYEE_TBL TO USER1;
```

Право предоставлено.

Несколько привилегий пользователю можно предоставить следующим образом.

```
GRANT SELECT, INSERT ON EMPLOYEE_TBL TO USER1;
```

Право предоставлено.

В случае предоставления пользователю нескольких привилегий в рамках одного оператора привилегии в списке разделяются запятыми.

Нескольким пользователям привилегии предоставляются следующим образом.

```
GRANT SELECT, INSERT ON EMPLOYEE_TBL TO USER1, USER2;
```

Опция GRANT OPTION команды GRANT является достаточно мощной. Если владелец объекта предоставляет привилегии относительно объекта другому пользователю и использует при этом опцию GRANT OPTION, это значит, что последний получает право пре-

доставлять другим привилегии использования объекта, не являясь при этом владельцем объекта. Вот пример использования опции:

GRANT SELECT ON EMPLOYEE_TBL TO USER1 WITH GRANT OPTION;

Право предоставлено.

ADMIN OPTION

Опция ADMIN OPTION команды GRANT подобна опции GRANT OPTION в том, что получающий привилегии пользователь наследует также и право предоставлять эти привилегии другим пользователям. Но GRANT OPTION используется для привилегий на уровне объектов, а ADMIN OPTION – на уровне системы.

Команда REVOKE отменяет привилегии, ранее предоставленные пользователю базы данных. Команда REVOKE имеет две опции – RESTRICT и CASCADE. При использовании опции RESTRICT команда REVOKE будет успешно завершена только в том случае, когда отсутствуют другие пользователи с оставшимися привилегиями, явно указанными оператором REVOKE. С помощью опции CASCADE отменяются и все оставшиеся привилегии других пользователей. Другими словами, если владелец объекта наделил пользователя USERI привилегиями с опцией GRANT OPTION, а пользователь USER1 наделил привилегиями пользователя USER2, то при отмене владельцем привилегий пользователя USER1 с опцией CASCADE будут автоматически отменены и соответствующие привилегии пользователя USER2

Синтаксис оператора для отмены привилегий следующий.

REVOKE Привилегия! [, Привилегия2] [GRANT OPTION FOR] ON Объект

FROM Имя_Пользователя { RESTRICT | CASCADE }

Вот пример использования подобного оператора.

REVOKE INSERT ON EMPLOYEE_TBL FROM USERI;

Право отменено.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создать роли в соответствие с заданием.
2. Наделить роли полномочиями.
3. Создать пользователей в соответствие с заданием.

4. Наделить пользователей полномочиями в соответствии с заданиям.

5. Выполнить проверку выполнения привилегий пользователей.

6. Команды создания заданных административных команд отобразить в отчёте.

4. ЗАДАНИЯ

Роли	Объекты администрирования	Пользователи
Роль1	Доступ ко всем таблицам базы данных. Кроме хранимых процедур	ПочтиАдмин1
Роль2	Доступ ко всем таблицам базы данных	НеАдмин1 НеАдмин2

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое пользователь?
2. Что такое имя входа, как оно задаётся?
3. Что такое привилегия? Как они назначаются?
4. Для чего используется объект «Роль»? С помощью каких команд создаётся?

РАБОТА №4 ЛАБОРАТОРНАЯ. РАЗРАБОТКА ТРЕБОВАНИЙ И КОНФИГУРИРОВАНИЕ КОРПОРАТИВНОЙ СЕТИ

1. ЦЕЛЬ РАБОТЫ

Получить практический навык создания заполнения объектов БД средствами языка SQL.

В связи с этим задачами работы являются:

- изучение основных операторов языка определения и манипулирования данными;
- создание и выполнение запросов в среде Management Studio, обеспечивающих создание и заполнение таблиц данными.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ.

SQL (Structured Query Language, структурированный язык запросов) – это язык программирования, предназначенный для выборки и обработки информации, содержащейся в реляционной базе данных. SQL является стандартным языком для работы с реляционными базами данных, его основа реляционная алгебра и реляционное исчисление. SQL содержит набор стандартных операторов доступа к данными и манипулирования.

Существуют следующие версии SQL:

- SQL1 (принята в 1986 году, дополнена в 1989 году стандарт ANSI);
- SQL2 (SQL-92 принята в 1992 году);
- SQL3 (SQL-99) расширяет SQL2 за счет включения объектно-реляционных инструментов и новых функциональных возможностей.

Существуют версии SQL предлагаемые основными поставщиками СУБД, они, как правило, удовлетворяют требованиям ANSI, реализуют многие возможности SQL2 и имеют некоторые особенности.

SQL – это слабо структурированный язык, особенно по сравнению с такими высокоструктурированными языками, как C, Pascal или Java. В нем нет инструкции IF..THEN для проверки условий,

нет инструкции GOTO для организации переходов и нет инструкций DO или FOR для создания циклов.

SQL обеспечивает независимость от конкретных СУБД: реляционную базу данных и программы, которые с ней работают, можно перенести с одной СУБД на другую с минимальными доработками и переподготовкой персонала. Все ведущие поставщики СУБД используют SQL.

Поставщики СУБД предлагают различные диалекты SQL позволяющие создавать самостоятельные программные модули, например, PL/SQL и Transact-SQL. В этих диалектах стандартный SQL дополнен инструкциями IF..THEN, GOTO и др., однако эти диалекты не получили статус стандарта и являются частными разработками отдельных компаний (PL/SQL применяется в СУБД Oracle, а Transact SQL – в СУБД MS SQL Server).

SQL не является отдельным программным продуктом. SQL – это неотъемлемая часть СУБД ее *Манипуляционная часть*, инструмент, с помощью которого осуществляется связь пользователя с БД.

Различают несколько групп операторов (подъязыки):

I. Язык определения данных DDL.

К языку запросов относятся операторы

CREATE TABLE – создания нового отношения;

DROP TABLE – удаление отношения;

ALTER TABLE – изменение структуры таблицы;

CREATE VIEW – создания представления;

DROP VIEW – удаления представления;

CREATE INDEX – удаление индексов.

II. Язык манипулирования данными DM (команды, DELETE, INSERT, UPDATE)

III. Язык запросов DQL (оператор SELECT)

IV. Средства управления транзакциями.

V. Средства администрирования данными.

Используемы в SQL типы данных аналогичны применяемым в других языках программирования (смотрим практическую работу 1)

Значение NULL и его применение.

Атрибутам отношения или переменным SQL допускает присвоение специального значения NULL.

Значение NULL имеет следующий смысл:

- значение не известно, то есть когда создается новый кортеж, а значение некоторого атрибута явно не задается и не заданно по умолчанию, то ему присваивается данное значение;
- значение не может быть заданно, то есть когда значения некоторого атрибута быть не может (например, атрибут Супруг, для некоторого кортежа отношения Сотрудники, когда рассматриваемый сотрудник не женат);
- значение умалчивается, то есть, если значение атрибута, выдается по запросу, но данный атрибут запрещен для просмотра для данного источника запроса.

Если атрибут или выражение со значением NULL участвует в арифметической операции, то результат операции будет иметь значение NULL.

При сравнении выражения, имеющего значение NULL с другим выражением с помощью операций сравнения (=, !=, <>, <, >, >=, <=, !> ,!<) результат будет иметь значение *unknown*.

Для проверки выражения на значение NULL операция сравнения не используется. Для Этого необходимо использовать специальный предикат IS NULL (IS NOT NULL), он будет рассмотрен ниже.

Операторы языка определения данных ddl и модификации отношений

Оператор задания схемы отношения, то есть, создаёт отношения (таблицы) – и их атрибуты.

Общий формат оператора

Create table ИмяОтн(Atr1 ТипАтр [ЗначПоУмолч] [*ОгрАтр*] *…, Atr2 …]*, … ОгрКортежа),

где

- *Atr1, Atr2* … – идентификаторы (имена) атрибутов отношения;
- *ЗначПоУмолч* – значение, присваиваемое атрибуту по умолчанию;
- *ОгрАтр* – ограничения на значение атрибута (будут рассмотрены позже;

- *ОгрКортежа* – ограничения на значение корежа (будут рассмотрены позже;

Оператор удаление отношения

Drop table ИмяОтн

Модификация отношений

Модификация отношения может быть следующих разновидностей:

- удаление атрибута

Alter table ИмяОтн drop ИмяАтр1 – из отношения ОТН будет удалён атрибут именем *ИмяАтр1*;

- вставка атрибута

Alter table *ИмяОтн* add *ИмяАтр1 типАтр1 ЗначПоУмолч].. ,*

где *ИмяАтр1 типАтр1 [ЗначПоУмолч]..*– описание атрибута, аналогичное используемому в операторе *Create table*.

Операторы SQL манипулирования данными

Операторы данной группы позволяют модифицировать существующие кортежи отношений. То есть, вставлять новые кортежи отношения, удалять, изменять значений атрибутов.

Вставка кортежей (INSERT)

insert into ИмяОтн(ИмяАтр1, ИмяАтр2, ..) values(знач1, знач2,…)

В результате выполнения данной команды в отношение с именем *ИмяОтн* будет вставлен кортеж, при этом атрибутам с именами *ИмяАтр1, ИмяАтр2*... будут присвоены значения *знач1, знач2,…* Атрибутам, не перечисленным в списке, будет присвоено значение по умолчанию. Если значения по умолчанию не заданы, то система попытается присвоить им значения NULL.

Удаление кортежей

delete from *имяОтн* where *условие*

При выполнении данной команды из отношения с именем *имяОтн* будут удалены кортежи, значение атрибутов которых будет соответствовать условию.

Модификация (обновление) кортежей

update ИмяОтн set ИмяАтр1= знач1, ИмяАтр1= знач1 [,…] where условие

При выполнении данной команды кортежам, отвечающим заданным условиям будет изменено значение заданных атрибутов.

Изменение атрибутов будет отменено, если они противоречат условиям целостности базы данных, или другим ограничениям.

Для работы с объектами СУБД MS SQL Server Menegment Studio предоставляет набор команд контекстного меню, в частности для работы с таблицами может использоваться иерархия контекстных меню, показанная на рис. 2.1.

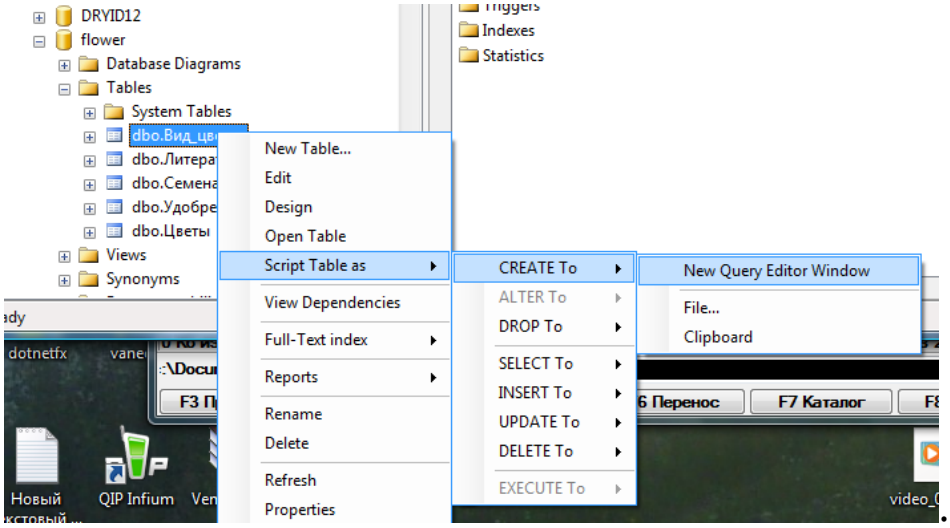


Рис. 2.1 Иерархия команд контекстного меню для работы с таблицами

Команда меню Script table as позволяет создавать шаблоны команд для создания, удаления, вставки кортежей и для выполнения других действий с таблицами.

Созданные команды можно сохранять во внешних файлах, с расширением SQL.

3. ПОРЯДОК ВЫПОЛНЕНИЯ И ЗАДАНИЕ

Заданием на лабораторную работу является формирование команд SQL, обеспечивающих создание таблиц, аналогичных полученным в результате выполнения лабораторной работы №1 и заполнение этих таблиц данными. Для создания команд можно воспользоваться шаблонами, получаемыми на с помощью контекстного меню.

Примерный порядок выполнения работы может быть следующим.

1. Создать запросы для создания таблиц с использованием контекстного меню соответствующей таблицы. Скорректировать запросы, изменив имя таблицы (можно добавить слеш в конце имени). Например, старая таблица Nomeklatura, создаваемая – Nomenklatura_

2. Создать запросы для вставки в таблицы значений. Использовать контекстное меню для создания запросов, при необходимости вопросы нужно скорректировать, так чтобы в таблицы вводились тестовые значения, введенные в аналогичных существующих таблицах.

3. Сформировать два файла запросов, первый содержит команды по созданию таблиц, второй по заполнению.

4. Создать файл, содержащий команды по удалению созданных таблиц.

5. Выполнить запросы создания, заполнения. Показать преподавателю результаты. Выполнить запрос по удалению. Показать преподавателю результаты.

6. Подготовить отчет. В отчет включить описание процесса создания запросов и тексты запросов и скриншоты отображающие структуру и состав полученных таблиц.

3. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите основные операторы языка определения данных.
2. Формат оператора SQL, используемого для создания отношений.
3. Какие операторы SQL позволяют менять состав атрибутов отношений?
4. Основные операторы языка модификации отношений.
5. Каким образом с помощью операторов SQL изменить тип атрибута в заполненной таблице?

РАБОТА №5 ЛАБОРАТОРНАЯ. РАЗРАБОТКА МЕХАНИЗМОВ СЕРВЕРА БАЗ ДАННЫХ. ХРАМИНЫЕ ПРОЦЕДУРЫ

1. ЦЕЛЬ РАБОТЫ

Получить практических навык организации бизнес-логики на стороне сервера, на основе использования хранимых процедур и пользовательских функций.

В связи с этим задачами работы являются:

1. Изучение принципов работы и создание хранимых процедур и пользовательских функций в среде MS SQL Server.

2. Создание хранимых процедур и пользовательских функций в соответствии с заданием к лабораторной работе.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Для реализации бизнес логики на стороне сервера используются, так называемые «механизмы сервера», включающие хранимые процедуры, пользовательские функции, триггеры, и другие объекты их состав может быть различен для СУБД различных производителей.

Наиболее распространенное средство реализации бизнес-логики – хранимые процедуры сервера.

Хранимые процедуры – это модули, хранящиеся непосредственно в базе данных в откомпилированном виде и которые могут запускаться пользователями или приложениями, работающими с базой данных. Хранимые процедуры обычно пишутся либо на специальном процедурном расширении языка SQL (например, PL/SQL для ORACLE или Transact-SQL для MS SQL Server), или на некотором универсальном языке программирования, например, C++, с включением в код операторов SQL в соответствии со специальными правилами такого включения. Основное назначение хранимых процедур – реализация бизнес-процессов предметной области на стороне сервера.

Процедура подразумевает выполнение не только обычных запросов SQL, но и более сложную обработку, связанную с вычислениями, переходами, сравнениями ветвлениями. Для этого существу-

ет расширение SQL включающее описание переменных, операторы ветвления, циклы и пр.

Создание хранимой процедуры

```
CREATE PROCedure [владелец.]имя_процедуры
  [@параметр1],
  [@параметр2]
AS
Операторы_SQL
```

где параметр – `@имя_параматра тип_данных [=default] [OUTPUT];`

`[=default]` – значение по умолчанию присеваемое параметру, при отсутствии передаваемого значения;

`[OUTPUT]` – ключевое слово, указывающее, что при изменении параметра в ходе выполнения процедуры новое значение возвращается в переменную, используемую для вызова этой процедуры.

Процедура может быть создана только в текущей базе данных, за исключением временных процедур, которые создаются в tempdb. Для создания временных процедур следует начинать ее имя с «#» или «##». Длина имени хранимой процедуры вместе с ## не может превышать 20 символов. Одна процедура может вызывать другую процедуру, уровень вложенности не может превышать 16, текущий уровень вложенности можно узнать из глобальной переменной @@NESTLEVEL

Пользователь может создавать свои системные процедуры; они начинаются с символов sp_. При попытке выполнения такой процедуры она сначала ищется в текущей базе данных, в случае же неудачи – в базе данных master. Таблицы, используемые в системной процедуре, определяемой пользователем, также сначала отыскиваются в текущей базе данных, и если это не удалось – в базе данных master.

Вызов хранимой процедуры осуществляется оператором EXEC, например:

```
EXEC имяПроцедуры [параметр]
```

В виде параметра может выступать как переменная, так и константа.

Пример:

Константа:

```
EXEC СводСотрудников «Инженер»
```

Использование переменной в качестве параметра:

```
DECLARE @Должность INT
SET @Должность = «Инженер»
EXEC СводСотрудников @Должность
```

Изменение хранимой процедуры

```
ALTER PROCedure [владелец.]имя_процедуры
  [@параметр1],
  [@параметр2]
AS
 Операторы_SQL
```

Удаление хранимой процедуры

```
DROP PROCEDURE имя_процедуры
```

Использование переменных в хранимых процедурах

Для использования переменных их необходимо объявить с помощью ключевого слова DECLARE

```
DECLARE имяПерем типДаных
```

где имяПерем – идентификатор переменной. Он должен быть без пробелов. В качестве первого символа должен использоваться символ «@». В среде SQL server могут быть созданы глобальные переменные, видимые на уровне сервера, данные переменные начинаются с двух символов «@@». Локальные переменные, начинающиеся с одного символа «@» являются локальные и видимы только внутри одной транзакции.

Для присвоения значений переменным используется оператор Set или Select.

Формат операторов SET и Select при присвоении значений.

```
SET имяПерем = значение
```

или

```
SELECT имяПерем значение
```

Для обращения к значению переменной в запросе используется та же конструкция SELECT

```
SELECT имяПерем as синоним.
```

Основные алгоритмические конструкции процедурного расширения языка SQL

Конструкция ветвления в SQL

```
IF логическоеВыражение
  {операторSQL | блокОпреаторовSQL}
[ ELSE
  { операторSQL | блокОпреаторовSQL }]
Конструкция цикла
WHILE <логическое_выражение>
  { операторSQL | блокОпреаторовSQL }
  [ BREAK ]
  { операторSQL | блокОпреаторовSQL }
  [ CONTINUE ]
```

где `блокОпреаторовSQL` – это операторные блоки, используемые для группировки операторов в скобки BEGIN END, аналогично паскалю, бейсику, или «{}» в C++

Для вывода результатов вычислений или значений переменных можно использовать оператор `print`

```
Print  выражение
```

T-SQL имеет набор функций, позволяющих выполнять набор математических, строковых, системных операций и операций преобразования (См. приложение).

Преимущества использования процедур

Использовать хранимые процедуры целесообразнее, чем отдельные операторы SQL по следующим причинам:

- Операторы хранимой процедуры всегда находятся в базе данных.
- Операторы хранимой процедуры уже проверены и находятся в готовом для использования виде.
- Возможность использования процедур позволяет использовать модульное программирование.
- Хранимые процедуры могут вызывать другие процедуры и функции.
- Сохраненные процедуры могут вызываться любыми приложениями.
- При использовании сохраненных процедур результат ответ от базы данных как правило получается быстрее.
- Использование процедур принципиально упрощено в СУБД.

Пользовательские функции UDF (user-defined function)

Основное отличие UDF от хранимых процедур заключается в том, что функция обязательно должна вернуть хотя бы какое-то значение. К сожалению, UDF имеют некоторые ограничения. Нельзя, например, изменять данные в таблицах БД, выводить данные с помощью команд print и select. Внутри UDF нельзя использовать недетерминированные функции типа GETDATE(). В SQL Server имеется 3 типа пользовательских функций: Scalar, Multi-statement, InLine.

Скалярная функция может возвращать значения любого скалярного типа данных, кроме данных типа text, image, table. Такая функция может объединять несколько команд языка T-SQL, находящихся в блоке `BEGIN…END`, например,

```
CREATE FUNCTION FunState
(@a varchar(20))
RETURNS varchar(20)
as
BEGIN
```

```
IF @a IS NULL
SET @a = «Not Applicable»
RETURN @a
END
```

Обратиться к такой функции можно с помощью конструкции select, указав вместо поля таблицы значение функции с параметром:

```
SELECT pub_name, City, dbo.FunState(state) AS State
FROM dbo.publishers
```

Multi-Statement Table

Другой тип UDF – Multi-Statement Table-valued Function. Как следует из названия, этот тип функции возвращает тип данных table. Тело такой функции может быть достаточно сложным и включать множество операторов, находящихся между ключевыми словами BEGIN…END. В данном простом примере в БД Pubs создается функция, которая может возвращать фамилию и имя либо автора книги, либо служащего издательства.

```
CREATE FUNCTION FunMult
(@b nvarchar(8))
RETURNS @Fun_Auth table
([First Name] nvarchar(80) not null,
[Last Name]nvarchar(80) not null)
AS
BEGIN
IF @b = «author»
INSERT @Fun_Auth SELECT au_fname, au_lname
FROM authors
ELSE IF @b = «employee»
INSERT @Fun_Auth SELECT fname, lname
FROM employee
RETURN
END
```

В зависимости от входного параметра, указываемого в конструкции select при вызове функции, получаем разный результат, обращаясь к таблице author или к employee:

```
SELECT * FROM dbo.FunMult(«author»)
```

```
SELECT * FROM dbo.FunMult(«employee»)
```

Функция InLine тоже возвращает значение типа table, но отличается от Multi-Statement Table-valued тем, что может состоять только из одной команды select.

```
CREATE FUNCTION FunInLine
@State nvarchar(30))
RETURNS table
AS
RETURN ( SELECT pub_name, city
FROM Pubs.dbo.publishers
WHERE state = @State
)
```

Обращение к функции происходит в предложении from конструкции select:

```
SELECT * FROM FunInLine(«Texas»)
```

Удаление функции

```
DROP FUNCTION Имя функции
```

Особенностью функции InLine является то, что код функции при выполнении программы вставляется непосредственно в исполняемый набор команд. Другими словами, происходит не вызов функции, а встраивание.

В среде SQL server хранимую процедуру или функцию можно создать в приложении Management Studio в разделе хранимых процедур с помощью контекстного меню. Или выполнением запроса соответствующего содержания из любого доступного приложения, например в Query Analyzer.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Уточнить задание применительно к сформированным отношениям.

2. Схематично описать необходимые запросы по заданию для своей БД.

3. Утвердить у преподавателя эскиз лабораторной работы.

4. Создать хранимые процедуры и функции с использованием описанных операторов в методичке (также использовать по одной из стандартных функций по выбору преподавателя).

5. Подготовить и предоставить контрольные примеры с использованием хранимых процедур.

4. ЗАДАНИЯ

1. Создать хранимую процедуру, в которой будет выполняться гибкий запрос на группировку по различным полям в зависимости от входного параметра

2. Создать пользовательскую функцию, которая возвращает значение, вычисленное, на основе одного или нескольких атрибутов отношения по заданным критериям поиска.

3. Создать хранимую процедуру, использующую созданную пользовательскую функцию.

4. Создать хранимые процедуры для вставки, изменения и удаления данных, переданных через входные параметры.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое хранимая процедура, общий формат описания и вызов?

2. Каков формат описания переменных в хранимых процедурах?

3. Тип параметров функций и процедур.

4. Описать типы пользовательских функций.

5. Формат описания конструкций цикла и ветвления.

6. Чем отличается хранимая процедура от пользовательской функции?

7. Что дает использование ХП при разработке БД?

РАБОТА №6 ЛАБОРАТОРНАЯ. РАЗРАБОТКА МЕХАНИЗМОВ СЕРВЕРА БАЗ ДАННЫХ. ТРИГГЕРЫ

1. ЦЕЛЬ РАБОТЫ

Получить практический навык использования триггеров для организации бизнес-логики на стороне сервера.

В связи с этим задачами работы являются:

- Изучить принципы работы и особенности построения триггеров в среде MS SQL Server.
- Создать триггеры, выполняющие действия, согласно индивидуальному заданию.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Триггер – это откомпилированная процедура, используемая для выполнения действий, инициируемых происходящими в базе данных событиями. Такими событиями являются запросы к базе данным, генерируемые операторами языка манипуляции данными – DML (INSERT, DELETE или UPDATE). Триггер может выполняться вместо или после операторов. С помощью триггеров можно отменять транзакции, а также модифицировать данные одних таблиц и читать данные других даже из других баз данных. Таким образом, результатом работы триггеров является обеспечение целостности базы данных.

Чаще всего триггеры использовать очень удобно, однако, их использование приводит к значительному увеличению числа операций ввода-вывода. Триггеры не следует использовать тогда, когда сохраненная процедура или программа может добиться тех же результатов с меньшими накладными расходами.

В Microsoft SQL Server синтаксис оператора для создания триггера выглядит следующим образом.

```
    CREATE TRIGGER имяТриггера
    ON имяТаблицыПредставления
    {{FOR|AFTER|INSTEAD OF} {[INSERT], [DELETE], [UP-
DATE]}
    AS
    операторыSQL
    [IF UPDATE (ПОЛЕ) [{AND|OR} UPDATE (ПОЛЕ)]] и тд…
```

Изменеие триггера
```
ALTER TRIGGER имяТриггера
```
On ……

Удаление триггера:
```
DROP TRIGGER имяТриггера
```

Триггер может быть создан только в текущей базе данных, но допускается обращение внутри триггера к другим базам данных, в том числе и расположенным на удаленном сервере.

Имя триггера должно быть уникальным в пределах базы данных. Дополнительно можно указать имя владельца.

При использовании триггеров, действия, выполняемые с таблицами, предварительно выполняются с временными таблицами. При вставке строк данные предварительно помещаются в таблицу `inserted`, при удалении – удаляемые данные помещаются в таблицу `deleted`, аналогично при модификации данных.

Работа триггера строится на предварительном контроле вводимых, удаляемых или модифицируемых данных с помощью временных таблиц. Данные таблицы имеют такую же структуру, как и таблицы, на которые создаются соответствующие триггеры.

Пример 1

Использование триггера для реализации ограничений на значение. В добавляемой в таблицу `Сделка` записи количество проданного товара должно быть не меньше, чем его остаток из таблицы `Склад`.

Команда вставки записи в таблицу `Сделка` может быть, например, такой:

```
INSERT INTO Сделка VALUES (3,1,-299,»01/08/2002»)
```

Создаваемый триггер должен отреагировать на ее выполнение следующим образом: необходимо отменить команду, если в таблице Склад величина остатка товара оказалась меньше продаваемого количества товара с введенным кодом (в примере код товара=3). Во вставляемой записи количество товара указывается со знаком «+», если товар поставляется, и со знаком «-», если он продается. Пред-

ставленный триггер настроен на обработку только одной добавляемой записи.

```
CREATE TRIGGER   Триггер_ins
ON Сделка FOR INSERT AS
IF  @@ROWCOUNT=1
BEGIN
IF NOT EXISTS(SELECT * FROM inserted
WHERE              -inserted.количество<=ALL(SELECT
Склад.Остаток
FROM Склад,Сделка
WHERE Склад.КодТовара= Сделка.КодТовара))
BEGIN
ROLLBACK TRAN
PRINT «Отмена поставки: товара на складе нет» END
END
```

Пример 2

Создать триггер для обработки операции удаления записи из таблицы Сделка, например, такой команды:

```
DELETE FROM Сделка WHERE КодСделки=4
```

Для товара, код которого указан при удалении записи, необходимо откорректировать его остаток на складе. Триггер обрабатывает только одну удаляемую запись.

```
CREATE TRIGGER Триггер_del
ON Сделка FOR DELETE AS
IF @@ROWCOUNT=1 -- удалена одна запись
BEGIN
DECLARE @y INT,@x INT
--определяется код и количество товара из удаленной
--из таблицы Склад записи
SELECT @y=КодТовара, @x=Количество
FROM deleted
--в таблице Склад корректируется количество товара
UPDATE  Склад
SET Остаток=Остаток-@x
WHERE КодТовара=@y
END
```

В среде SQL server триггер можно построить в приложении Management Studio: В контекстном меню таблицы, на которую создается триггер (конМеню→ВсеЗадачи→ManageTriggers)

Или выполнением запроса соответствующего содержания из любого доступного приложения, например в Query Analyzer.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Определить событие, на которое должен реагировать триггер.
2. Определить тип триггера, то есть он должен отреагировать на событие после его совершения или вместо его.
3. Выявить операции сравнения, выявляющие ситуацию в которой происходит триггер.
4. Выявить действия, которые он выполняет.
5. Создать триггеры на сервере.
6. Разработать контрольные примеры, обеспечивающие демонстрацию работы триггера.
7. Продемонстрировать работу триггера на контрольных примерах.
8. Защитить работу.

4. ЗАДАНИЕ

Заданием для работы является создание триггеров на таблицы базы данных по заданным требованиям.

1. Создать триггер, для самой модифицируемой таблицы. Триггер должен вести журнал изменений (журнал – отдельная таблице).
2. Создать триггеры, препятствующие вставке в отношении определенных кортежей, не отвечающих заданным требованиям, предъявляемым к значению каких либо атрибутов.
3. Создать триггер препятствующий изменению численного атрибута отношения выходящего за определенные установленные рамки.
4. Создать триггер, выполняющий каскадное удаление кортежей с внешними ключей из таблицы, с которыми связаны удален-

ные кортежи с первичными ключами в исходной таблице, если в ней таких кортежей больше не осталось.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое триггер? На какие события создаются триггеры СУБД MS SQL Server?
2. Какие действия может выполнять триггер?
3. Общий формат команды создания триггера.
4. На каком принципе построена логика работы триггеров?

РАБОТА №7 ПРАКТИЧЕСКАЯ. УСТАНОВКА И НАСТРОЙКА СЕРВЕРА MS SQL SERVER EXPRESS

1. ЦЕЛЬ РАБОТЫ

Целью работы является получение практических навыков установки и настройки сервера баз данных.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

MS SQL Server Express является свободно распространяемой версией сервера MS SQL Server. Отличается некоторыми ограничениями в создании и выполнении программных модулей.

Для установки MS SQL Server Express необходимо скачать и установить соответствующие программные компоненты.

Ссылка на скачивание https://www.microsoft.com/ru-ru/download/details.aspx?id=29062

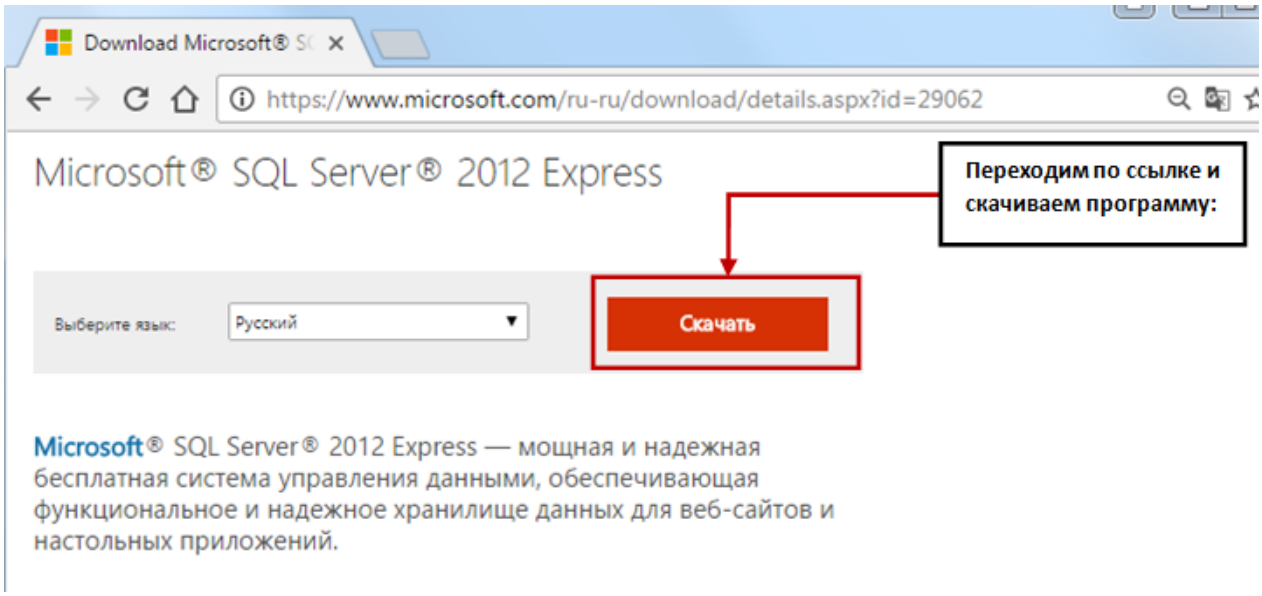


Рис.7.1 Источник для скачивания версии

При выборе файла загрузки необходимо учесть тип своей операционной системы

Для Для 32-разрядной системы:
RUS\x86\SQLEXPR32_x86_RUS.exe
Для 64- разрядной системы:
RUS\x64\SQLEXPR_x64_RUS.exe

Установка выполняется в виде следующих шагов.

ШАГ 1: Запуск установки

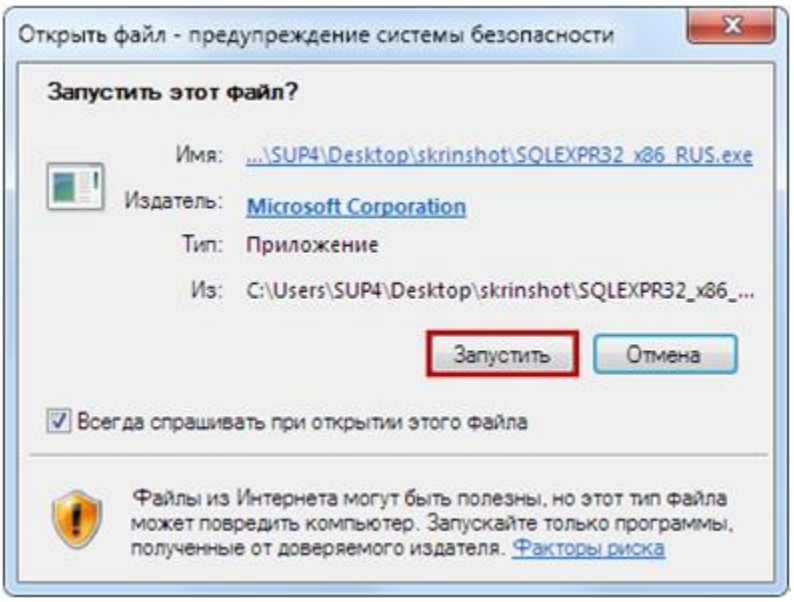


Рис.7.2 Запуск установки MS SQL Sever

ШАГ 2: Распаковка компонентов

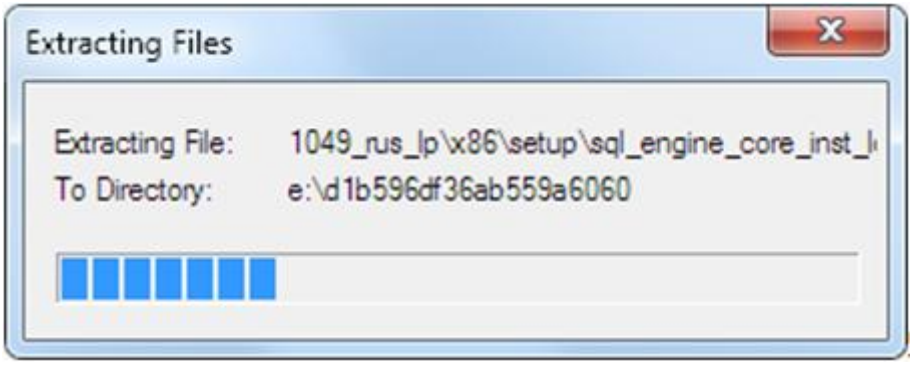


Рис.7.3 Распаковка компонентов

ШАГ 3: Выбор типа установка (обновление или установка)

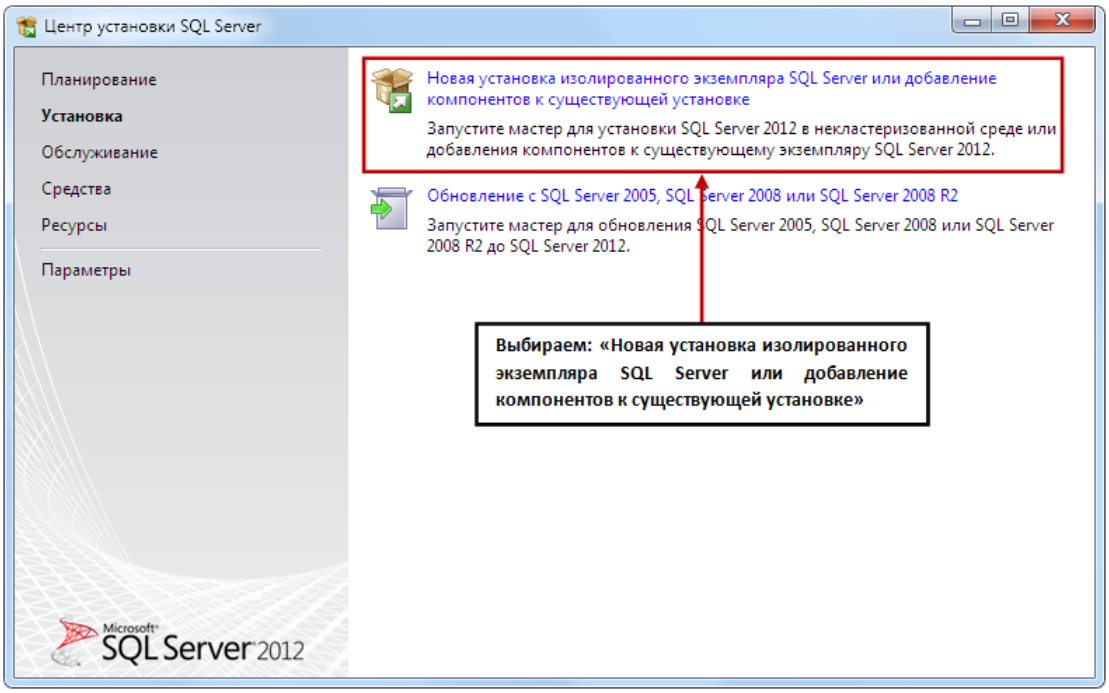


Рис.7.4 Выбор типа установки

ШАГ 4: Лицензионное соглашение

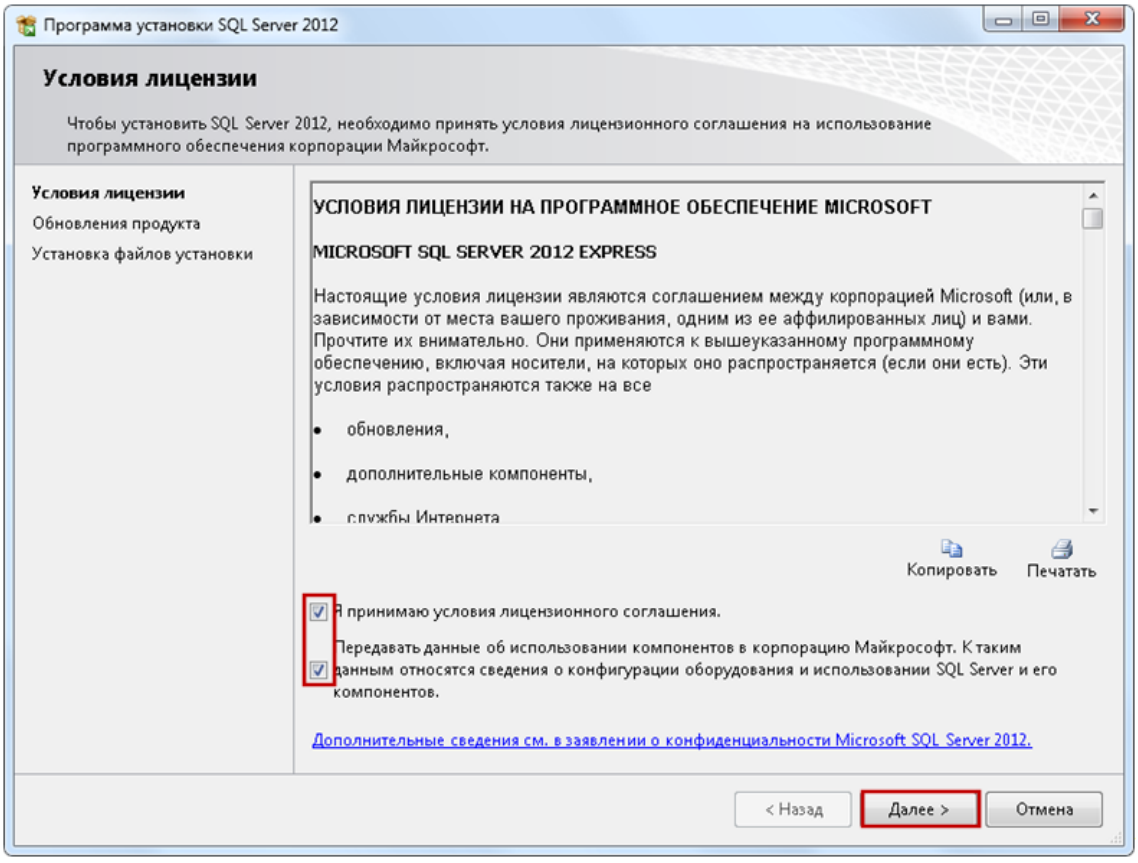


Рис.7.5 Подтверждение лицензионного соглашения

ШАГ 5: Проверка и загрузка обновления продукта

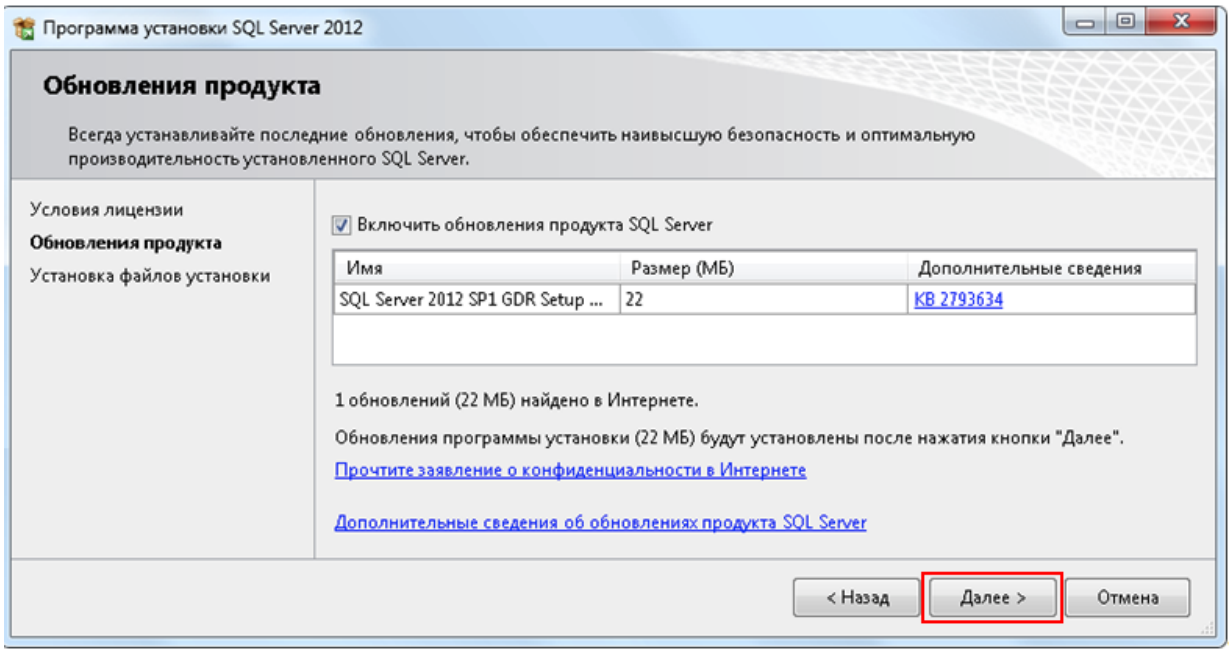


Рис.7.6 Проверка обновлений продукта

ШАГ 6: Непосредственно установка

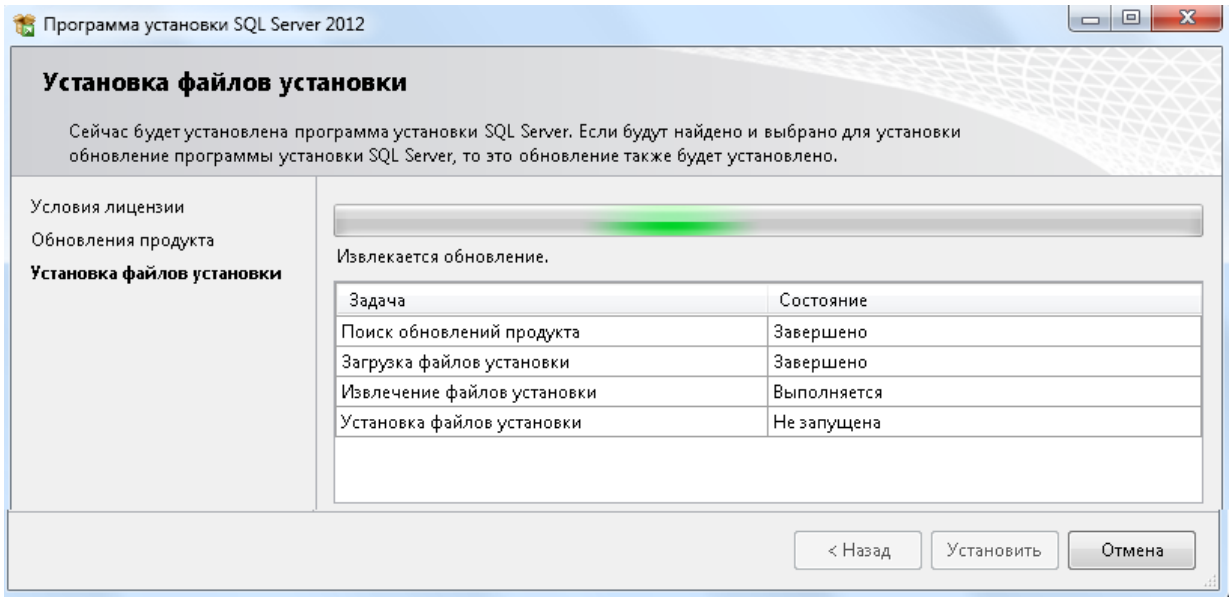


Рис.7.7 Процесс непосредственной установки

ШАГ 7: Выбор компонентов установки

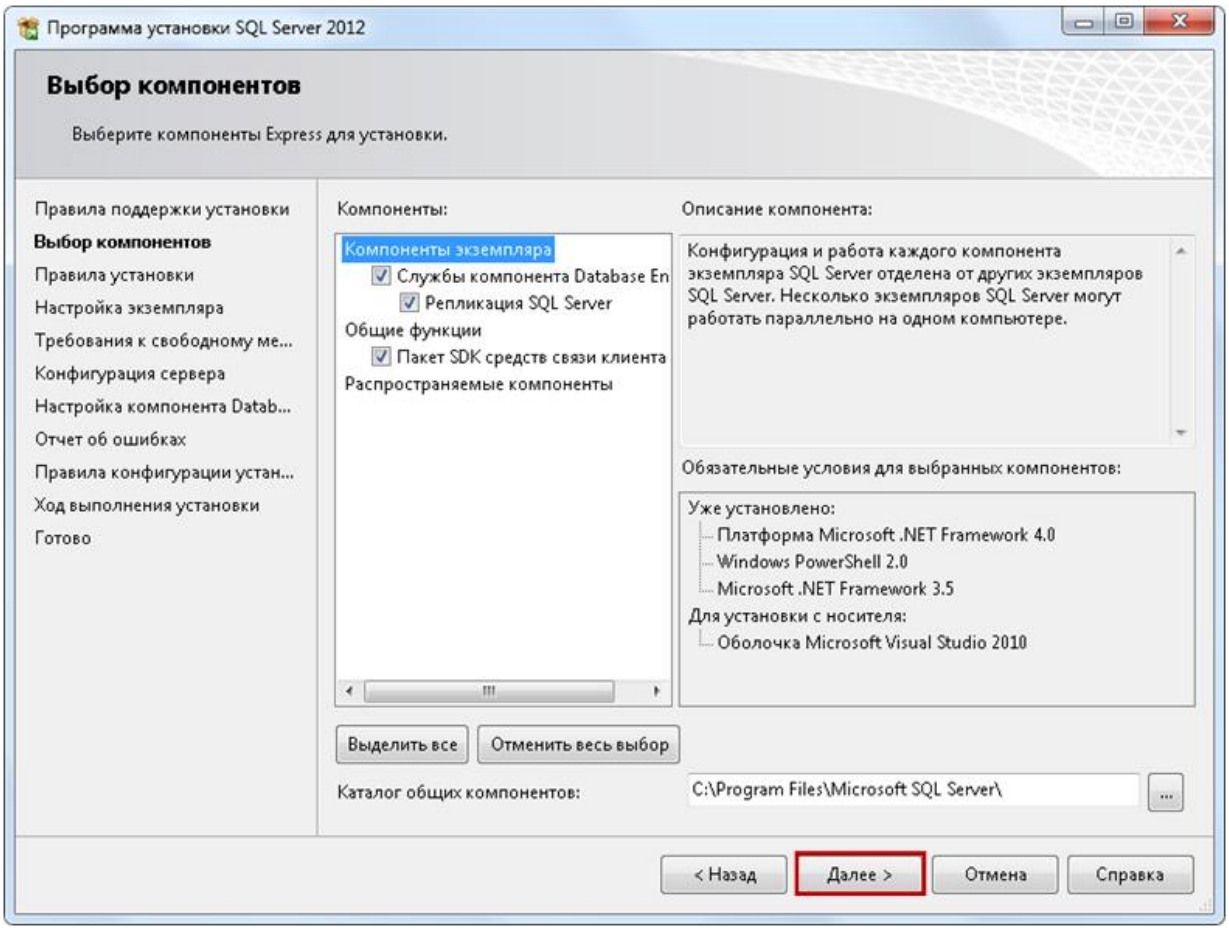


Рис.7.8 Выбор компонентов установки

ШАГ 8: Задание имени устанавливаемому серверу

При выполнении данного шага целесообразно изменить значение граф «Именованный экземпляр» и «Идентификатор экземпляра» на выбранное имя экземпляра.

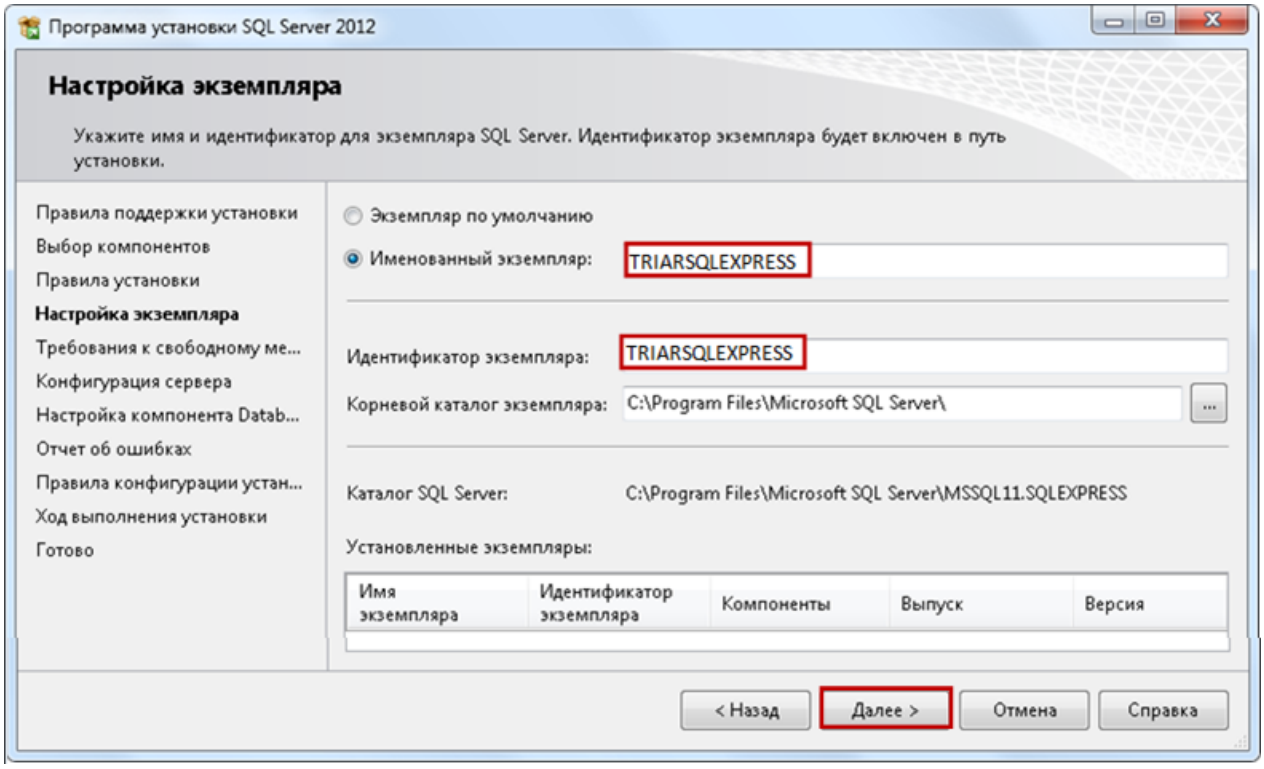


Рис.7.9 Задание имени устанавливаемому экземпляру сервера

ШАГ 9: Конфигурирование сервера

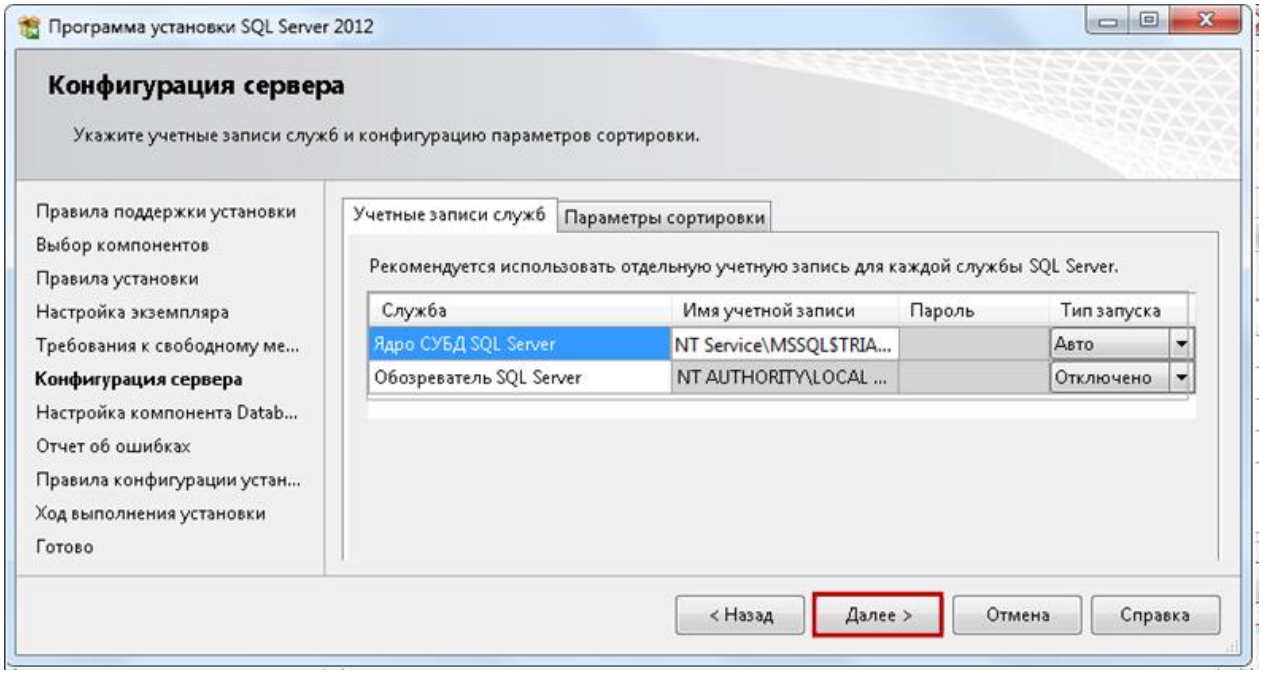


Рис.7.10 Конфигурирование сервера

ШАГ 10: Настройка компонента движка данных

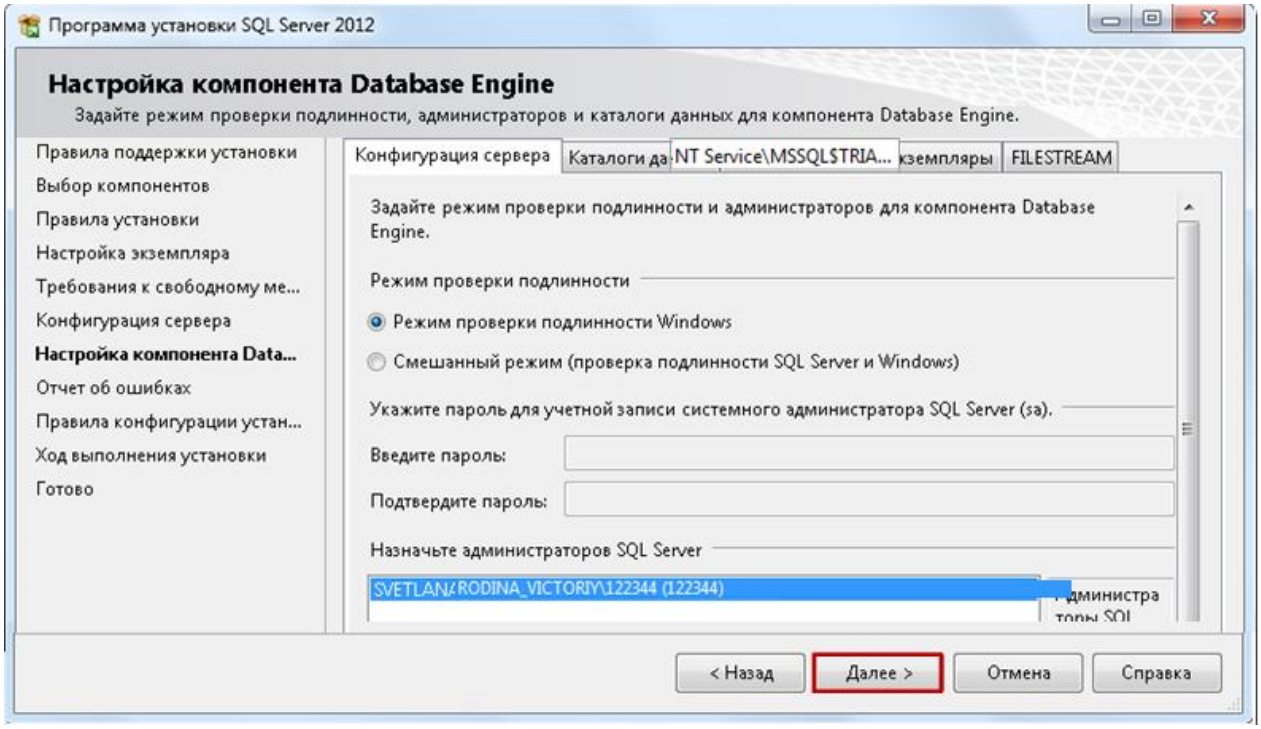


Рис.7.11 Настройка службы Database Engine

ШАГ 11: Просмотр отчёта об ошибках

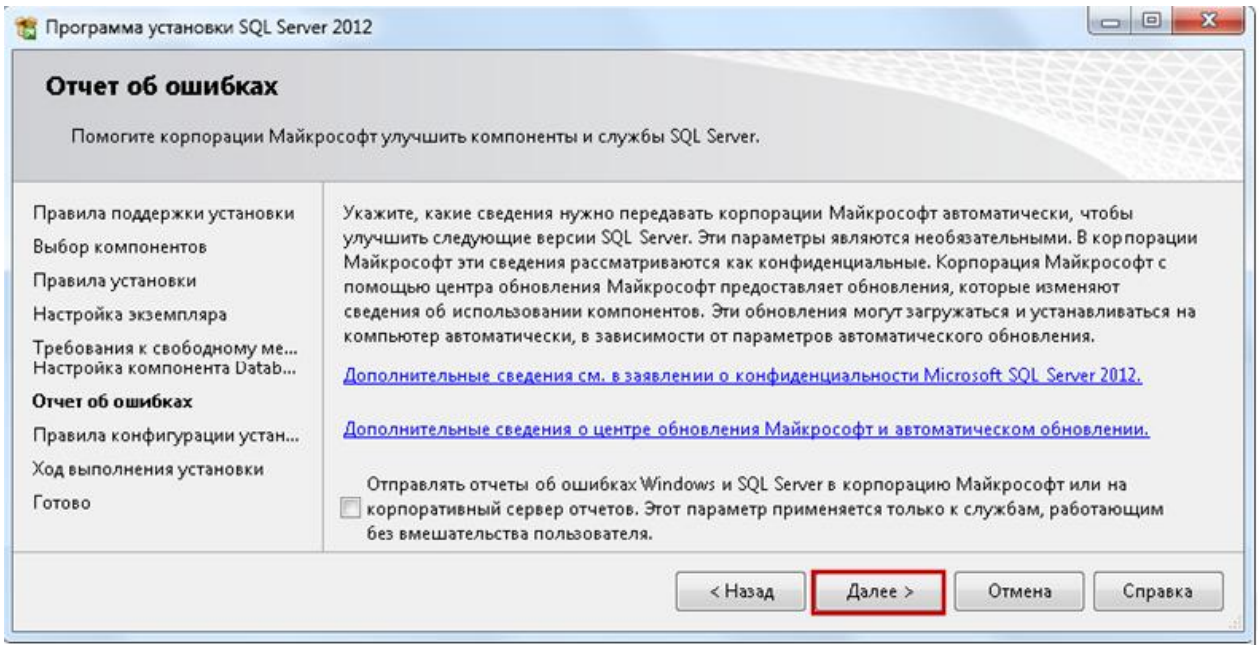


Рис.7.12 Отчёт об ошибках

ШАГ 12: Непосредственно установка в соответствии с настройками. Создание компонентов

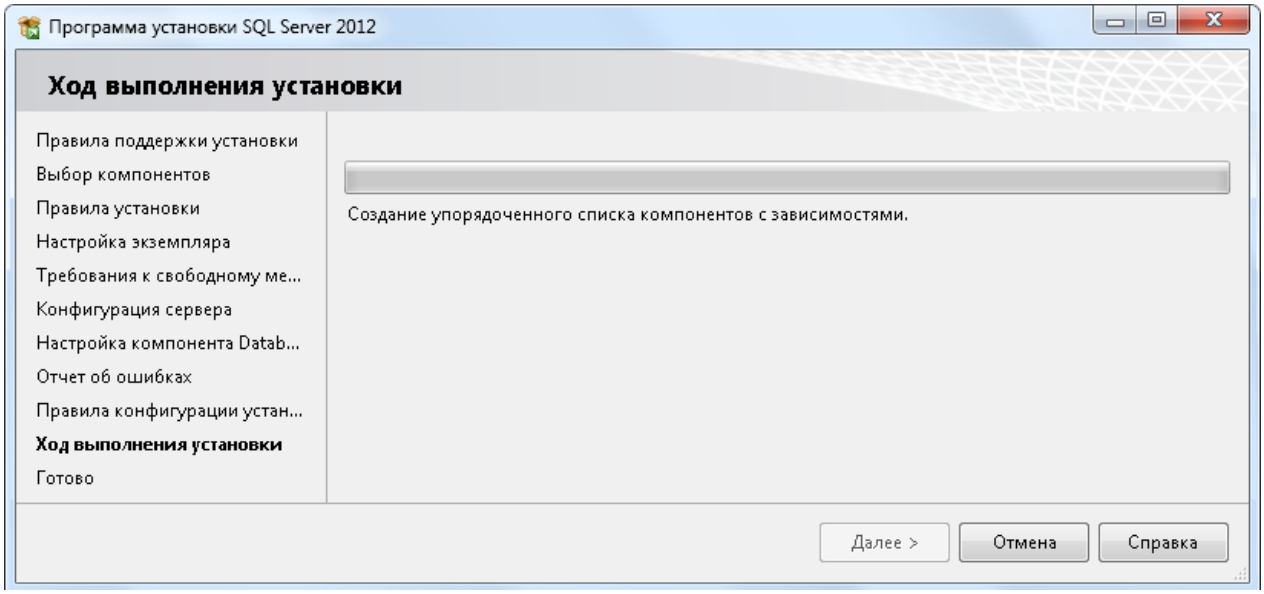


Рис.7.13 Установка. Создание устанавливаемых компонентов

ШАГ 13: Установка компонентов

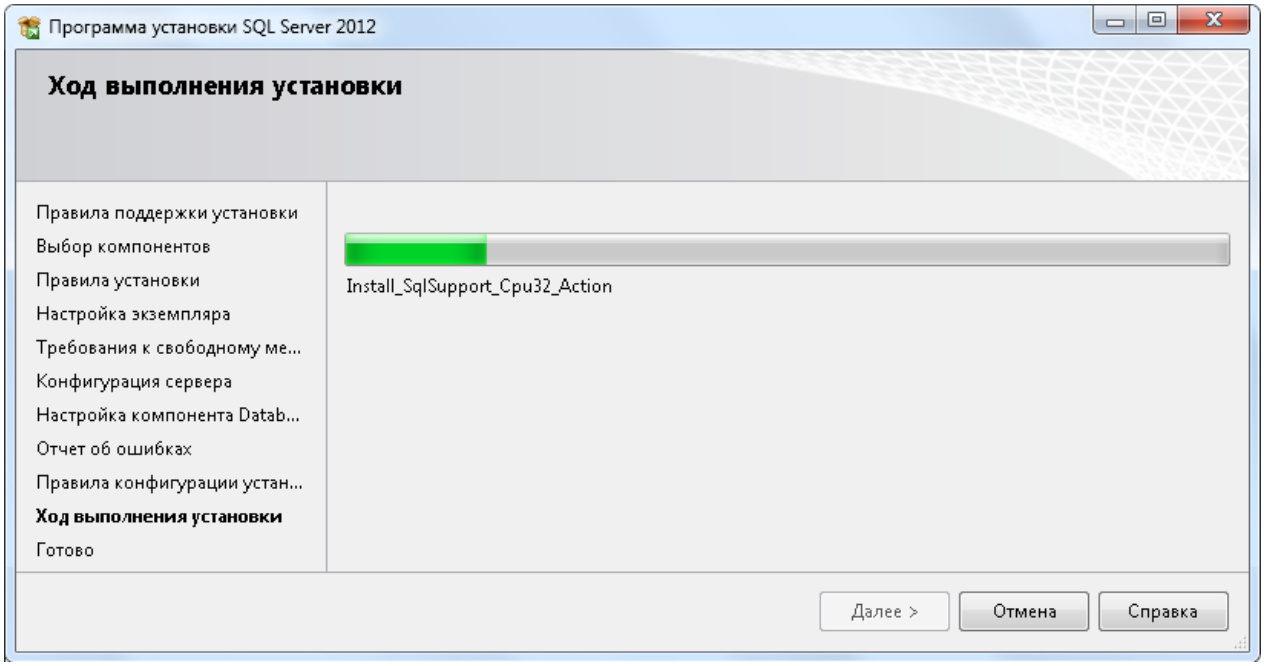


Рис.7.14 Установка компонентов

ШАГ 14: Завершение установки

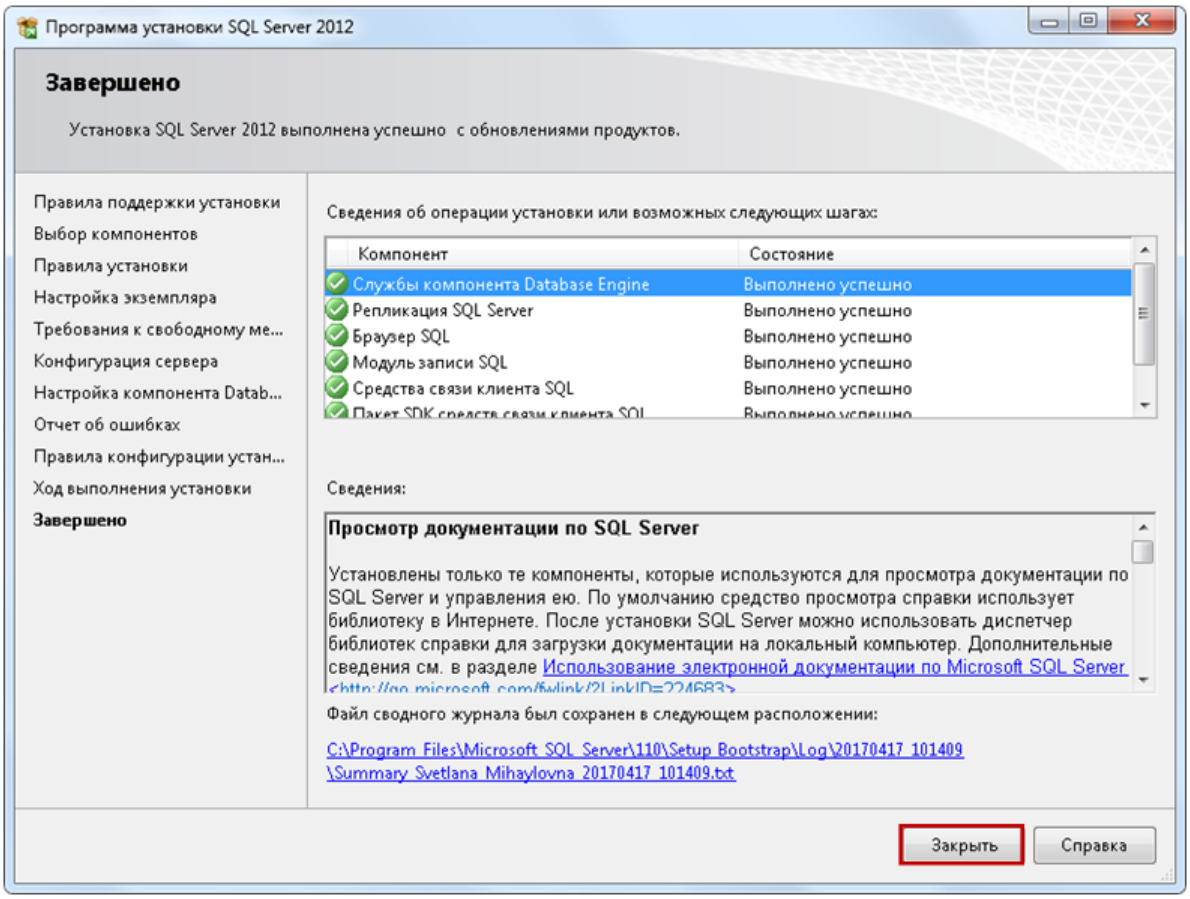


Рис.7.15 Завершение установки

3. ПОРЯДОК ВЫПОЛНЕНИЯ И ЗАДАНИЯ ДЛЯ РАБОТЫ

1. Подготовить вычислительную машину для установки. Освободить при необходимости дисковое пространство.
2. Скачать необходимый файл установки.
3. Выполнить установку MS SQL Server скачанной версии.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие версии MS SQL Server могут быть установлены?
2. Что нужно указать, для того что бы с помощью курсора можно было менять значение атрибутов отношения?
3. Какие функции расширения возможностей обработки данных поддерживают курсоры?
4. Какая последовательность действий при работе с курсором?
5. Описать синтаксис операторов используемых при работе с курсором.
6. Как контролировать работу курсора серверными переменными?

РАБОТА №8 ПРАКТИЧЕСКАЯ. УСТАНОВКА И НАСТРОЙКА СУБД MY SQL

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является получение практических навыков установки и настройки СУБД MeSQL.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

MySQL – свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

MySQL портирована на большое количество платформ: FreeBSDLinux, macOS, NetBSD, OpenBSD, OS/2 Warp, Solaris, SunOS, Windows 95 – Windows 7 и Windows 10.

На официальном сайте СУБД для свободной загрузки предоставляются не только исходные коды, но и откомпилированные и оптимизированные под конкретные операционные системы готовые исполняемые модули СУБД MySQL.

Источник для установки
https://dev.mysql.com/downloads/installer/.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Загрузка установочных компонентов

1.1. Перейдите по адресу http://dev.mysql.com/downloads/installer/.

1.2. Нажмите кнопку «Загрузить».

1.3. Сохраните файл программы установки на вашем компьютере.

2. Начало установки

2.1. После завершения загрузки запустите программу установки следующим образом.

2.2. Щелкните правой кнопкой мыши загруженный установочный файл (например, mysql-installer-community-5.6.14.0.msi) и выберите пункт «Выполнить». Запустится программа установки MySQL.

2.3. На панели приветствия выберите ***«Установить продукты MySQL».***

2.4. На панели информации о лицензии ознакомьтесь с лицензионным соглашением, установите флажок принятия и нажмите кнопку ***«Далее».***

2.5. На панели «Найти последние продукты» нажмите кнопку ***«Выполнить».***

2.6. После завершения операции нажмите кнопку ***«Далее».***

3. Настройка установки сервера

3.1. На панели ***«Тип настройки»*** выберите параметр ***«Пользовательская»***, а затем нажмите кнопку ***«Далее».***

3.2. На панели «Выбор компонентов обеспечения» убедитесь, что выбран MySQL Server 5.6.x, и нажмите кнопку «Далее».

3.3. На панели ***«Проверить требования»*** нажмите кнопку ***«Далее».***

4. Окончание установки.

4.1. На панели ***«Установка»*** нажмите кнопку ***«Выполнить».***

4.2. После успешного завершения установки сервера на панели ***«Установка»*** отображается информационное сообщение. Нажмите кнопку ***«Далее».***

5. Настройка установленного сервера.

5.1. На странице ***«Настройка»*** нажмите кнопку ***«Далее».***

5.2. На первой странице конфигурации сервера MySQL (1/3) установите следующие параметры:

5.2.1. Тип конфигурации сервера.

5.2.2. Выберите вариант ***«Компьютер для разработки».***

5.2.3. Включите поддержку сети TCP/IP. Убедитесь, что флажок установлен.

5.2.4. Задайте следующие параметры ниже:

- Номер порта. Укажите порт подключения. По умолчанию установлено значение 3306; не следует изменять его без необходимости.
- Откройте порт брандмауэра для доступа к сети. Выберите исключение добавления брандмауэра для указанного порта.

5.3. Расширенная настройка.

5.3.1. Выберите флажок ***«Показать расширенные параметры»*** для отображения дополнительной страницы конфигурации для настройки расширенных параметров для экземпляра сервера (если требуется). *Примечание. При выборе этого параметра необходимо перейти к панели для установки параметров сети, где будет отключен брандмауэр для порта, используемого сервером MySQL.*

5.3.2. Нажмите кнопку «Далее».

- На второй странице конфигурации сервера MySQL (2/3) установите следующие параметры:
  - Пароль учетной записи root. *Пользователь root – это пользователь, который имеет полный доступ к серверу баз данных MySQL – создание, обновление и удаление пользователей и так далее. Запомните пароль пользователя root (администратора) – он понадобится вам при создании примера базы данных.*
  - Пароль root для MySQL. Введите пароль пользователя root.
  - Повторите ввод пароля. Повторно введите пароль пользователя root.

5.3.3. Учетные записи пользователя MySQL. Нажмите кнопку ***«Добавить пользователя»*** для создания учетной записи пользователя. В диалоговом окне «Сведения о пользователе MySQL» введите имя пользователя, роль базы данных и пароль (например, !phpuser). Нажмите кнопку «ОК».

5.3.4. Нажмите кнопку «Далее».

- На третьей странице конфигурации сервера MySQL (3/3) установите следующие параметры:
  - «Имя службы Windows». – Укажите имя службы Windows, которая будет использоваться для экземпляра сервера MySQL.

- «Запуск сервера MySQL при запуске системы». – Не снимайте этот флажок, если сервер MySQL требуется для автоматического запуска при запуске системы.
- «Запуск службы Windows в качестве». Возможны следующие варианты.
  - «Стандартная системная учетная запись» – Рекомендуется для большинства сценариев.
  - «Нестандартный пользователь» – Существующая учетная запись пользователя рекомендуется для сложных сценариев.

5.3.5. Нажмите кнопку «Далее».

5.3.6. На странице «Обзор конфигурации» нажмите кнопку «Далее».

5.4. После успешного завершения настройки на панели «Завершение» появляется информационное сообщение. Нажмите кнопку «Завершить».

6. Примечание. Для проверки успешности настройки запустите диспетчер задач. Если MySQLd-nt.exe присутствует в списке «Процессы», сервер базы данных запущен.

4. ЗАДАНИЕ

Заданием для практической работы является установка и настройка экземпляра сервера на домашнем компьютере. И подготовка отчёта о настройке.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какова область применения сервера MY SQL?
2. Где можно взять компоненты для установки сервера MY SQL?
3. Какие этапы установки можно выделить?
4. Какие параметры установки необходимо задать?
5. На какой порт необходимо настраивать установку сервера MY SQL?
6. Что такое пользователь «Root»?

РАБОТА №9 ПРАКТИЧЕСКАЯ. КОПИРОВАНИЕ БАЗ ДАННЫХ, ИМПОРТ ЭКСПОРТ ДАННЫХ В СРЕДЕ MS SQL SERVER EXPRESS СРЕДСТВАМИ MANAGEMENT STUDIO

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Под копированием подразумевается перенос базы данных из одного сервера на другой.

Данные баз данных в среде MS SQL SERVER располагаются на отдельных файлах одноимённых базе данных. Но они не могут быть скопированы как обычные файлы в файловой системе, так как их защищает от копирования служба сервера.

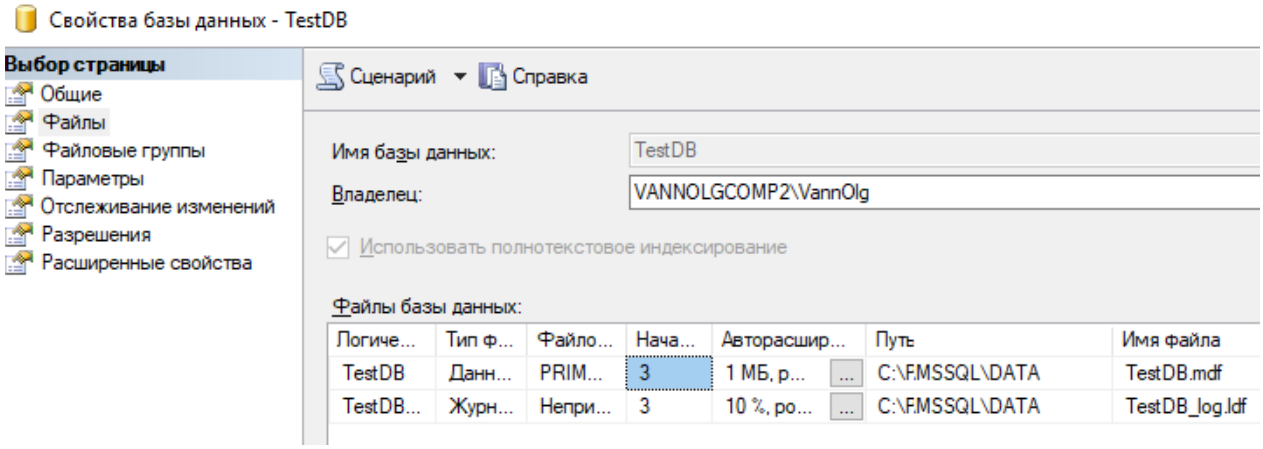


Рис.9.1 Файлы базы данных TestDB

Для копирования базы данных необходимо использовать специальные процедуры, предоставляемые MS Management Studio – «Создать резервную копию», «Восстановить».

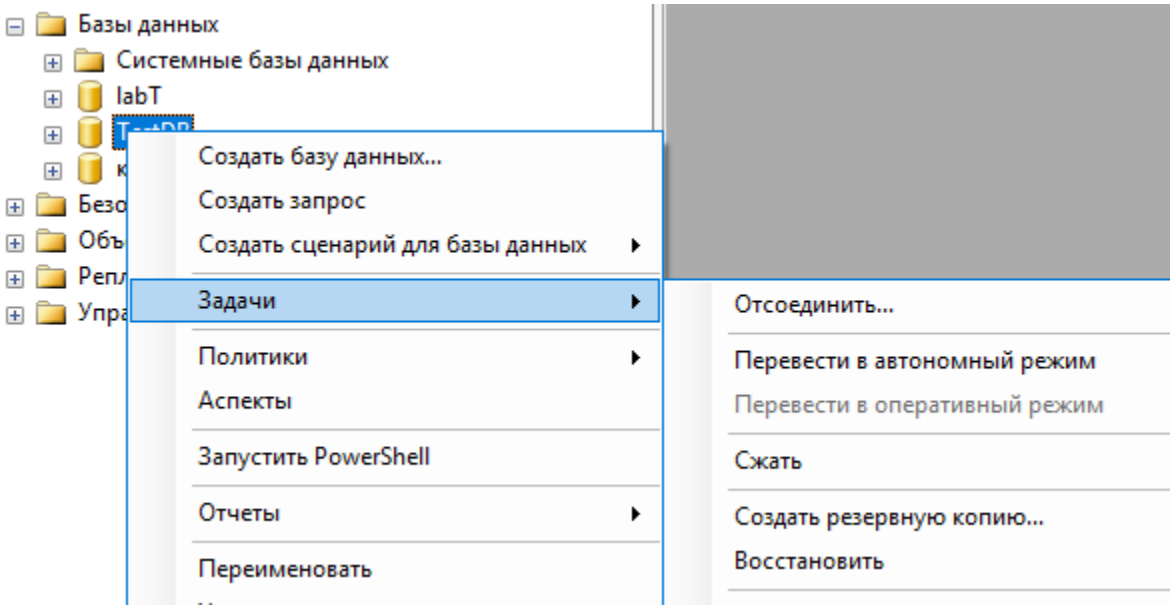


Рис.9.2 Задачи работы с базой данных, вызываемых из контекстного меню базы данных, в том числе «Создать резервную копию» и «Восстановить»

При создании резервной копии необходимо задать общие параметры создания копии (страница «Общие» окна копирования) и дополнительные параметры создания копии (Страница «Параметры» окна копирования).

Общие параметры создания копии (рис. 9.3):

- Тип копии:
  - возможно создание полной копии;
  - разностной (разностная копия, содержит данные об отличии от предшествующей копии. Для баз малого и среднего размера целесообразно выбирать полное копирование).
- Компонент резервного копирования (база данных в целом или отдельные файлы).
- Параметры «Назначение» они определяют место в которое будет производиться копирование.

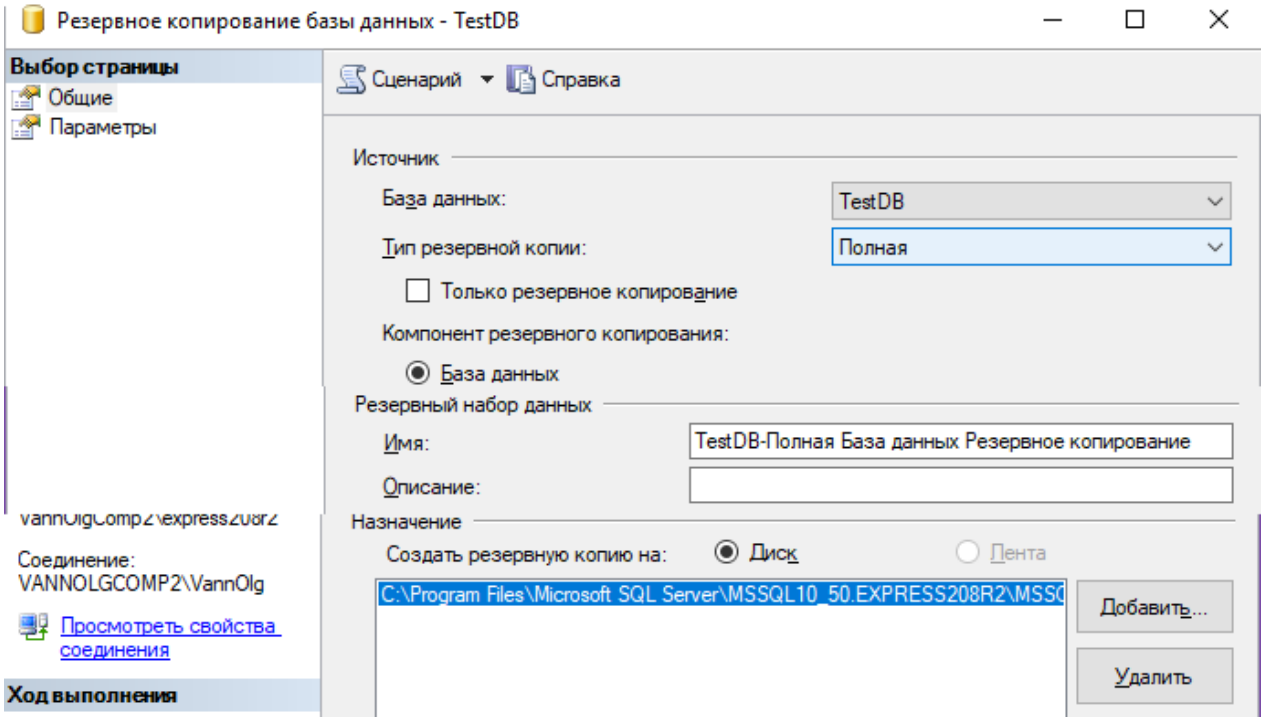


Рис.9.3 Диалоговое окно создания резервной копии базы данных

В параметрах назначения необходимо выбрать файл, в котором будет создаваться резервная копия. По умолчанию этот файл одноимённый с базой данных и имеет расширение «***.bak***» и располагается в каталоге, предназначенном для файлов такого типа при установке сервера.

Кнопкой «добавить» можно добавить файл резервной копии. Тогда база будет создаваться на двух файла, что не удобно. *Поэтому если есть необходимость создать файл копии отличный от существующего или от установленного по умолчанию нужно добавить требуемый файл, а указание на другие файлы копирования удалить.*

При создании нового файла резервной копии можно указать любой путь для расположения резервной копии и любое имя файла.

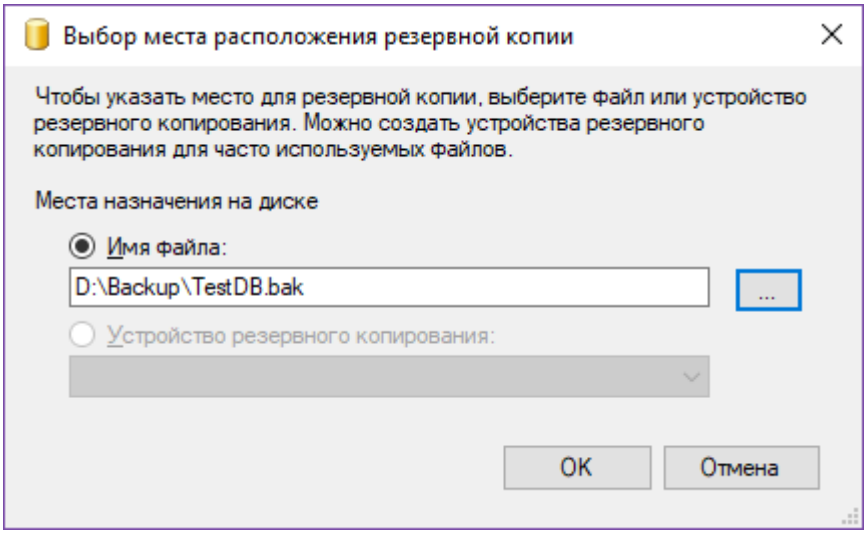


Рис.9.4 Диалоговое окно добавления нового файла резервной копии

После завершения копирования откроется окно сообщений с подтверждением выполнения данного действия.

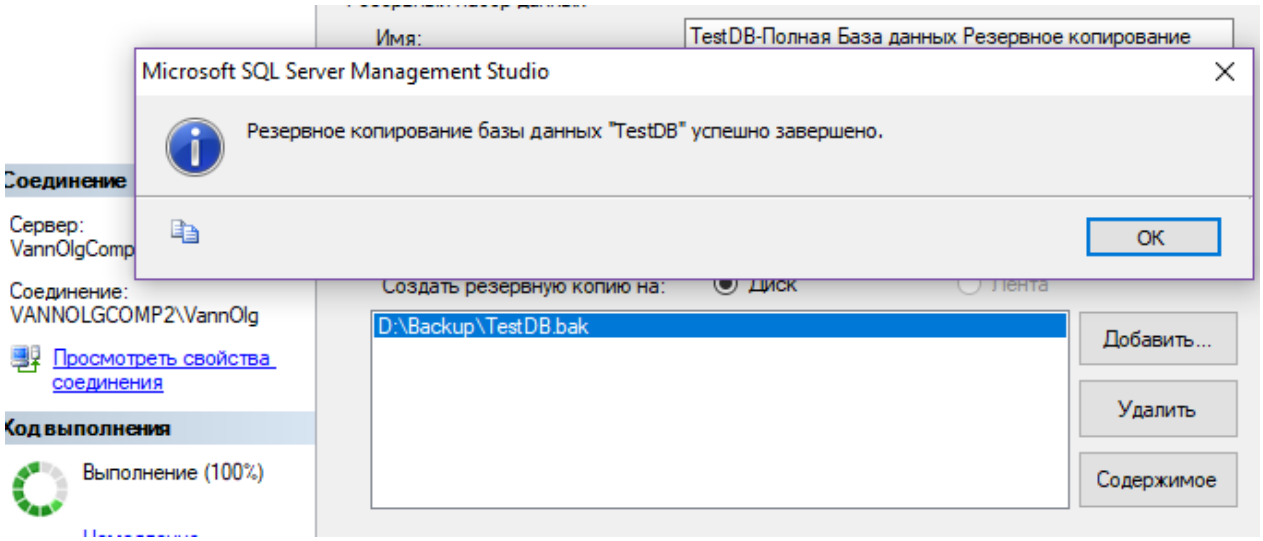


Рис.9.5 Успешное завершение создания резервной копии

Восстановление базы данных

Восстановление БД может быть вызвано как из контекстного меню узла Базы данных, так и из узла конкретной базы данных.

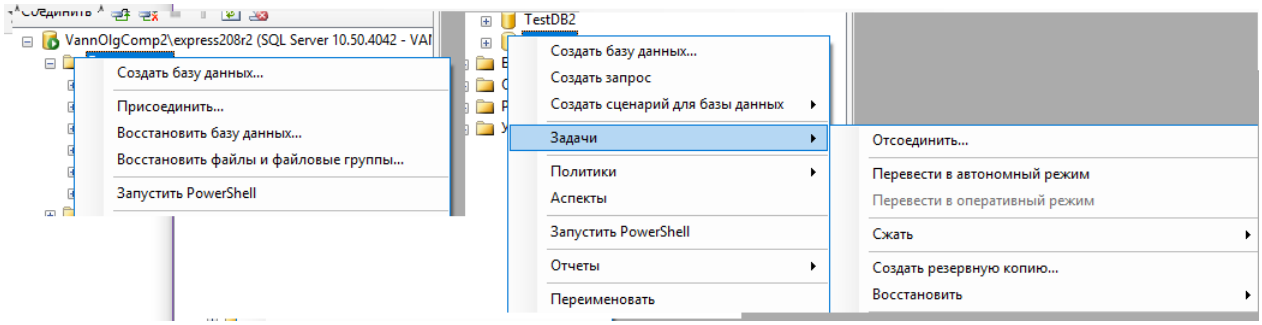


Рис.9.6 Восстановление базы данных из файла .bak
в новую базу и в существующую

В первом случае в окне восстановления необходимо задать имя базы данных, в которую будет происходить восстановление, во втором случае восстановление будет происходить в существующую базу. При этом данные базы данных будут уничтожены.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Выполнить сохранение базы данных в резервную копию.
2. Восстановить базу данных в новую базу данных.
3. Восстановить базу данных в существующую базу данных.
4. Создать отчёт подтверждающий выполнение отдельных этапов соответствующими копиями экрана.

4. ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ

В качестве исходной базы данных для создания резервной копии необходимо взять базу данных с результатами ранее выполненных лабораторных работ.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В виде чего хранятся данных базы данных в среде MS SQL Server?
2. Какие способы могут быть использованы для копирование базы данных?
3. Каким какие параметры должны быть заданы при создании резервной копии?
4. Из каких узлов обозревателя объектов сервера можно выполнить восстановление базы данных? Какие особенности восстановления при этом получаются?

РАБОТА №10 ПАРКТИЧЕСКАЯ. КОПИРОВАНИЕ БАЗ ДАННЫХ СРЕДСТВАМИ КОМАНД SQL

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью выполнения работы является получение практических навыков по копированию баз данных средствами команд SQL.

Задачами работы является изучение команд создание объектов данных DDL, команд модификации данных, обеспечивающих создание копии базы данных.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Создание копии с использованием средств администрирования рассмотренное в прошлой работе позволяет создавать копии для базы данных или для одного и того же сервера или для совместимых версий серверов MS SQL SERVER. Но часто возникает задача переноса данных на другую, более позднюю версию сервера, что не возможно средствами, создания и восстановления резервной копии. Для копирования данных на более позднюю версию сервера или даже на другой сервер всегда могут быть использованы команда подъязыков SQL DDL и DML. В данном случае копирование будет заключатся в создании таблиц и других объектов данных идентичных исходной базе и заполнения данных в таблицы через команды вставки данных.

Для упрощения работы по генерации команд создания отношений можно воспользоваться предоставляемые средой SSMS возможностью автоматической генерации данных команд через контекстное меню.

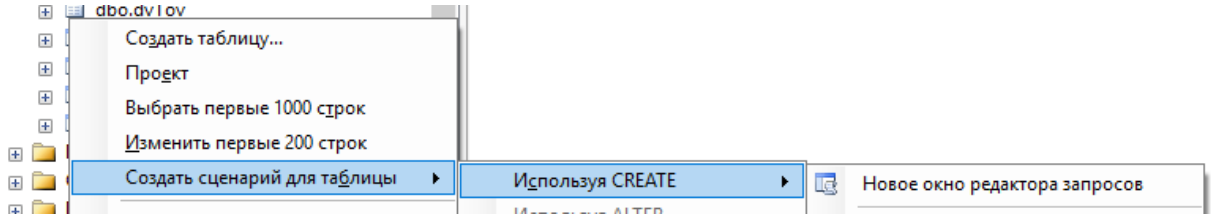


Рис.10.1 Вызов автоматической генерации команд создания таблиц

Код, который будет сгенерирован при этом, приведён ниже.

```
USE [TestDB]
GO

CREATE TABLE [dbo].[OstTov](
    [kodTOv] [int] NOT NULL,
    [NameTov] [nvarchar](12) NULL,
    [kolTov] [int] NULL,
    [ris] [image] NULL,
CONSTRAINT [PK_OstTov] PRIMARY KEY CLUSTERED
(
    [kodTOv] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY =  OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
```

Рис.10.2 Автоматически сгенерированный код
для создания существующей таблицы

Существует возможность автоматической генерации сценария для создания всех объектов базы данных.

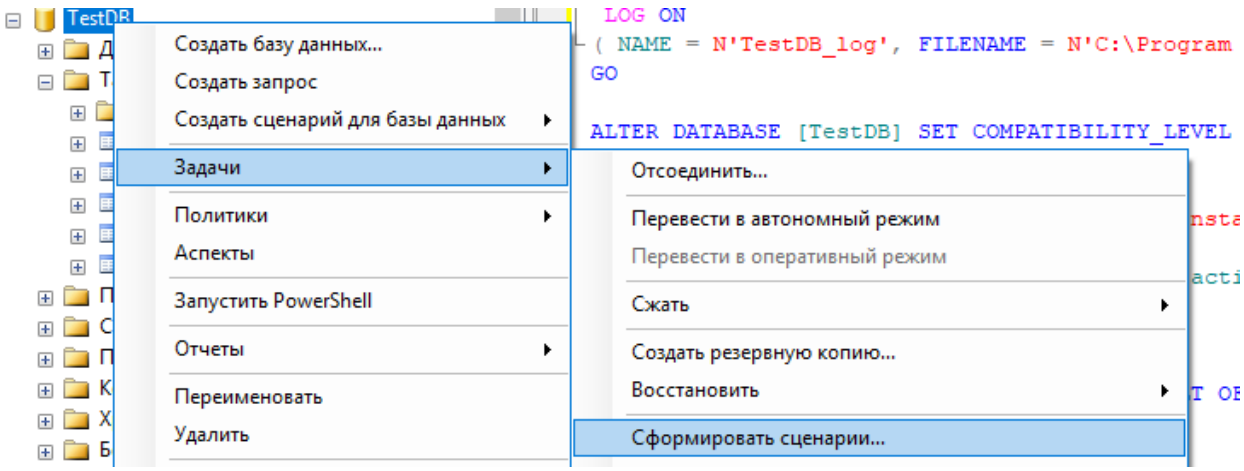


Рис.10.3 Вызов автоматической генерации сценариев
создания объектов базы данных

Однако полученный сценарий, в случае копирования на другой сервер необходимо скорректировать.

Заполнение данных производится через команды insert. Генерация шаблонов этих команд также может быть выполнена автоматически.

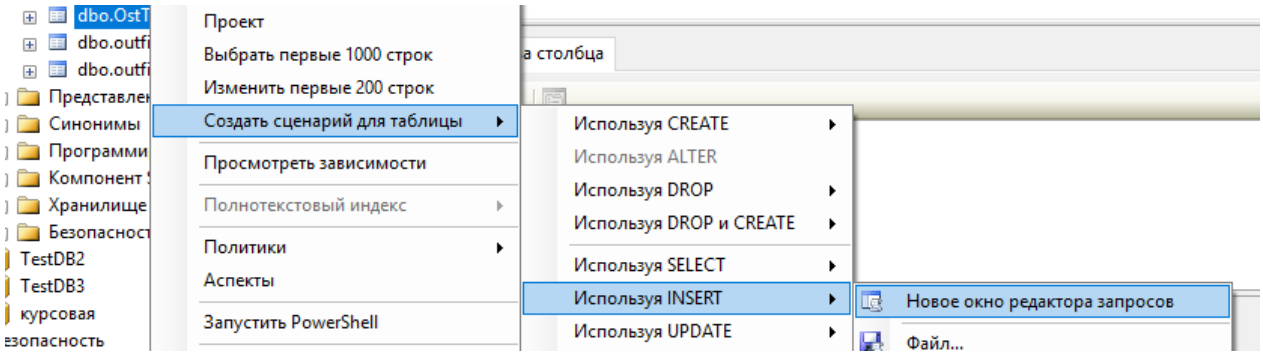


Рис.10.4 Вызов автоматической генерации команд вставки данных в существующую таблицу

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создать новую базу данных в текущем подключении к серверу.
2. Создать команды для создания объектов данных в новой базе идентичных существующим в исходной базе данных. Данные команды оформить в виде отдельно файла «.sql».
3. Создать команды для заполнения таблиц данными. Оформить данные команды в виде второго файла «.sql».
4. Выполнить скрипты для создания и заполнения объектов в новой базе данных.
5. Выполнить запросы, подтверждающие идентичную работу исходной и целевой баз данных.
6. Создать отчёт по работе, включающий отображение в виде копий экрана этапы выполнения работы.

4. ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ

Заданием для работы является создание копии базы данных, в которой выполнялись работы по созданию хранимых процедур и триггеров.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие объекты в базе данных могут присутствовать?
2. Назовите команды создания таблиц в базе данных, хранимых процедур, триггеров.
3. Какие команды используются для заполнения таблиц данными?
4. Каким образом можно автоматически сгенерировать команды для создания и заполнения базы данных?
5. Какие методы создания копии баз данных вы знаете?

РАБОТА №11 ПРАКТИЧЕСКАЯ. ПЕРЕНОС БАЗЫ ДАННЫХ НА ДРУГОЙ ТИП СЕРВЕРА

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цель данной работы – получить практические навыки переноса данных между серверами баз данных различного типа. В данном случае будет рассматриваться перенос данных между базами данных в среде сервера MS SQL Server 2008R2 и сервером MySQL.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Перенос данных между различными типами серверов не может быть сделан административными средствами. Для этого не существуют некоторые специальные инструменты. Но в любом случае администратор должен владеть навыками использования для выполнения этих задач обычных команд SQL. Подход в данном случае аналогичен тому что рассматривался в работе по переносу данных между базами данных в среде одной СУБД.

Сложности могут возникнуть при внесении данных. Так как данных может большое количество. Для автоматизации этого процесса можно выполнить выгрузку данных из исходной базы данных в некоторый файл из которого возможна загрузка данных в целевой базе данных. Обычно подходит файл формата «.xml» или «.txt». В данной работе будет использоваться файл формата «.txt».

Для выгрузки данных в текстовый файл можно воспользоваться средствами мастера импорта и экспорта данных MS SQL Server Management Studio (SSMS), запускаемого из контекстного меню любой базы данных, командой «задачи/экспорт данных».

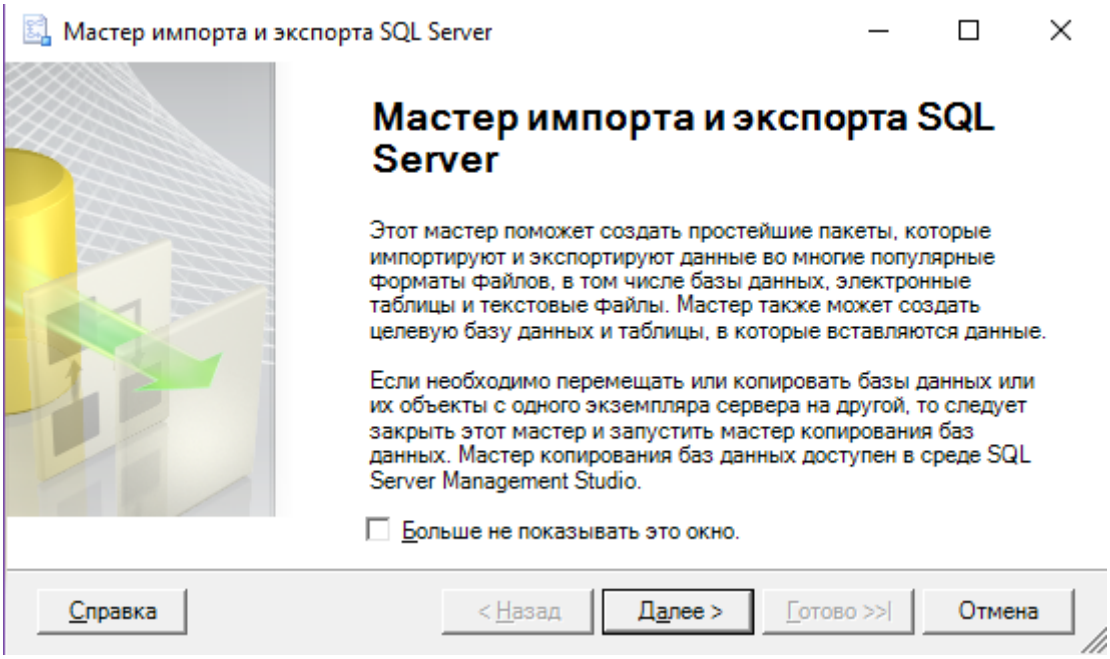


Рис.11.1 Окно импорта экспорта данных

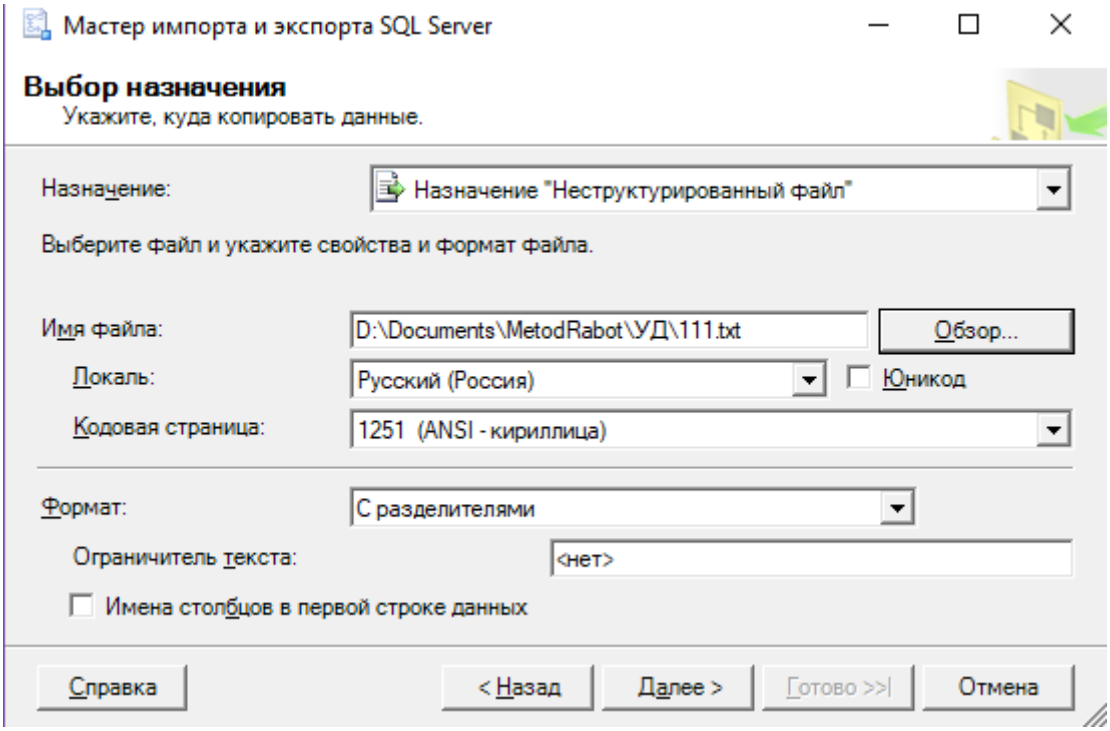


Рис.11.2 Выбор места назначения при экспорте данных в текстовый файл

При экспорте в текстовый файл необходимо выбрать в качестве параметра назначения – «неструктурированный файл».

Для выполнения загрузки данных из файла в таблицу в среде MySQL может быть использована специальная команда из диалекта SQL MYSQl ***«LOAD DATA INFILE «XXX.txt» INTO TABLE XXXXXX;***

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создать скрипты для создания объектов данных, существующих в исходной базе данных в среде MS SQL Server.
2. Создать базу данных в сред MySQL, в которую предполагается перенос данных.
3. Скорректировать исходные скрипты в соответствие с особенностями диалекта MySQL.
4. Создать необходимые данные в целевой базе данных с помощью скорректированного скриптового файла.
5. Экспортировать данные из таблиц исходной базы данных в текстовые файлы.
6. Создать скрипт для загрузки данных из файлов в таблицы целевой базы данных.

4. ЗАДАНИЕ ДЛЯ РАБОТЫ

Заданием для работы является экспорт базы данных, в которой выполнялись работы по созданию хранимых процедур и триггеров в базу данных под управлением сервера MySQL.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие объекты в базе данных могут присутствовать?
2. Назовите команды создания таблиц в базе данных, хранимых процедур, триггеров.
3. Какие команды используются для заполнения таблиц данными?
4. Каким образом можно автоматически сгенерировать команды для создания и заполнения базы данных?
5. Какие методы создания копии баз данных вы знаете?
6. Каким образом можно выгрузить данные из таблицы в базы данных в среде SQL Server в текстовый файл?
7. Какие команды для загрузки данных из текстовых файлов можно использовать в среде MySQL?

РАБОТА №12 ПРАКТИЧЕСКАЯ. СОЗДАНИЕ МЕХАНИЗМОВ СЕРВЕРА ДЛЯ ОБСЛУЖИВАНИЯ БАЗЫ ДАННЫХ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является получение практических навыков разработки механизмов сервера для обслуживания базы данных.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В MS SQL Server есть несколько системных баз данных:

master – В этой базе данных хранятся все данные системного уровня для экземпляра SQL Server;

model – Используется в качестве шаблона для всех баз данных, создаваемых в экземпляре SQL Server. Изменение размера, параметров сортировки, модели восстановления и других параметров базы данных model приводит к изменению соответствующих параметров всех баз данных, создаваемых после изменения;

msdb – Используется агентом SQL Server для планирования предупреждений и задач, так же является хранилищем пакетов SSIS, хранилищем информации по резервному копированию;

tempdb – База данных для временных объектов или для промежуточных результирующих наборов;

resource – База данных только для чтения. Содержит системные объекты, которые входят в состав SQL Server. Системные объекты физически хранятся в базе данных Resource, но логически отображаются в схеме sys любой базы данных.

Типичные задачи обслуживания для системных баз данных (за исключением БД TempDb и resource):

- создание резервной копии баз данных (с глубиной хранения минимум 7 дней);
- проверка целостности баз данных инструкцией DBCC CHECKDB.

Все эти операции можно оформить в виде в задание sql agent-а и выполнять ежедневно, благо объем данных баз данных обычно

небольшой, то и операции проходят довольно быстро, а спокойствия прибавляет).

К типичным вышеуказанным задачам добавляются специфичные задачи обслуживания **msdb**.

Как известно, в базе данных msdb хранится история резервных копий по базам данных. Теперь представим сервер, у которого баз данных более 50, каждые 10-15 минут проходит создание резервное копирование файла транзакций, какой объем

Очистка истории резервного копирования осуществляется через процедуру:

sp_delete_backuphistory [@oldest_date =] «oldest_date»

где***[@oldest_date =] «oldest_date»***

Самая ранняя дата, сохраненная в таблицах журнала резервного копирования и восстановления. Аргумент ***oldest_date*** имеет тип ***datetime*** и не имеет значения по умолчанию

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Открыть узлы системных баз данных.
2. Выполнить резервное копирование системных баз данных.
3. Проверить целостность системных баз данных с помощью инструкции «***DBCC CHECKDB***».
4. Отобразить историю копирования.
5. Очистить историю копирования в базе **msdb**.

4. ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ

Заданием для работы является обслуживание баз данных используемых для выполнения лабораторных работ.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие системные базы данных есть в среде MS SQL Server? Каково их назначение?

2. Какие задачи обслуживания системных баз данных существуют?

3. Каким образом выполняется обслуживание системных баз данных MS SQL Server?

4. Что такое «История резервного копирования»? Каким образом она очищается?

РАБОТА №13 ПРАКТИЧЕСКАЯ. РАБОТА С ЖУРНАЛОМ АУДИТА БАЗЫ ДАННЫХ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью данной работы является получение практических навыков работы с журналами аудита базы данных.

Для достижения цели работы необходимо решить следующие задачи.

- Изучить основные журналы, используемые для отображение событий происходящих с сервером и с его окружением.
- Ознакомиться с основными средствами просмотра данных этих журналов, изучить их возможности.
- Выполнить практическое задание по работе с журналами сервера изученными средствами.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

При работе сервера необходимо отслеживать и события, связанные с его работой. Главная цель при этом – как можно быстрее обнаруживать проблемы, возникающие на сервере, и оперативно реагировать на них.

Самое простое средство мониторинга работы MS SQL Server 2005 – журналы сервера. Считается, что рабочий день администратора на предприятии должен начинаться с просмотра журналов событий на всех серверах. Для MS SQL Server начиная с версии 2005 выделяются 4 журнала:

- журнал событий самого SQL Server;
- журнал событий SQL Server Agent;
- журнал событий операционной системы Windows;
- журнал событий приложений Windows.

Журналы можно просматривать разными способами. Самый простой и рекомендованный – использовать просмотрщик, который встроен в SQL Server Management Studio. Запустить его можно из контейнера **Management | SQL Server Logs (Управление | Журналы SQL Server)**. В списке журналов нужно щелкнуть правой кнопкой мыши

по требуемому журналу и в контекстном меню выбрать View SQL Server Log (Просмотреть журнал SQL Server). Откроется окно просмотрщика журналов (рис. 13.1).

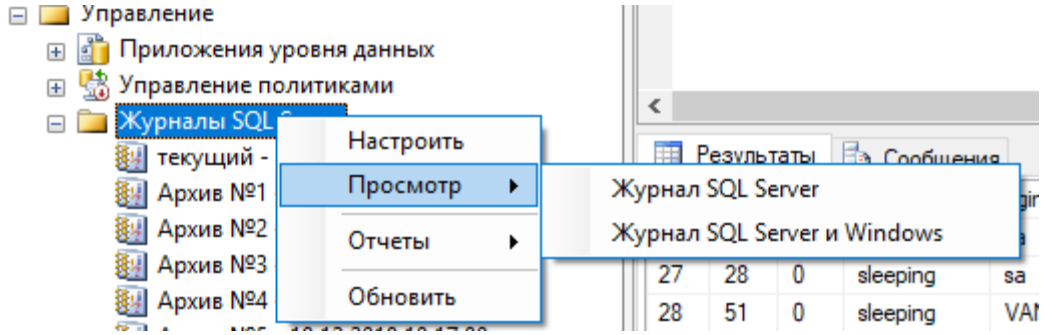


Рис.13.1 Открытие окна просмотра журналов

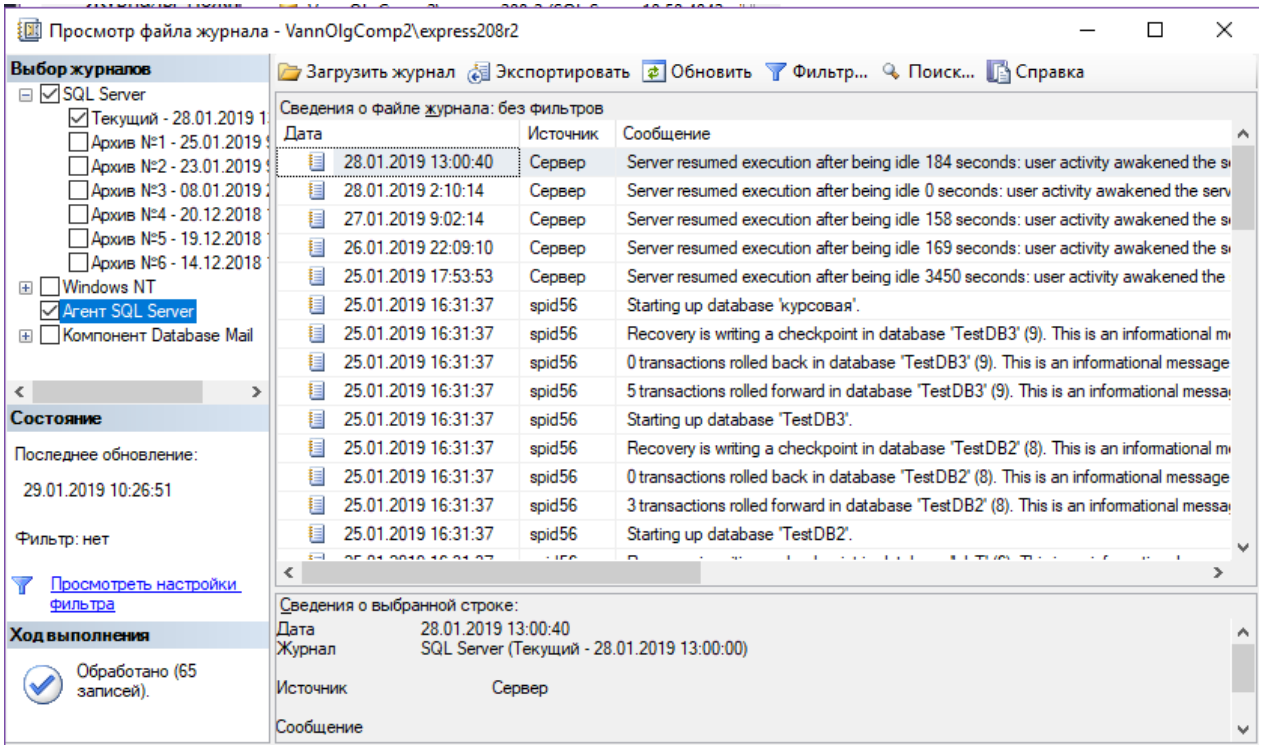


Рис.13.2 Окно просмотра журналов

При помощи просмотрщика журналов можно просматривать не только журналы SQL Server, но и журналы SQL Server Agent, Windows и Database Mail, можно экспортировать данные из журналов при помощи кнопки Export (Экспортировать), в том числе и в очень удобный для загрузки в базу данных формат CSV, можно настраивать фильтрацию и производить поиск нужной информации.

Журналы событий SQL Server можно просматривать и «вручную», при помощи любого текстового редактора. По умолчанию

они находятся в каталоге C:\Program Files\Microsoft SQL Server\MSSQL\LOG. Там же находятся и журналы SQL Server Agent.

Если вам нужен более подробный протокол событий, происходящий на сервере, можно воспользоваться параметром C2 Audit Tracing. Его можно установить на вкладке Security (Безопасность) свойств сервера в Management Studio. В этом режиме в каталоге Data (Данные) для данного экземпляра сервера будут автоматически создаваться текстовые файлы с очень подробной информацией, которая может потребоваться для аудита в соответствии со стандартом безопасности C2.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Открыть журнал сервера.
2. Выявить основные события сервера за последние 12 часов.
3. Отобразить события в виде отчёта.
4. Выгрузить журнал в текстовый файл и файл «.csv».
5. Определить положение и открыть с помощью текстового редактора файл, соответствующий журналу.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие события могут происходить при работе сервера? Зачем их отслеживать?
2. Каким образом сохраняются данные о событиях сервера?
3. Какие типы журналов сервера выделяются?
4. Какие средства просмотра журналов существуют? Каковы их возможности?
5. Как запустить просмотрщик событий сервера?

РАБОТА №14 ПРАКТИЧЕСКАЯ. МОНИТОРИНГ НАГРУЗКИ СЕРВЕРА

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является получение практических навыков отслеживания работы сервера MS SQL Server.

Для достижения цели работы необходимо решить следующие задачи:

- изучить средств мониторинга Activity monitor, Denali Activity Monitor;
- ознакомиться с использованием системных процедур sp_who и sp_who2.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Мониторинг экземпляров SQL Server и баз данных позволяет получить информацию, необходимую для диагностики и устранения неполадок производительности SQL Server, а также для тонкой настройки SQL Server.

Для нормального функционирования SQL Server, администратор баз данных должен постоянно следить за производительностью, иметь набор метрик, которые оперативно могут сообщить о деградации в работе системы. Вовремя получать уведомления, когда текущая нагрузка на сервер выходит за рамки базовых показателей системы, и принять адекватные меры.

Activity Monitor отслеживает наиболее важные показатели эффективности SQL Server. Чтобы получить их, он выполняет запросы к экземпляру SQL Server каждые 10 секунд. Мониторинг осуществляется только когда инструмент открыт, поэтому побочный эффект от его использования минимальный.

Все метрики показаны на 5 разных панелях: Overview (Обзор), Processes (Процессы), Resource Waits (Ожидания ресурсов), Data File I/O (Ввод/вывод файлов данных), и Recent Expensive Queries (последние затратные запросы).

Overview (Общие сведения). Содержит графики Processor Time (Процессорное время), Number of Waiting Tasks (Количество ожи-

дающих задач), Database I/O (Ввод-вывод в базе данных) и Number of Batch Requests/second (Количество пакетных запросов в секунду).

Activity Monitor можно открыть в SQL Server Management Studio toolbar используя иконку Activity Monitor на панели, сочетанием клавиш Ctrl+Alt+A или через контекстное меню в Object Explorer.

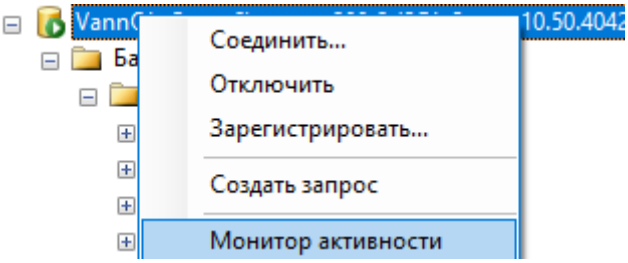


Рис.14.1 Запуск Activity Monitor

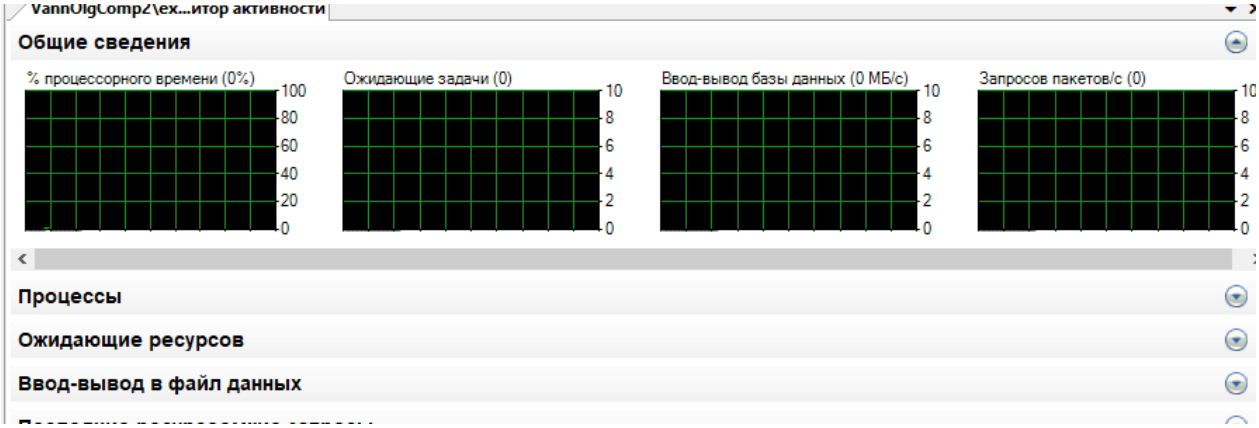


Рис.14.2 Окно монитора активности

Как уже было сказано выше, Activity Monitor отслеживает только заранее определенный набор наиболее важных показателей производительности SQL Server. Дополнительных параметров указать нельзя, нельзя и удалить что-то из показателей. Мониторинг возможен только в режиме реального времени. Нет возможности сохранить результаты мониторинга для последующего анализа. Таким образом Activity Monitor – это полезный инструмент для беглого анализа и поиска неисправностей, но он не подходит для детального сбора информации, т. к. в нём отсутствует возможность гибкой настройки счётчиков производительности, указания пороговых значений и нет возможности сбора исторических данных.

Системная процедура `sp_who`.

Предоставляет сведения о текущих пользователях, сеансах и процессов в экземпляре Microsoft Компонент SQL Server Database Engine. Данные могут быть отфильтрованы, чтобы возвращать только те процессы, которые не простаивают, принадлежат конкретному пользователю или принадлежат определенному сеансу.

Общий формат запуска процедуры

sp_who [[@loginame =] «login» | session ID | «ACTIVE»]

Аргументы

[@loginame =] «login» – определяет процессы, принадлежащие конкретному имени входа.

Session ID – идентификатор сеанса является идентификационным номером сеанса, принадлежащего SQL Server экземпляра.

«ACTIVE» – исключает сеансы, ожидающие следующей команды от пользователя.

Если значение не указано, эта процедура возвращает все сеансы, принадлежащие экземпляру.

Пример использования:

```
USE master;
GO
EXEC sp_who «10» --specifies the process_id;
GO
```

У хранимой процедуры sp_who есть недокументированный вариант – sp_who2. Эта хранимая процедура запускается на выполнение точно так же, но возвращает более подробную информацию.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Запустить мониторинг активности.
2. Выявить выполняемые процессы, время ожидания ресурсов и другие отображаемые параметры.
3. Запустить процедуру ***«sp_who»***:

- без параметров;
- с параметром указывающим на конкретное имя входа;
- с параметром, указывающим на идентификатор процесса.

4. Аналогично выполнить запуск процедуры ***«sp_who2».***
5. Задокументировать работу процедур в виде отчёта.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что собой представляет процесс мониторинга, какие его задачи?
2. Какие средства мониторинга рассматриваются в этой работе?
3. Какие показатели работы сервера позволяет получать Activity Monitor?
4. Как запускается Activity Monitor?
5. Какие показатели работы позволяют получать хранимые процедуры «sp_who», «sp_who2»?
6. Какие параметры запуска могут быть использованы для данных процедур?

РАБОТА №15 ПРАКТИЧЕСКАЯ. УСТАНОВКА И НАСТРОЙКА СЕРВЕРА БД ORACLE

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является получение практических навыков установки и настройки сервера БД ORACLE. Для достижения цели необходимо решить следующие задачи:

- Изучить основные особенности сервера БД ORACLE.
- Изучить данные об разработчиках и источниках получения компонентов установки.
- Изучить основные этапы установки.
- Выполнить установку и настройку сервера на пользовательском компьютере.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

СУБД Oracle – это мощный программный комплекс, позволяющий создавать приложения любой степени сложности. Ядром этого комплекса является база данных, хранящая информацию, количество которой за счет предоставляемых средств масштабирования практически безгранично. С высокой эффективностью работать с этой информацией одновременно может практически любое количество пользователей (при наличии достаточных аппаратных ресурсов), не проявляя тенденции к снижению производительности системы при резком увеличении их числа.

Механизмы масштабирования в СУБД Oracle последней версии позволяют безгранично увеличивать мощность и скорость работы сервера Oracle и своих приложений, просто добавляя новые и новые узлы кластера. Выход из строя отдельных узлов кластера также не приводит к остановке приложения.

Встраивание в СУБД Oracle JavaVM, полномасштабная поддержка серверных технологий (Java Server Pages, Java-сервлеты, модули Enterprise JavaBeans, интерфейсы прикладного программирования CORBA), привело к тому, что Oracle на сегодняшний день де-факто является стандартом СУБД для Internet.

Особенностью СУБД Oracle является многоплатформенность, так как она поставляется практически для всех существующих на сегодня операционных систем. СУБД Oracle одинаково хорошо работает на любой платформе.

Структура Oracle включает:

- Программное обеспечение Oracle.
- Базы данных – Database.

В каждой базе данных имеется много схем, и имя схемы (Schema) так же является именем пользователя для доступа в данную схему. В каждой схеме имеется система таблиц, представлений (view), функций (function), процедур (procedure), пакетов (package) и других объектов. То есть после установки программы Oracle, можно создать одну или более баз данных.

2.1. Описание этапов установки СУБД Oracle

1. Загрузка компонента установки

Компонент установки загружаются с официальной страницы Oracle https://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html.

Установочные файлы предоставляются бесплатно, однако при этом требуется наличие регистрационной записи на сайте Oracle.

Установка требует ввода электронной почты и пароля соответствующих учётной записи.

Рис.15.1 Форма ввода электронной почты

2. Задание опций установки

При задании опций установки можно задать создание и конфигурирование баз данных.

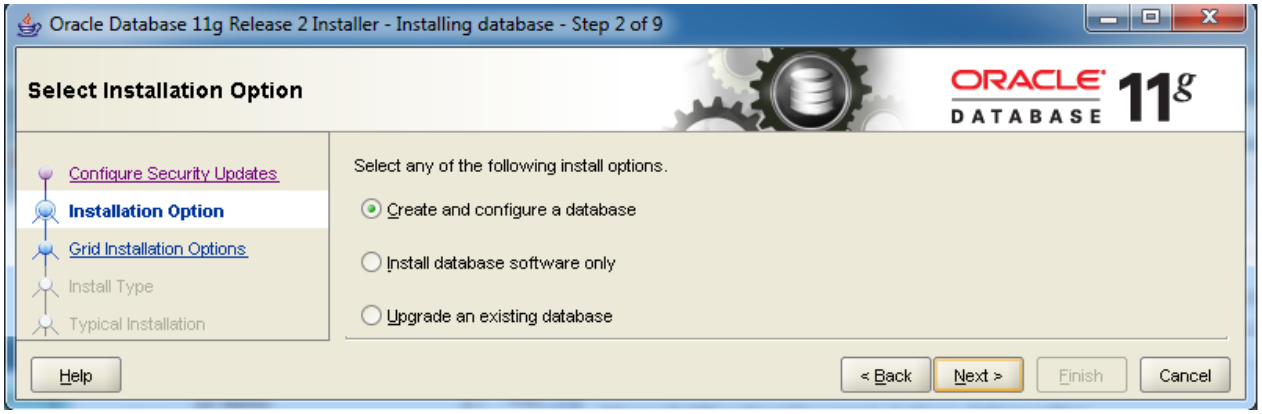


Рис.15.2 Задание исходных опций установки

3. Выбор класса системы

Возможна установка системы класса настольной системы (Desktop class) и класс серверной системы (Server class).

Для обучения и небольших компаний достаточен Desktop class.

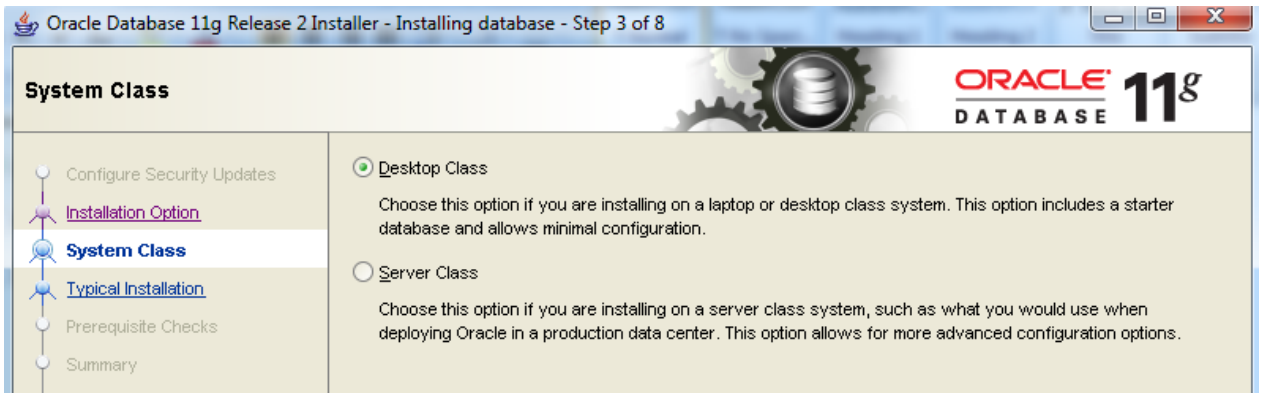


Рис.15.3 Выбор класса системы

4. Выбор базовых настроек установки

В качестве кодировки символов (**Character Set)**: целесообразно выбрать – Unicode (AL32UTF8).

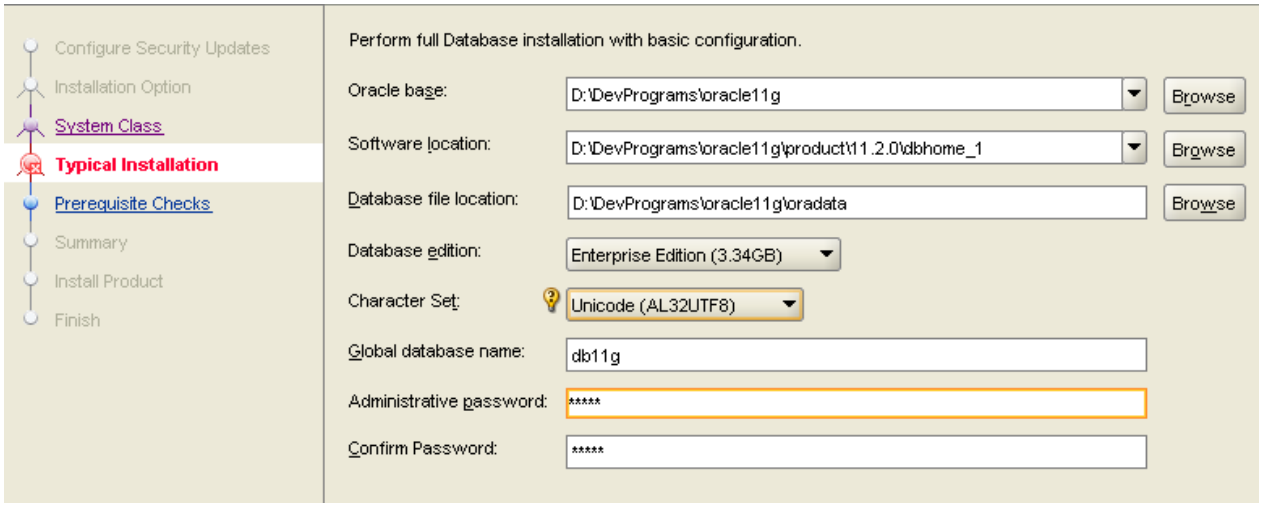


Рис.15.4 Задание базовых настроек установки

5. Непосредственно выполнение установки

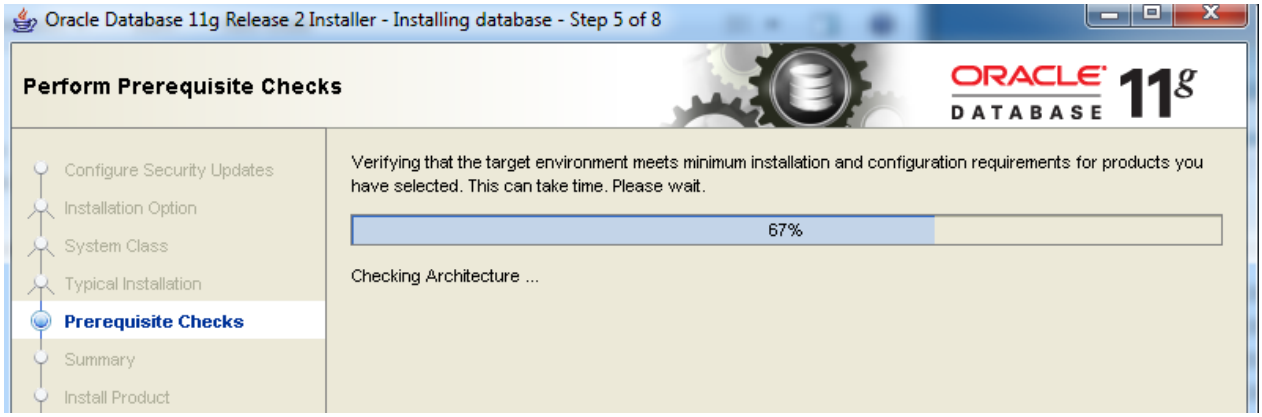


Рис.15.5 Процесс установки

6. Отображение результатов установки

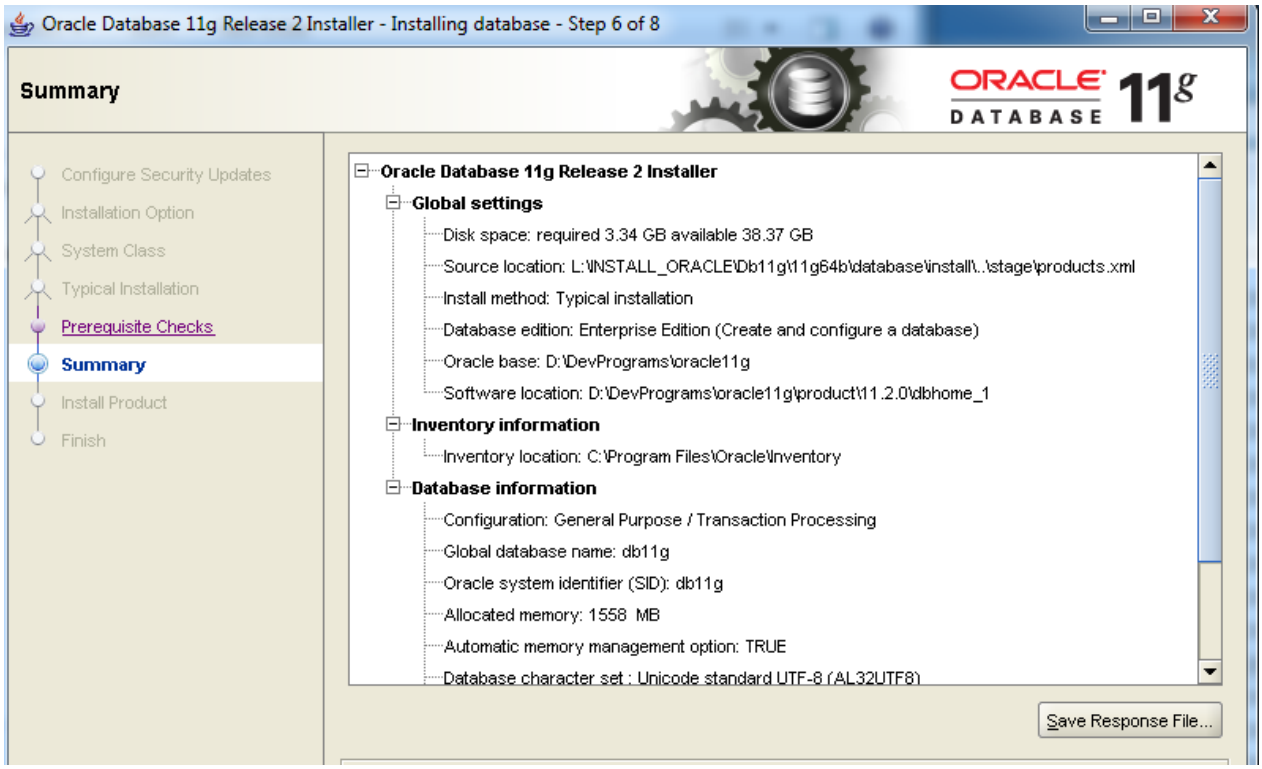


Рис.15.6 Отображение результатов установки

7. Создание базы данных с именем, указанным при задании основных параметров

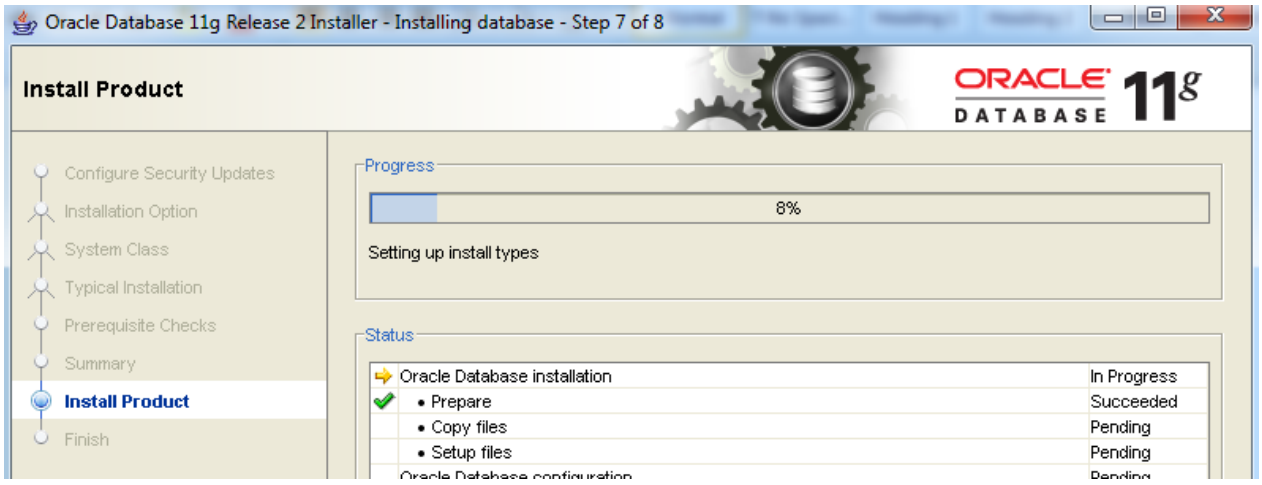


Рис.15.7 Создание базы данных

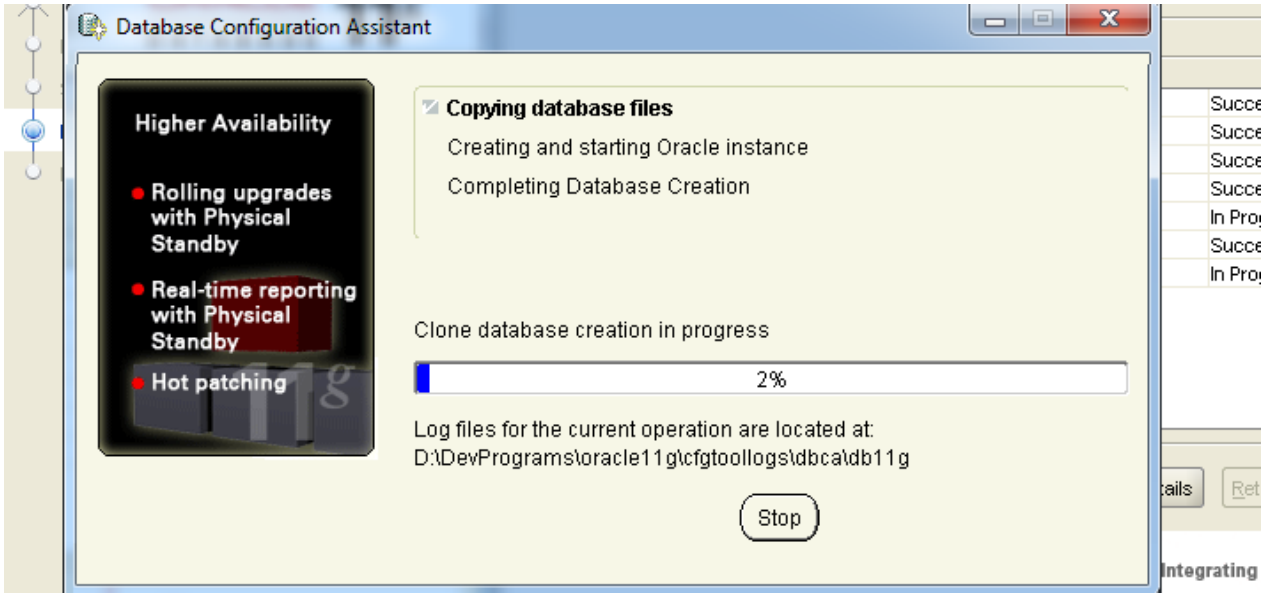


Рис.15.8 Копирование файлов базы данных

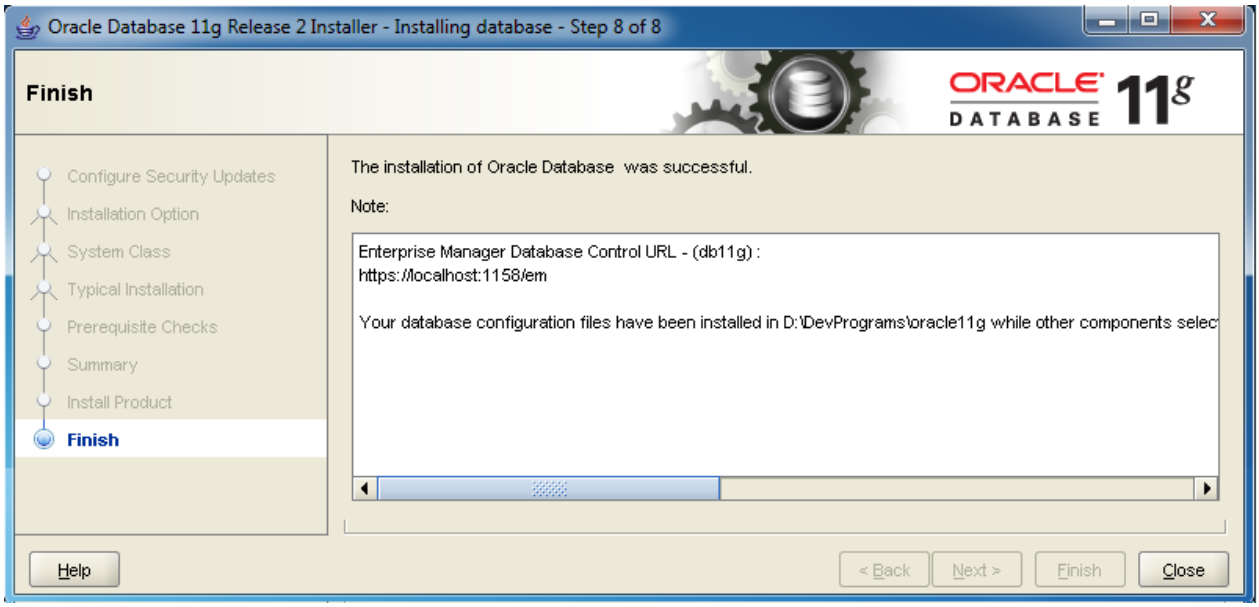


Рис.15.9 Завершение установки

8. *Настройка брандмауэра* ***Windows***

В случае блокировки Oracle брандмауэра Windows. Необходимо нажать кнопку «Allow access» чтобы разрешить работу процесса oracle (задать разрешение для пропуска процесса через брандмауэра).

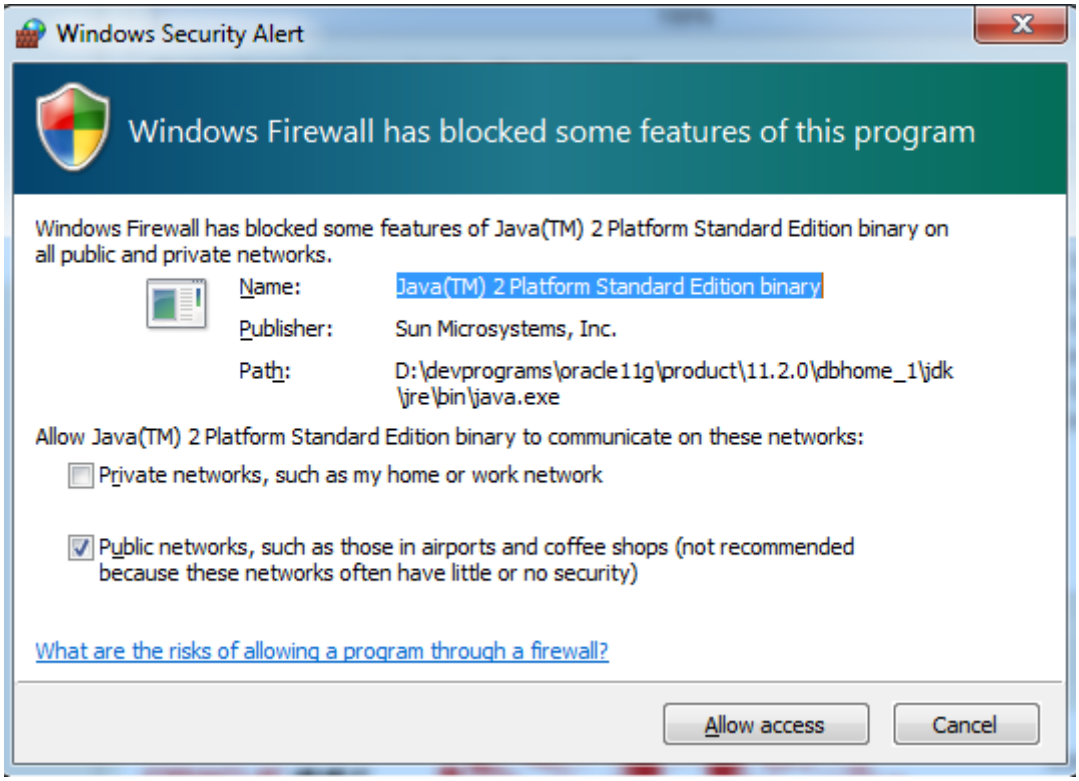


Рис.15.10 Задание пропуска процесса Oracle
через брандмауэра Windows

9. Проверка установки и работы служб ORACLE.

После установки СУБД Oracle отображается в виде системных служб.

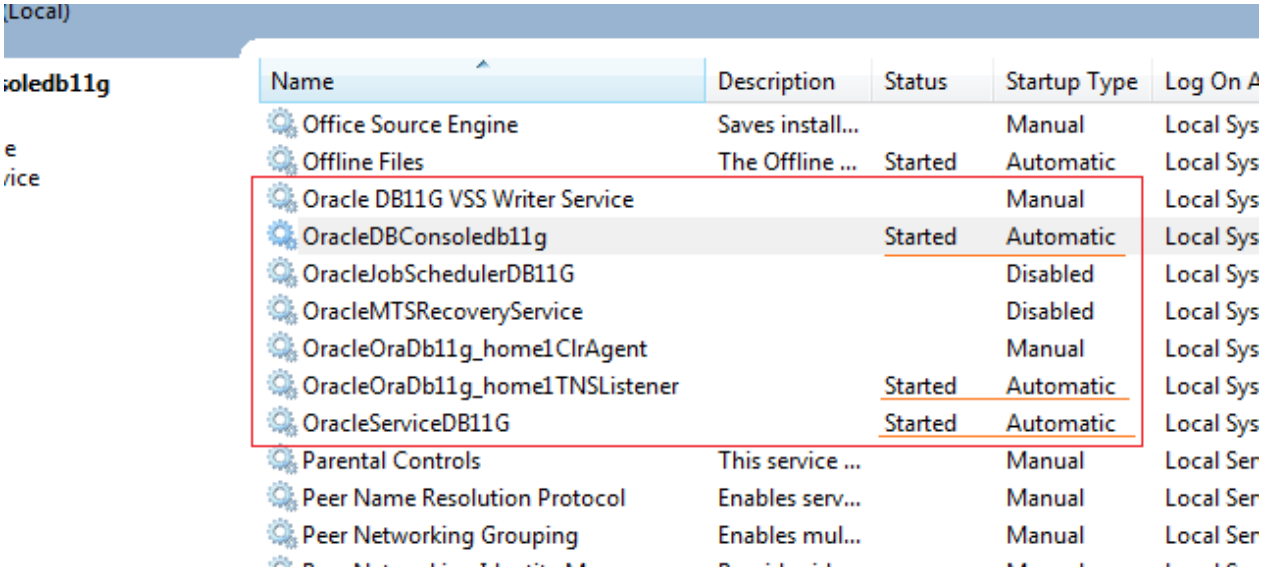


Рис.15.11 Отображение служб Oracle в диспетчере заданий

2.2. Описание служб Oracle

1. Oracle DB Console

Это служба для позволяющая осуществлять доступ к СУБД по протоколу http через браузер. Для этого используется адрес – https://localhost:1158/em.

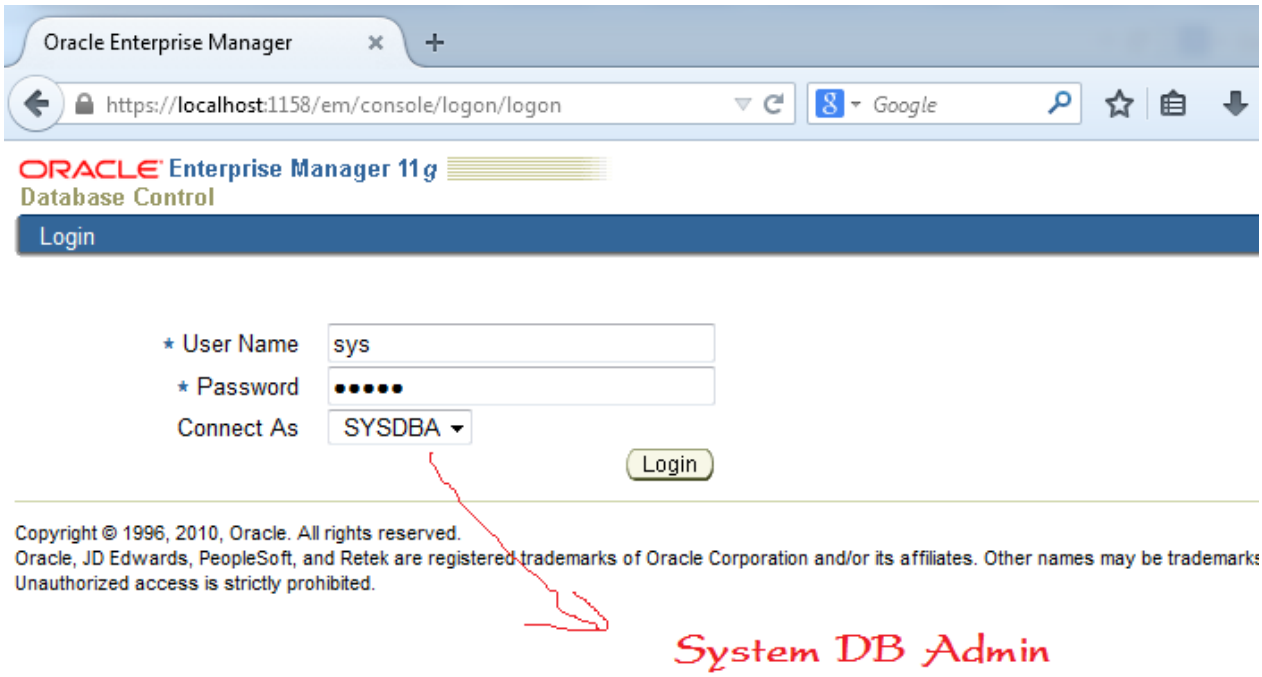


Рис.15.12 Страница доступа к СУБД через браузер

Используя **OracleDBConsole,** можно отслеживать, как работает сервис (DB), как используются ресурсы, как работают пользователи и в каком порядке выполняются команды.

При использовании **Oracle** только с целью обучения сервис **OracleDBConsole** может быть выключен.

2. SQL Plus

SQL Plus – это простой консольный инструмент, позволяющий выполнять команды **SQL**.

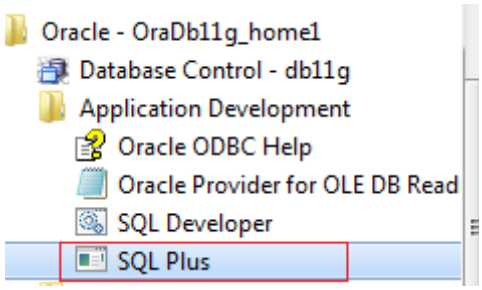


Рис.15.13 Запуск службы **SQL Plus**

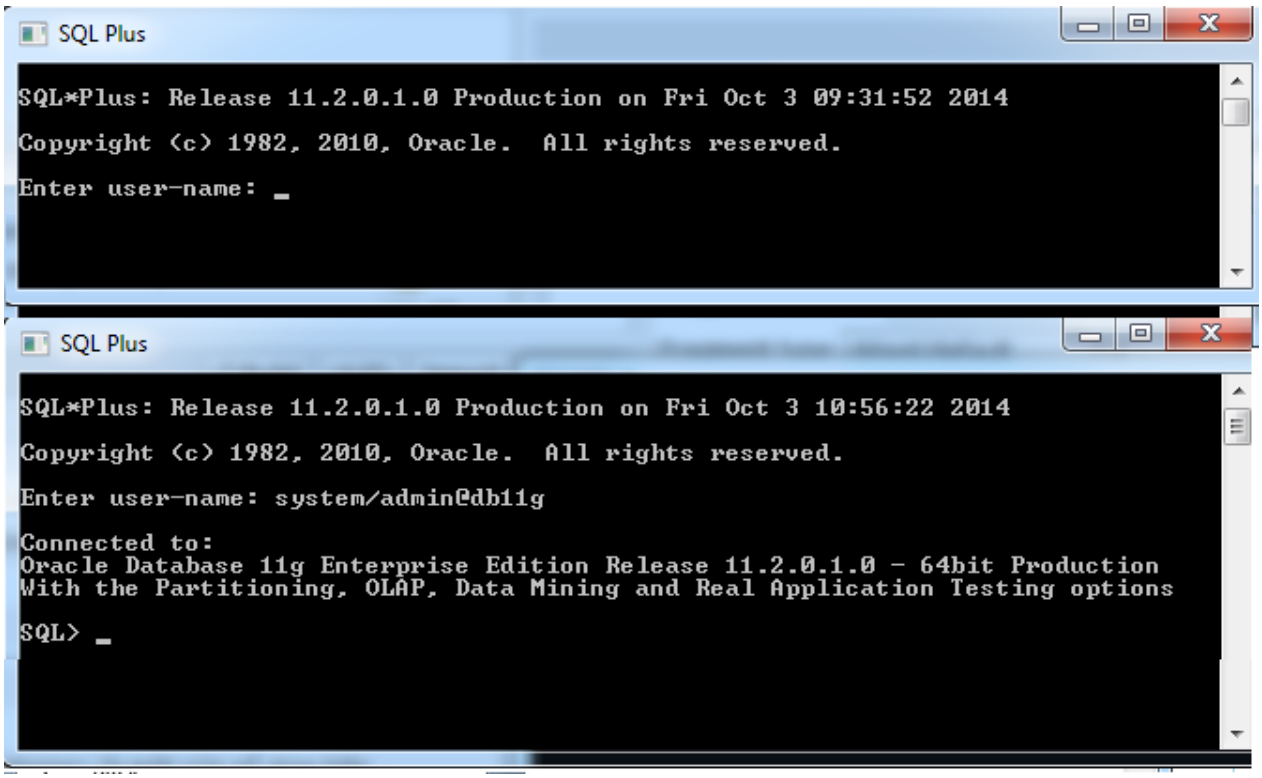


Рис.15.14 Работа в среде SQL Plus

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Скачать компоненты установки.
2. Установить СУБД ORACLE. При установке должна быть создана, как в рассмотренном примере, база «DBOracle11».
3. Создать в базе данных 2 таблицы командами DDL.
4. Заполнить таблицы данными с помощью команд DML.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каковы особенности и архитектура СУБД Oracle?
2. Из какого источника можно получить компоненты установки Oracle?
3. Каковы основные этапы установки?
4. В каком виде представляется установленная система Oracle?
5. Какие службы используются для управления СУБД?
6. Какие службы используется для доступа к базам данных?

СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Цель самостоятельной работы обучающихся – получить новые знания по дисциплине «Управление и автоматизация баз данных».

Самостоятельная работа необходима для формирования у обучающихся способности самостоятельно решать задачи профессиональной деятельности, формирования умения и навыков планирования времени, формирования стремления развиваться и совершенствоваться.

Виды самостоятельной работы обучающихся указаны в табл. 1.

Таблица 1

Виды самостоятельной работы

№ п/п	Вид СРС
1	Определение требований к БД и разработка схемы базы данных в рамках работы над КП
2	Разработка сценариев работы с данными в рамках работы над КП
3	Разработка механизмов реализации сценариев работы с данными в виде хранимых процедур и триггеров в рамках работы над КП
4	Разработка клиентской части КП
5	Выполнение индивидуальных заданий по теме «Администрирование баз данных и серверов»

УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Основная литература

1. Перлова, О. Н. Соадминистрирование баз данных и серверов [Электронный ресурс] : учебник для студентов среднего профессионального образования по специальности 09.02.07 Информационные системы и программирование / О. Н. Перлова, О. П. Ляпина. – Москва : Академия, 2018. – 304 с. – Режим доступа: http://www.academia-moscow.ru/catalogue/4831/345911/. – Загл. с экрана.

Дополнительная литература

1. Компьютерные сети [Электронный ресурс] : учебник для среднего профессионального образования по специальностям 09.02.06 Сетевое и системное администрирование, 09.02.07 Информационные системы и программирование / В. В. Баринов [и др.]. – Москва : Академия, 2018. – 192 с. – Режим доступа: http://academia-moscow.ru/catalogue/4831/345920/. – Загл. с экрана.

2. Гохберг, Г. С. Информационные технологии [Текст] : учебник для образовательных организаций, реализующих программы среднего профессионального образования по специальностям «Информационные системы и программирование», «Сетевое и системное администрирование» : [для студентов СПО] / Г. С. Гохберг, А. В. Зафиевский, А. А. Короткин. – Москва : Академия, 2017. – 240 с. – Доступна электронная версия: http://academia-moscow.ru/catalogue/4831/297236/

Программное обеспечение и интернет-ресурсы

1. https://msdn.microsoft.com/ru-RU/library/e80y5yhx(v=vs.110).aspx Официальный сайт. Microsoft. Руководство по разработке. Данные и модели. ADO.NET
2. https://docs.microsoft.com/ru-ru/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-2017 Официальный сайт. Microsoft Руководство по разработке. Типы данных Transact SQL

ПРИЛОЖЕНИЕ

Простейшие операторы манипуляции данными

Вставка строки

```
INSERT INTO имяТабл ([имяПоле1], [имяПоле2],[…])
VALUES(1значение1, значение>[,…])
```

Изменение значения поля в строке

```
update имяТаблицы
set имяСтолбца = значение
[where условие];
```

Удаление строки

```
delete from имя_таблицы
[where условие];
```

Стандартные функции SQL Server

Агрегативные функции

Возвращают суммарные значения.

AVG	COUNT(*)	MIN
COUNT	MAX	SUM

Функции манипуляции датой и временем

DATEADD	Добавляет к дате отрезок
DATENAME	Возвращает часть даты строкой
GETDATE	Текущая дата
DATEDIFF	Вычисляет разницу
DATEPART	Возвращает определенную часть заданной даты

Математические функции

ABS	Модуль
RAND	Генерация случайного числа
ROUND	Округление
FLOOR	Округление в меньшую сторону

CEILING	Округление в большую сторону
SQRT	Корень квадратный
POWER	Возведение в степень
EXP	Вычисление экспоненты
LOG	Натуральный логарифм
LOG10	Десятичный логарифм
PI	Значение π
DEGREES	Возвращает градусы из радиан
RADIANS	Преобразует градусы в радианы
SIN, COS, TAN, COT	Тригонометрические функции
ACOS, ASIN, ATAN , ATN2	Тригонометрические функции
SIGN	Возвращает

Niladic-функции

Эти функции возвращают различные системные значения.

CURRENT_TIMESTAMP – текущая дата	SYSTEM_USER - пользователь системы
CURRENT_USER – текущий пользователь (например, DBO)	SESSION_USER – сессия

Функции для манипуляции со строками

ASCII	Переводит в соответствующую кодировку
STR	Конвертирует в строку
SPACE	Возвращает пробелы
CHAR	Возвращает номер символа в кодировке
REPLICATE	Удваивает строку
STUFF	Вставляет строку в другую
CHARINDEX	Порядковый номер символа,
REVERSE	Зеркалирует строку
SUBSTRING	Возвращает подстроку
RIGHT	Возвращает правую часть строки
UPPER	Возвращает строку с верхнем регистре
LOWER	Возвращает в строку в нижнем регистре
LTRIM	Удаляет пробелы из начала строки
RTRIM	Удаляет пробелы из конца строки

Системные функции

HOST_NAME	Имя сервера
DB_ID	Возвращает номер БД
DB_NAME	Возвращает имя БД
COL_NAME	Возвращает имя колонки в таблице
COL_LENGTH	Возвращает длину колонки в таблице
DATALENGTH	Возвращает размер данных поля
ISNULL	Проверяет на значение NULL

Функции для преобразования различных типов данных

CAST	Конвертирует типы данных
CONVERT	Конвертирует типы данных

Пример отчета по работе №2

Задание: Разработать БД, отображающую музыкальные произведения, их исполнителей, авторов и музыкальные стили

Объекты:

1. Произведение – play; соответствует музыкальному произведению как таковому, может не исполненному никем.
2. Person – содержит данные об авторах, исполнителях, вообще всех персонах, как то связанных с музыкальными произведениями.
3. Стиль – style – содержит данные об музыкальных стилях.
4. Ispoolnenie – содержит данные об исполнении музыкального произведения каким либо исполнителем.

Атрибуты:

- Произведения (proizv):

1. id_play – числовой тип integer; код про изведения искусственный атрибут, введен для идентификации произведения.
2. Название (play) – текстовый тип varchar(50).
3. id_avts – числовой тип integer; код автора слов используется для указания на автора, сочинившего слова.
4. id_avtm – числовой тип integer; код автора музыки используется для указания на автора, сочинившего музыку.

- Стиль (style):

1. id_style – искусственный атрибут, используется для обозначения стиля числовой тип integer.
2. Название (style) – текстовый тип varchar(50).

- ispolnenie:

1. id_play – код исполнения – искусственный атрибут, введен для обознаяения произведения.
2. id_avt – код исполнителя, ссылка на таблицу person– числовой тип integer.
3. id_style – числовой тип integer ссылка на значение кодов персон в табл person.

Связи:

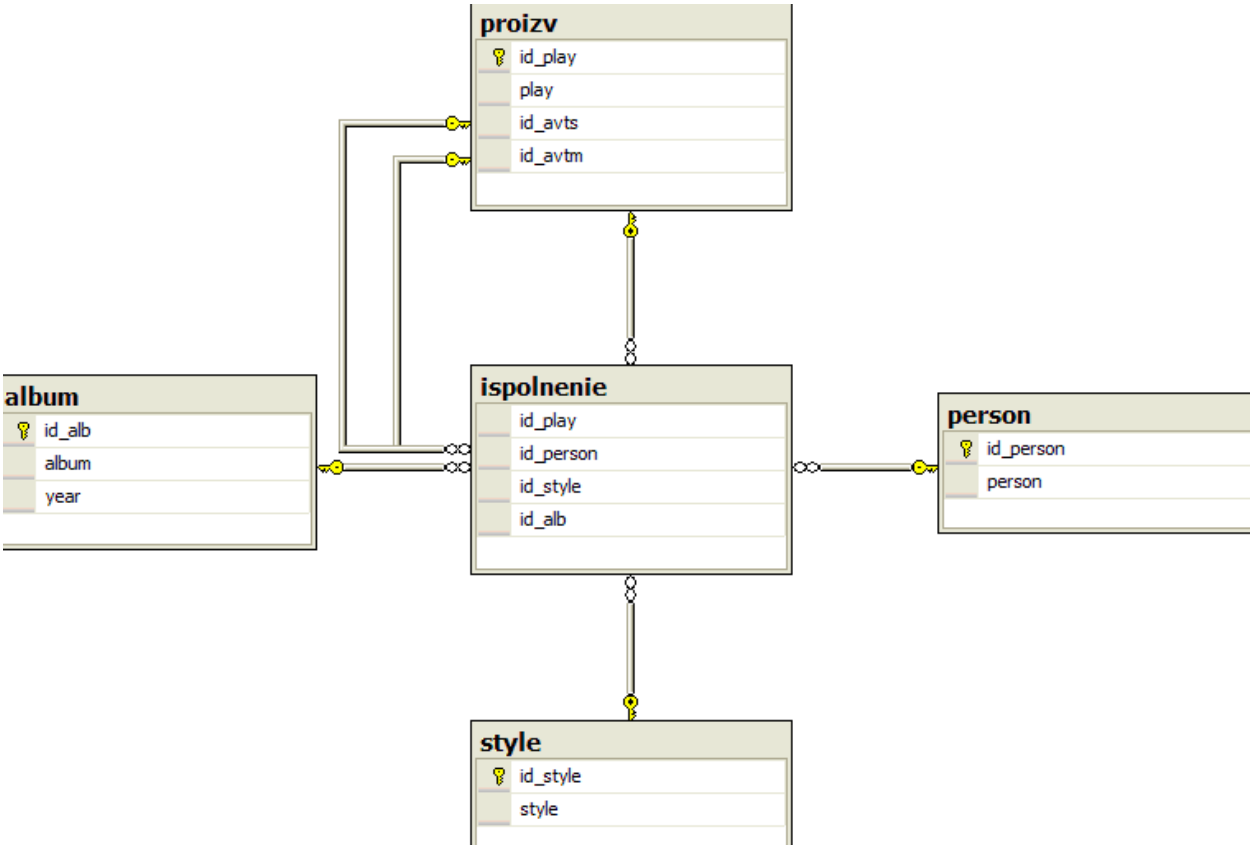
1. Произведение связано с ispolnenie связью один к многим, так как одно произведение может иметь много вариантов исполнений.

2. Группа связана с ispolnenie связью один к многим, так как одна группа может записать много исполнений произведений.

3. Стиль связан с ispolnenie связью один к многим, так как в одном стиле может быть сделано много исполнений.

4. Альбом связан с ispolnenie связью один к многим, так как один альбом может содержать много исполнений.

Диаграмма связей в БД



Таблицы, соответствующие выделенным отношениям

Таблица album

Table - dbo.proizv | Table - dbo.ispolnenie | Table - dbo.person | Table - dbo.style | **Table - dbo.album**

id_alb	album	year
1	Стихия огня	2006
2	Смутное время	1997
3	Неформат	2000
4	See You The Other Side	1996
5	Aska	2005
6	Крылья	2005
NULL	NULL	NULL

Таблица person

Table - dbo.proizv | Table - dbo.ispolnenie | **Table - dbo.person** | Table - dbo.style | Table - d

id_person	person
1	Aska
2	Catharsis
3	Легион
4	Ария
5	Korn
6	Маврин
NULL	NULL

Таблица proizv

Table - dbo.proizv | Table - dbo.ispolnenie | Table - dbo.person | Table - dbo.style | Table - dbo.

id_play	play	id_avts	id_avtm
1	Вольная птица	3	3
2	Я свободен	4	6
3	Стая	6	6
4	Twisted Transistor	5	5
5	Angels of war	1	1
6	Крылья	2	2
NULL	NULL	NULL	NULL

Таблица style

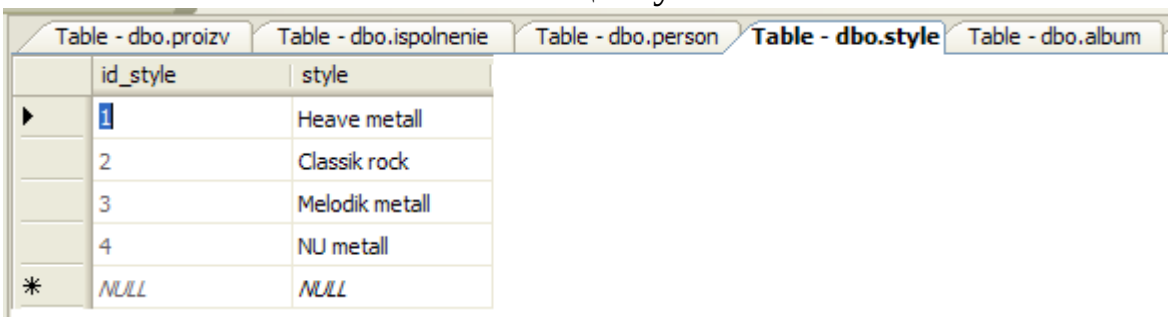


Table - dbo.proizv | Table - dbo.ispolnenie | Table - dbo.person | **Table - dbo.style** | Table - dbo.album

id_style	style
1	Heave metall
2	Classik rock
3	Melodik metall
4	NU metall
NULL	NULL

Таблица ispolnenie

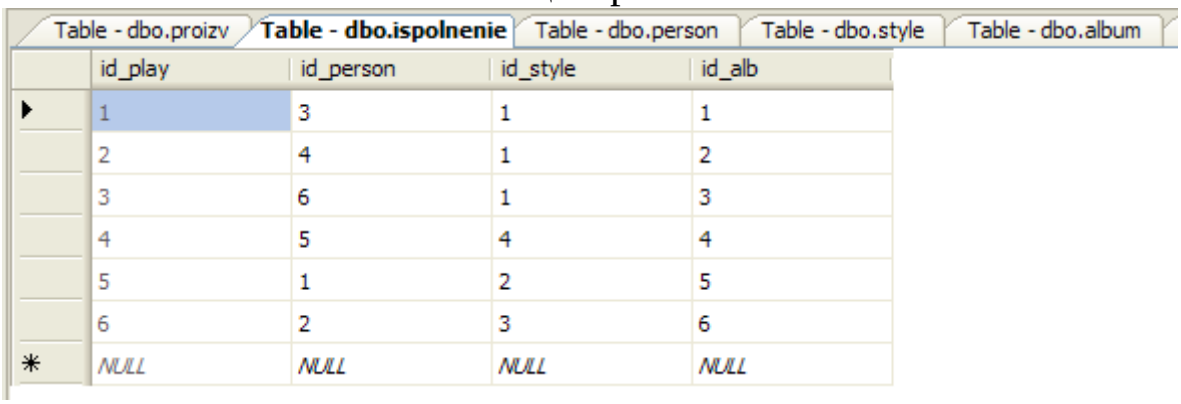


Table - dbo.proizv | **Table - dbo.ispolnenie** | Table - dbo.person | Table - dbo.style | Table - dbo.album

id_play	id_person	id_style	id_alb
1	3	1	1
2	4	1	2
3	6	1	3
4	5	4	4
5	1	2	5
6	2	3	6
NULL	NULL	NULL	NULL

Пример выполнения работы №5

Дана информация об установленном оборудовании.

Отдел	НачОтдела	ЦехИнвНомер	Модель	Стоимость	СрокСлужбы
Цех1	Сидоров	2	1К62	160000	4.5
Цех1	Сидоров	3	1Т62	160000	2
Цех2	Петров	2	2Ф14	260000	5

1. Строим диаграмму зависимостей

А) В качестве потенциального ключа приняты атрибуты (Отдел, ЦехИнвНомер. По значению данных атрибутов можно определить любой кортеж (строку, запись).

В) Частичные зависимости:

Отдел →НачОтдела,

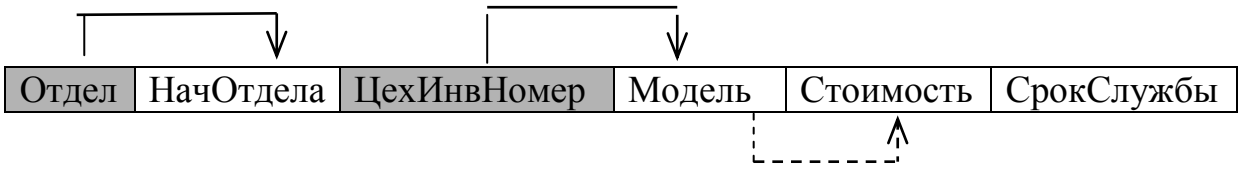
ЦехИнвНомер→Модель

Данная зависимость Отдел →НачОтдела показывает, что по номеру отдела можно определить его начальника. Зависимость ЦехИнвНомер→ Модель показывает, что по цеховому номеру станка можно определить его модель, то есть цеховой номер станка подразумевает некоторый конкретный станок, который имеет некоторую модель.

С) Транзитивная зависимость:

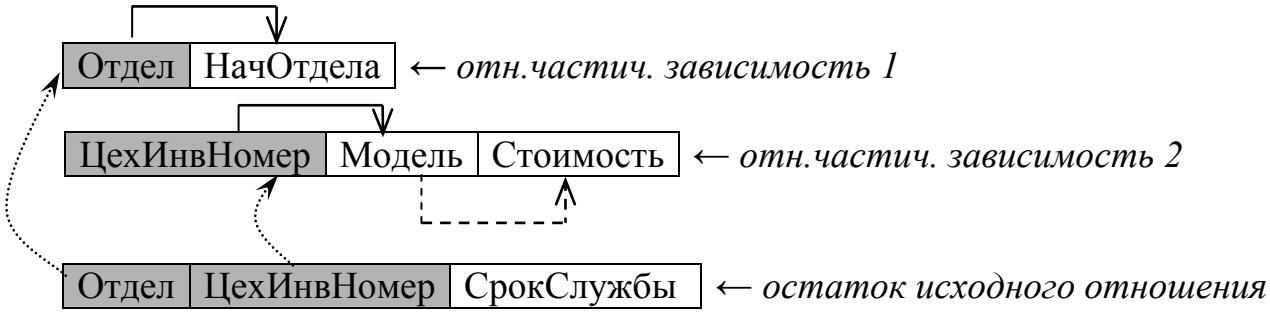
Модель →Стоимость.

Данная зависимость подразумевает то, что стоимость станка определяется его моделью, что соответствует условиям предметной области.



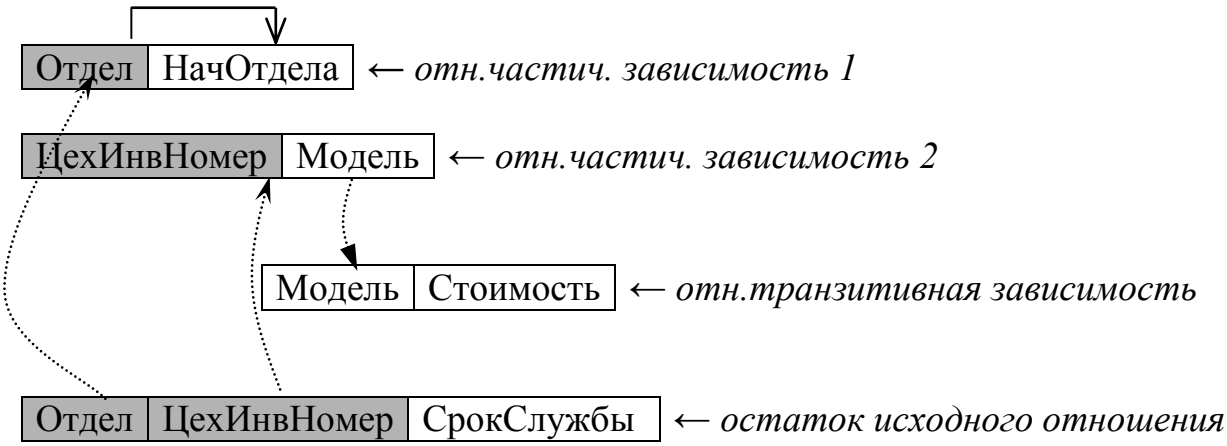
2. Приведение ко 2 нормальной форме

Для приведения ко 2 ф.н. выделяем объекты «Отдел» и «Модель» в отдельные отношения.

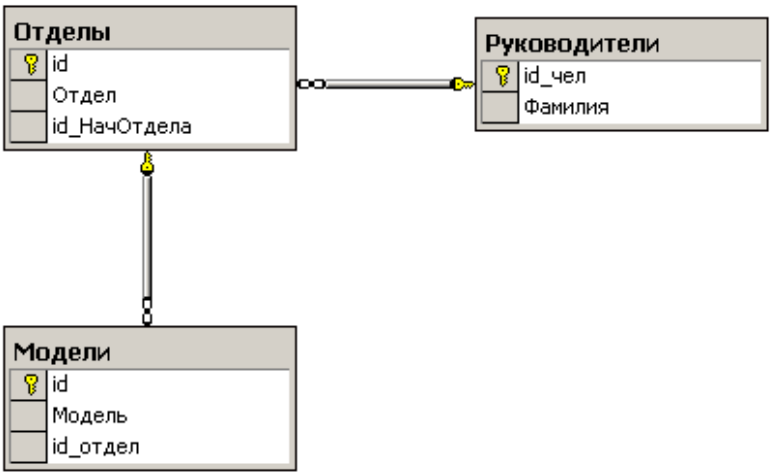


3. Приведение к 3 нормальной форме

Для приведения к 3NF выделяем транзитивную зависимость «Модель-Стоимость» в отдельное отношения.



4. Строим связь отношений в Enterprise Manager



5. Формируем запрос, позволяющий получать требуемую информацию

```
SELECT M.Модель, O.Отдел, R.Фамилия
FROM dbo.Модели M INNER JOIN Отделы O
        ON M.id_отдел = O.id
        INNER JOIN Руководители R
                ON O.id_НачОтдела = R.id_чел
```

Результат:

1К62	Цех1	Иванов
1Т65	Цех2	Петров
С1Е12	Цех2	Петров

СОДЕРЖАНИЕ

Работа №1 Практическая. Построение базы данных в среде одной из суБД 2
Работа №2 Практическая. Построение схемы и словаря базы данных 12
Работа №3 Практическая. Изучение команд администрирования данных для среды одной из СУБД 19
Работа №4 Лабораторная. Разработка требований и конфигурирование корпоративной сети 26
Работа №5 Лабораторная. Разработка механизмов сервера баз данных. Храминые процедуры 32
Работа №6 Лабораторная. Разработка механизмов сервера баз данных. Триггеры 40
Работа №7 Практическая. Установка и настройка сервера Ms SQL Server Express 45
Работа №8 Практическая. Установка и настройка СУБД MY SQL 54
Работа №9 Практическая. Копирование баз данных, импорт экспорт данных в среде Ms SQL Server Express средствами Management Studio 58
Работа №10 Парктическая. Копирование баз данных средствами команд SQL 63
Работа №11 Практическая. Перенос базы данных на другой тип сервера 67
Работа №12 Практическая. Создание механизмов сервера для обслуживания базы данных 70
Работа №13 Практическая. Работа с журналом аудита базы данных 73
Работа №14 Практическая. Мониторинг нагрузки сервера 76
Работа №15 Практическая. Установка и настройка сервера БД ORACLE 80
СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ 90
Учебно-методические материалы по дисциплине 91
Приложение 92