

Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Кузбасский государственный технический университет
имени Т. Ф. Горбачева»

Институт профессионального образования

Кафедра информатики и информационных систем

Составитель О. С. Семенова

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 2023

Рецензенты:

Е. А. Ощепкова, преподаватель кафедры информатики и информационных систем.

Семенова Ольга Сергеевна

Управление и автоматизация баз данных : методические указания для обучающихся специальности СПО 09.02.07 «Информационные системы и программирование» / сост. О. С. Семенова, Кузбасский государственный технический университет имени Т. Ф. Горбачева. – Кемерово, 2023. – Текст : электронный.

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

© Кузбасский государственный
технический университет
имени Т. Ф. Горбачева, 2023
© Семенова О. С.,
составление, 2023

Практическая работа №1 ПОСТРОЕНИЕ БАЗЫ ДАННЫХ В СРЕДЕ ОДНОЙ ИЗ СУБД

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цель работы – получить навыки разработки баз данных в среде MS SQL SERVER Management Studio 2008 (2012).

Задачами работы является изучение архитектуры СУБД MS SQL SERVER; изучение функционала среды MS SQL SERVER; изучение принципов создания модели базы данных; создание базы данных в соответствии с индивидуальным заданием.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Система управления базами данных (СУБД) – вспомогательная система, обеспечивающая работу базы данных. СУБД обеспечивает логически согласованную работу файлов, хранящих данные; использование языка 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.

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 современному распределенному клиент-серверному вычислению 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 обрабатывает все запросы, приходящие на сервер.

Отображение службы MS SQL SERVER в разделе «Службы» диспетчера задач операционной системы показано на рисунке 1.

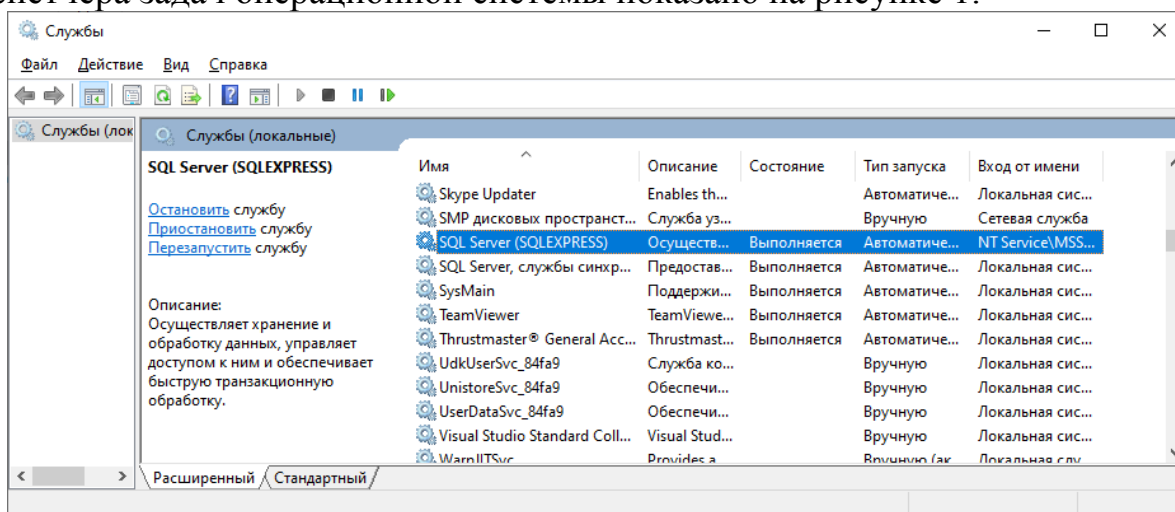


Рисунок 1 – Служба MSSQLSERVER

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.

При запуске MS SQL SERVER Management Studio открывается окно соединения приложения с сервером (рисунок 2).

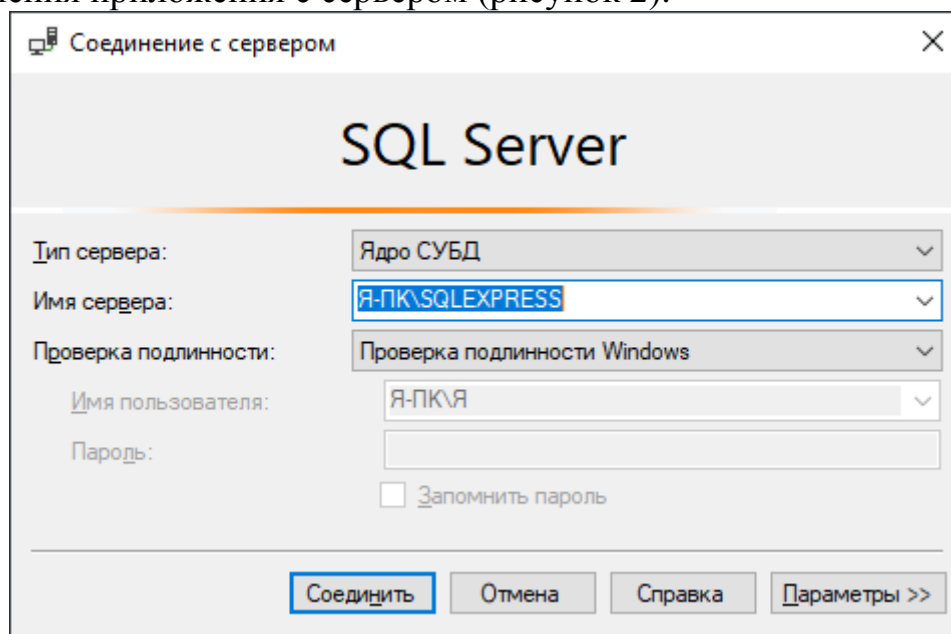


Рисунок 2 – Окно соединения с сервером

Для соединения с сервером необходимо знать его тип, имя сервера, выбрать проверку подлинности, указать имя пользователя и пароль (при необходимости). После соединения с сервером открывается окно приложения MS SQL SERVER Management Studio.

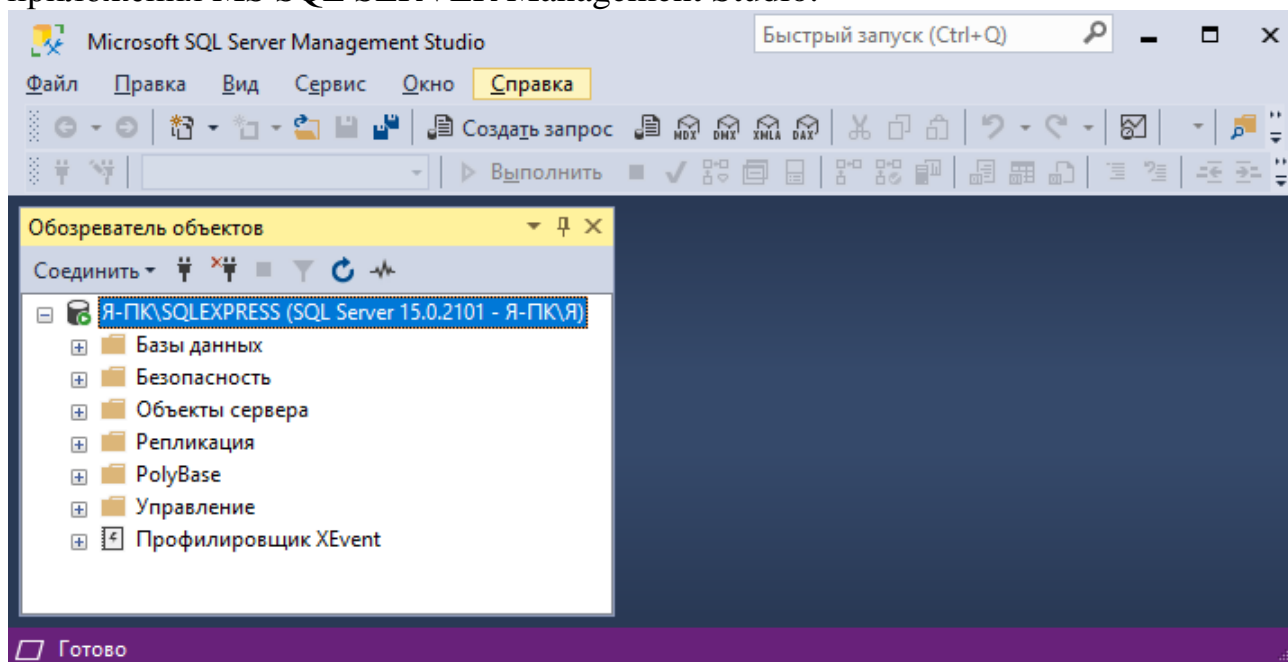


Рисунок 3 – Рабочее окно MS SQL SERVER Management Studio

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 (Базы данных). В среде MS SQL Server база данных содержит в себе различные типы объектов (рисунок 4).

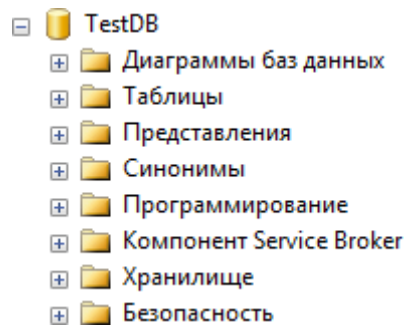


Рисунок 4 Объекты базы данных в среде MS SQL Server

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 работам основоположника теории реляционных баз данных Дейту 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.

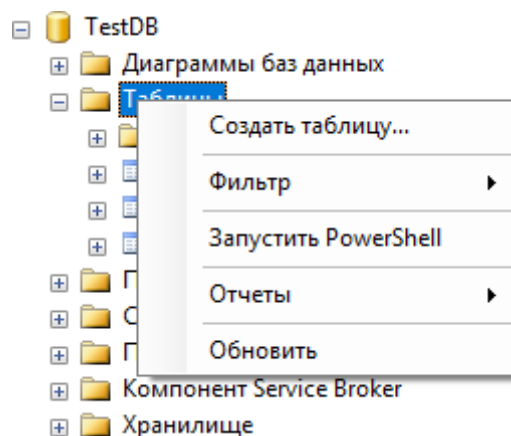


Рисунок 5 – Создание новой таблицы в среде MS SQL Server Management

Studio

Создание таблицы подразумевает создание ее 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.

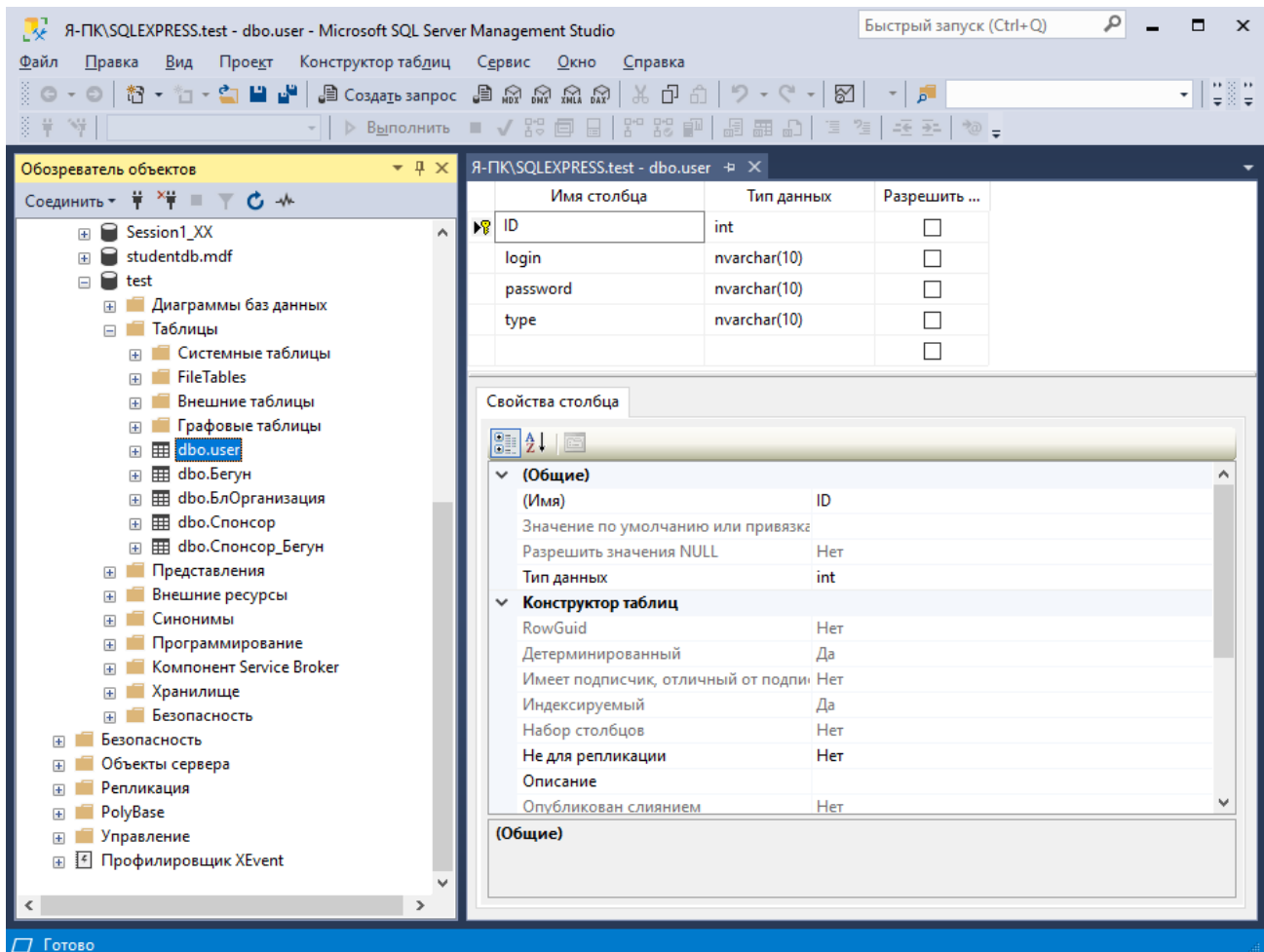


Рисунок 6 – Работа с таблицей в режиме ее модификации

При создании и модификации таблицы ее 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 имен атрибутов использовать латинский шрифт и не использовать внутри имени пробелы и другие служебные символы.

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 представлены в табл. 1.

Таблица 1

Тип данных	Описание
binary(n)	двоичные данные фиксированной длины до 8000 байт
char(n)	строковый тип данных фиксированной длины без поддержки Unicode длиной до 8000 байтов; данные зависят от установленной кодовой страницы;
Varchar(n)	строковый тип с переменной длиной; если ANSI_PADDING=OFF, то будет выполняться удаление конечных пробелов, если ANSI_PADDING=ON, то удаление пробелов производиться не будет.
Nchar(n)	строковый тип, но с поддержкой Unicode; в этом случае для строковых констант надо задавать впереди букву N: N'ABC'.
Nvarchar(n)	строковый тип, как varchar(n), но с поддержкой Unicode.
Int	целый тип длиной в 8 байт и с диапазоном от -263 до 263-1.
Decimal[(p[,s])]	десятичный двоично-кодированный тип с p десятичными разрядами, из которых s - дробных;
Float[(n)]	плавающий (приблизительный) тип длиной в 4 байта
Datetime	тип данных для хранения даты (4 первых байта) и времени (4 последних байта)
Smalldatetime	тип данных для хранения даты (первых 2 байта) и времени (последние 2 байта)
Money	тип данных для хранения больших денежных величин с точностью до 4 знаков после запятой; для хранения данных отводится 8 байт.
Smallmoney	тип данных для хранения нормальных денежных величин с точностью до 4 знаков после запятой в диапазоне от -214 748.3648 до 214 748.3647; для хранения данных отводится 4 байта.
Bit	битовый (логический) тип со значениями 0 и 1; для хранения выделяется 1 разряд байта памяти.

3. *Инкрементность (счетчик)* – автозаполнение поля в добавленной записи неким значением (как правило, числового целого типа).
4. *Ключ* – уникальный идентификатор, характеризующий
5. *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.

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

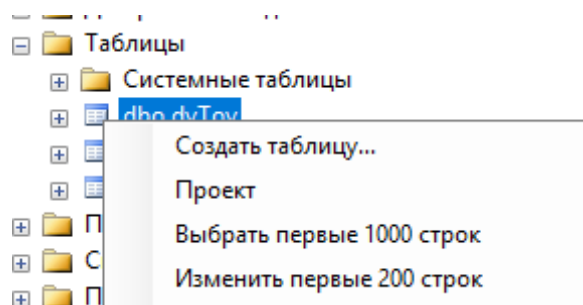


Рисунок 7 – Вызов таблицы для изменения и внесения данных

ID	login	password	type
1	1	1	бегун
3	2	2	координатор
4	4	4	администра
NULL	NULL	NULL	NULL

Рисунок 8 – Вид таблицы вызванной для внесения данных и редактирования

ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ

1. Создать базу данных Группа_ФИО (например, ИСт-191СмирновП.О.) с помощью SQL-запроса.
2. Осуществить проектирование БД для информационной системы «Поликлиника».

Описание предметной области: Поликлиника оказывает различные медицинские услуги. Прием пациентов осуществляется врачами строго по талонам. Для врача каждой специальности определен набор талонов, используемый ежедневно. На каждого пациента заводится медицинская карта. Оплата услуги осуществляется после приема и постановки диагноза. Стоимость визита к врачу зависит от категории врача (1-я, 2-я, 3-я) и цели посещения: консультация, обследование, лечение и др. Некоторым пациентам предоставляется скидка на обслуживание. В БД должна храниться информация: о ВРАЧАХ: Ф.И.О. врача, специальность, категория; ПАЦИЕНТАХ: номер

медкарты, Ф.И.О. пациента, дата рождения, адрес, пол, скидка на обслуживание (%); ежедневном ПРИЕМЕ пациентов: номер талона на прием к врачу, дата визита, цель посещения, стоимость визита (руб.); ДИАГНОЗАХ: код диагноза, наименование диагноза. При проектировании БД необходимо учитывать следующее: врач осуществляет по талонам ежедневно несколько приемов. Каждый прием осуществляется одним врачом; пациент может приходить на прием к одному врачу несколько раз. На прием по талону приходит только один пациент; один и тот же диагноз может выставляться нескольким пациентам. На одном приеме выставляется один диагноз. Каждый врач обязательно принимает пациентов, которые взяли талон. Каждый прием обязательно осуществляется врачом; каждый пациент обязательно приходит на прием по талону. На каждый прием обязательно приходит пациент; возможный диагноз не обязательно выставляется на приеме (его может не быть у принятых врачом пациентов).

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?

Практическая работа №2 ПОСТРОЕНИЕ СХЕМЫ И СЛОВАРЯ БАЗЫ ДАННЫХ

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является получение практических навыков разработки схемы базы данных.

Задачами работы, обеспечивающими выполнение цели, является изучение принципов и получение практических навыков разработки схемы базы данных; выявления отношений в заданной предметной области; определение атрибутов отношений; выявление связей отношений; отображение связей отношений на диаграмме базы данных.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Схема базы данных (от англ. Database schema) – ее 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 между ними.

Для выявления сущностей предметной области необходимо ее проанализировать и выявить объекты и их свойства. 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 определяться 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.

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, обучающийся в учебном заведении. Характеризуется фамилией, именем, отчеством (отдельные атрибуты типа строка); номером зачетной книжки (атрибут целого типа). Информация о студентах хранится в отдельном отношении (таблице). Учебное заведение – это тоже отдельное отношение, так как оно имеет свои характеристики, например, название, адрес и т.д.

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, атрибут «фамилия студента» состоит из символов, это обязательный атрибут; атрибут «номер зачетной книжки» – число в диапазоне от 10000 до 99999.

В бизнес-правилах, характеризующих связи между сущностями, должна быть указана следующая информация:

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

Например, отношение «Покупка» связано с отношением «Товар», так как покупка должна всегда содержать товар. Данная связь имеет множественность «многие ко многим», так как одна покупка может содержать много товаров, а один товар может присутствовать во множестве покупок. Каждое выявленное бизнес-правило реализуются в виде фрагмента ER диаграммы.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Для информационной системы «Поликлиника»:

1. Выделить отношения предметной области 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 MS SQL Server Management Studio таблицы в соответствии с заданием.
4. Построить диаграмму отношений в среде MS SQL Server Management Studio.
5. Произвести заполнение отношений тестовыми данными.

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?

Практическая работа №3 ИЗУЧЕНИЕ КОМАНД АДМИНИСТРИРОВАНИЯ ДАННЫХ ДЛЯ СРЕДЫ ОДНОЙ ИЗ СУБД

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Получение практических навыков администрирования в среде MS SQL SERVER.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Управление разрешениями в Компонент Database Engine осуществляется на уровне сервера с помощью имен входа и ролей сервера и на уровне базы данных с помощью пользователей и ролей базы данных.

Security principal (Субъект безопасности) – это официальное название пользователей или групп пользователей, которые используют SQL Server и которым можно назначать разрешения для выполнения действий.

Создавать субъекты безопасности и управлять ими можно с помощью Transact-SQL или визуального интерфейса SQL Server Management Studio.

Уровень сервера

Имена входа – это учетные записи отдельных пользователей для входа в Компонент SQL Server Database Engine. SQL Server и База данных SQL поддерживают имена входа на основе проверки подлинности Windows и на основе проверки подлинности SQL Server.

Создать имя входа можно с помощью инструкции T-SQL CREATE LOGIN. Например:

```
CREATE LOGIN [TestLogin]
WITH PASSWORD=N'Pa$$w0rd',
DEFAULT_DATABASE=[Test],
DEFAULT_LANGUAGE=[русский],
CHECK_EXPIRATION=OFF,
CHECK_POLICY=ON
GO
```

Предопределенные (фиксированные) роли сервера (таблица 2) – это набор предварительно настроенных ролей, который представляет собой удобную группу разрешений на уровне сервера. Имена входа можно добавить в роли, используя инструкцию ALTER SERVER ROLE ... ADD MEMBER.

Таблица 2 – Предопределенные роли уровня сервера

Имя	Описание
sysadmin	Могут выполнять любые действия на сервере.
serveradmin	Могут изменять параметры конфигурации на уровне сервера, а также выключать сервер.

securityadmin	Управляют именами входа и их свойствами (GRANT, DENY и REVOKE) на уровне сервера или базы данных. Могут сбрасывать пароли для имен входа SQL Server.
processadmin	Могут завершать процессы, работающие на экземпляре SQL Server.
setupadmin	Могут добавлять или удалять связанные серверы с помощью инструкций Transact-SQL.
bulkadmin	Могут выполнять инструкцию BULK INSERT.
diskadmin	Используется для управления файлами на диске.
dbcreator	Могут создавать, изменять, удалять и восстанавливать любые базы данных.
public	Каждое имя для входа SQL Server принадлежит к роли сервера public . Если для участника на уровне сервера не были предоставлены или запрещены конкретные разрешения на защищаемый объект, он наследует разрешения роли public на этот объект. Разрешения роли public следует назначать только тому объекту, который будет доступен всем пользователям. Нельзя изменить членство в роли public.

Фиксированную роль сервера нельзя удалить или модифицировать. Нельзя также создать новую фиксированную роль. Предоставить права доступа к серверу можно только путем включения пользователя в требуемую роль сервера. Таким образом, роли сервера позволяют объединять пользователей, выполняющих одинаковые функции, для упрощения администрирования системы безопасности SQL Server.

Начиная с SQL Server 2012 можно создавать собственные роли сервера и назначать им разрешения на уровне сервера. Для создания и удаления этих ролей используются инструкции языка Transact-SQL CREATE SERVER ROLE и DROP SERVER ROLE соответственно. Имена входа можно добавить в роли сервера, используя инструкцию ALTER SERVER ROLE ... ADD MEMBER.

Пример:

```
USE TestDb;
```

```
GO
```

```
CREATE SERVER ROLE program_admin;
```

```
ALTER SERVER ROLE program_admin ADD MEMBER Anna;
```

Уровень базы данных

Именам входа доступ к базе данных предоставляется путем создания

пользователя базы данных в базе данных и сопоставления этого пользователя базы данных с именем входа.

Как правило, имя пользователя базы данных совпадает с именем входа, хотя это и необязательно. Один пользователь базы данных сопоставляется с одним именем входа. Имя входа может быть сопоставлено только с одним пользователем в базе данных, однако может сопоставляться как пользователь базы данных в нескольких базах данных.

Кроме того, можно создать пользователей базы данных без соответствующих имен входа. Они называются пользователями автономной базы данных. Microsoft рекомендуют использовать пользователей автономной базы данных, поскольку это упрощает перенос базы данных на другой сервер.

Пользователи создаются с помощью конструкции CREATE USER.

Например:

```
USE Test
GO
CREATE USER [TestLogin] FOR LOGIN [TestLogin]
WITH DEFAULT_SCHEMA=[dbo]
GO
```

Предопределенные роли базы данных – это набор предварительно настроенных ролей, который представляет собой удобную группу разрешений на уровне базы данных.

При создании базы данных сервер автоматически создает для нее фиксированные роли, которые, как и фиксированные роли сервера, нельзя удалить или модифицировать:

Db_owner – для выполнения любых действий в базе данных;

Db_accessadmin – для добавления и удаления пользователей;

Db_securityadmin – для управления всеми разрешениями, объектами, ролями и именами ролей;

Db_ddladmin – для выполнения любых команд DDL, кроме GRANT, DENY и REVOKE;

Db_backupoperator – для выполнения команд DBCC, CHECK, POINT и BACKUP;

Db_datareader – для контроля данных во всех таблицах базы данных и их чтения;

Db_datawriter – для модификации данных в любых таблицах базы данных;

Db_denydatareader – для запрета просмотра данных в любой таблице базы данных;

Db_denydatawriter – для запрета модификации данных во всех таблицах базы данных.

Public – автоматически становятся все пользователи, имеющие тот или иной доступ к базе данных. Эта роль имеет специальное назначение и обеспечивает минимальные права доступа к базе данных тем пользователям,

для которых их права не определены явно.

Обычно определяемые пользователем роли базы данных применяются, когда группе пользователей базы данных требуется выполнять общий набор действий в базе данных и отсутствует подходящая группа пользователей Windows. Для создания, изменения и удаления этих ролей применяется или среда Management Studio, или инструкции языка Transact-SQL CREATE ROLE, ALTER ROLE и DROP ROLE.

Пользовательские роли и роли приложения создают индивидуально для групп пользователей и групп приложений, наделяя их необходимыми правами доступа к конкретной базе данных.

В любую роль базы данных можно включать:

- а) пользователей сервера;
- б) роли сервера;
- в) пользователей Windows;
- г) группы пользователей Windows.

Для создания новой определяемой пользователем роли базы данных в текущей базе данных применяется инструкция CREATE ROLE.

CREATE ROLE role_name [AUTHORIZATION owner_name]

Где role_name - имя создаваемой роли,

owner_name - пользователь базы данных или роль, которая будет владельцем новой роли.

Для изменения имени определяемой пользователем роли базы данных применяется инструкция ALTER ROLE, а для удаления роли из базы данных - инструкция DROP ROLE.

Пример: Создание определяемой пользователем роли базы данных и добавление в нее членов:

USE SampleDb;

CREATE ROLE marketing AUTHORIZATION Vasya; /* создается определяемая пользователем роль базы данных marketing*/

GO

ALTER ROLE marketing ADD MEMBER 'Vasya ';

ALTER ROLE marketing ADD MEMBER 'Anna';

/* добавляются два члена - Vasya и Anna*/

Модель безопасности компонента Database Engine

Выполнять инструкции или осуществлять операции с объектами баз данных могут только авторизованные пользователи. Попытка выполнения любой из этих задач неавторизованным пользователем будет неудачной. Для выполнения задач, связанных с авторизацией, используются следующие три инструкции языка Transact-SQL: GRANT, DENY и REVOKE.

Модель разделяет мир сервера базы данных на принципалов безопасности и защищаемые объекты. Каждый защищаемый объект имеет связанные с ним разрешения, которые могут быть предоставлены принципалу.

Принципалы (отдельные лица, группы или приложения) могут обращаться к защищаемым объектам.

Защищаемые объекты - это ресурсы, доступ к которым регулируется подсистемой авторизации.

Существует три основных класса защищаемых объектов: сервер, база данных, схема. Они содержат другие защищаемые объекты, такие как регистрационные имена, пользователи базы данных, таблицы и хранимые процедуры.

С каждым защищаемым объектом в SQL Server связаны разрешения, которые могут быть предоставлены участнику. Управление разрешениями в Компонент Database Engine осуществляется на уровне сервера

- назначение разрешений именам входа
 - назначение разрешений ролям сервера
- на уровне базы данных
- назначение разрешений пользователям
 - назначение разрешений ролям базы данных.

Объект, предоставляющий разрешение, должен иметь либо само разрешение, выданное с помощью параметра GRANT OPTION, либо разрешение более высокого уровня, которое неявно включает предоставляемое.

Владельцы объектов могут предоставлять разрешения на объекты, которыми они владеют.

Предоставление разрешений на защищаемый объект участнику:

```
GRANT { ALL [ PRIVILEGES ] }
    | permission [ ( column [ ,...n ] ) ] [ ,...n ]
    [ ON [ class :: ] securable ] TO principal [ ,...n ]
    [ WITH GRANT OPTION ] [ AS principal ]
```

Пример. Пользователю Anna предоставляется разрешение CREATE TABLE для базы данных TestDB:

```
USE TestDB;
GRANT CREATE TABLE TO Anna;
GO
```

Удаление разрешений, выданных или запрещенных ранее:

```
REVOKE [ GRANT OPTION FOR ]
    { [ ALL [ PRIVILEGES ] ] | permission [ ( column [ ,...n ] ) ] [ ,...n ] }
    [ ON [ class :: ] securable ]
    { TO | FROM } principal [ ,...n ]
    [ CASCADE ] [ AS principal ]
```

Пример. У пользователя Anna отзывается разрешение CREATE VIEW для базы данных TestDB:

```
USE TestDB;
REVOKE CREATE VIEW FROM Anna;
GO
```


ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создать роли в соответствии с заданием (таблица 3).
2. Предоставить ролям разрешения.
3. Создать пользователей БД в соответствии с заданием.
4. Включить созданных пользователей БД в указанную роль.
5. Выполнить проверку выполнения разрешений пользователей БД.

Таблица 3

Роли	Объекты администрирования	Пользователь
Роль1	Чтение/запись/удаление информации из любой таблицы базы данных	Администратор
Роль2	Чтение/запись/удаление информации из таблиц «Прием» и «Талон» базы данных	Врач
Роль3	Чтение информации из таблиц «Прием» и «Талон» базы данных	Пациент

КОНТРОЛЬНЫЕ ВОПРОСЫ

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

Лабораторная работа №1

РАЗРАБОТКА ТРЕБОВАНИЙ И КОНФИГУРИРОВАНИЕ КОРПОРАТИВНОЙ СЕТИ

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Получить практический навык разработки требований к серверу баз данных и конфигурирования корпоративной сети.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

В контексте сетевых технологий, конфигурация означает установку, настройку и поддержку аппаратного и программного обеспечения, которое обеспечивает функционирование корпоративной сети.

Конфигурирование корпоративной сети, особенно в случае, когда в ней присутствует сервер баз данных, требует тщательного планирования и выполнения нескольких шагов.

1. Анализ требований: важно сначала определить требования компании к сети и серверу баз данных. Необходимо определить, сколько пользователей будет использовать сеть, какие приложения будут запущены на сервере баз данных, сколько данных будет храниться, какие ограничения по безопасности необходимо установить и т.д.

2. Выбор архитектуры: на основе требований, необходимо выбрать подходящую архитектуру для корпоративной сети. При этом рассматриваются возможности использования физических и виртуальных серверов, распределение нагрузки, резервное копирование данных и другие важные факторы.

3. Приобретение оборудования: определяется, какое оборудование понадобится для создания сети и возможности его приобретения. В случае использования сервера баз данных, может потребоваться мощный сервер, сетевые коммутаторы, маршрутизаторы, сетевые кабели и другое оборудование.

4. Установка и настройка сервера баз данных, которая включает в себя настройку доступа к базам данных, установку прав доступа, задание резервного копирования и восстановления данных, настройку масштабирования и другие параметры.

5. Настройка сетевого оборудования (коммутаторов и маршрутизаторов), установку IP-адресов, настройку VLAN (виртуальных локальных сетей), настройку сетевой безопасности и т.д.

6. Тестирование и отладка: после завершения настройки сети и сервера баз данных, необходимо провести тестирование для проверки работоспособности и производительности системы. Здесь важно проверить соединение между клиентами и сервером, выполнение запросов к базе данных, скорость передачи данных и другие параметры.

7. Обслуживание и мониторинг: После развертывания сети и сервера баз данных, важно регулярно проводить обслуживание и мониторинг (проверку

работоспособности оборудования, обновление программного обеспечения, мониторинг производительности сети и базы данных, обеспечение безопасности и т.д.).

Разработка требований к серверу баз данных состоит из нескольких этапов.

1. Изучение требований бизнеса. Важно понять, какую информацию необходимо хранить и обрабатывать, а также какие операции с данными будут выполняться. Разработчики должны внимательно изучить бизнес-процессы и взаимодействие с другими системами для определения потребностей в хранении и обработке данных.

2. Анализ объема данных. Определение объема данных, которые будут храниться на сервере баз данных, поможет выбрать соответствующее оборудование и оптимизировать структуру базы данных. Так же необходимо учитывать прогнозируемый рост объема данных со временем для обеспечения масштабируемости системы.

3. Определение требований к производительности сервера баз данных. Производительность включает в себя ожидаемое время отклика, пропускную способность, количество одновременных подключений и другие параметры. Здесь важно прогнозировать нагрузку на систему, чтобы выбрать подходящее оборудование и оптимизировать настройки базы данных.

4. Выбор типа сервера баз данных. В зависимости от требований бизнеса и объема данных, необходимо выбрать подходящий тип сервера баз данных. Это может быть реляционная база данных, NoSQL база данных или другие альтернативные СУБД. Каждый тип базы данных имеет свои особенности и преимущества, и выбор зависит от конкретных требований проекта.

5. Управление безопасностью данных. Безопасность данных является критически важным аспектом разработки требований к серверу баз данных. Необходимо определить требования к аутентификации, авторизации, шифрованию и аудиту данных. Защита данных от несанкционированного доступа и утечек информации должна быть приоритетом при разработке требований к серверу баз данных.

6. Резервное копирование и восстановление данных. Требования к серверу баз данных должны включать политику резервного копирования и восстановления данных. Необходимо определить периодичность резервного копирования, сохранение и восстановление данных.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Провести анализ серверов баз данных: SQL Server, ORACLE и MySQL. Подготовить отчет. В отчет включить описание реализации требований к серверам баз данных для каждого из рассматриваемых производителей.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите основные требования к серверу баз данных
2. Опишите основные этапы конфигурирования корпоративной сети.

Лабораторная работа №2 РАЗРАБОТКА МЕХАНИЗМОВ СЕРВЕРА БАЗ ДАННЫХ. ХРАНИМЫЕ ПРОЦЕДУРЫ

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цель работы – получить практический навык организации бизнес-логики на стороне сервера, посредством использования хранимых процедур и пользовательских функций.

Задачами работы являются:

1. Изучение принципов работы и создания хранимых процедур и пользовательских функций в среде MS SQL Server.
2. Создание хранимых процедур и пользовательских функций.

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 существует 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_параметра      тип_данных
[=default] [OUTPUT];
```

[=default] – значение по умолчанию, присваиваемое 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 использования константы в качестве параметра:

EXEC СводСотрудников «Инженер»

Пример использования 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 типДанных

где имяПерем – идентификатор переменной. 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 «@» 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

SELECT имяПерем=значение

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

SELECT имяПерем AS псевдоним.

Основные алгоритмические конструкции процедурного расширения языка SQL

Конструкция ветвления в SQL:

IF логическоеВыражение

{операторSQL | блокОператоровSQL}

[ELSE { операторSQL | блокОператоровSQL }]

Конструкция цикла:

WHILE <логическое_выражение>

операторSQL | блокОператоровSQL

где блокОператоровSQL – это операторные блоки, используемые для группировки операторов в скобки BEGIN ... END, аналогично «{ }» 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.

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 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
```

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»)
```

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.

Удаление функции

```
DROP FUNCTION Имя функции
```

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.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создать хранимую процедуру, в которую в качестве значения параметра передается фамилия врача. Найти все свободные талоны этого врача.
2. Создать хранимую процедуру, которой в качестве значения параметра передается месяц N. Для каждого врача рассчитать сумму услуг, оказанных в месяце N текущего года. Информацию записать в итоговую таблицу. Если врач в указанном месяце услуг не оказывал, то записать NULL.
3. Создать хранимую процедуру, которой в качестве значения параметра передается фамилия врача, дата, время начала работы, количество талонов, которые необходимо сгенерировать. Для заданного врача добавить указанное количество талонов. Время каждого приема – 15мин для врачей 1 категории, 10 – для врачей остальных категорий.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое хранимая процедура, общий формат описания и вызов?
2. Каков формат описания переменных в хранимых процедурах?
3. Типы 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?

Лабораторная работа №3 **РАЗРАБОТКА МЕХАНИЗМОВ СЕРВЕРА БАЗ ДАННЫХ. ТРИГГЕРЫ**

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цель работы – получить практический навык использования триггеров для организации бизнес-логики на стороне сервера.

Задачи работы: изучить принципы работы и особенности построения триггеров в среде MS SQL Server, создать триггеры, выполняющие действия, согласно заданию.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Триггер – это откомпилированная процедура, используемая для выполнения действий, инициируемых происходящими в базе данных событиями. Такими событиями являются запросы к базеданным, генерируемые операторами языка манипуляции данными – DML (INSERT, DELETE или UPDATE). Триггер может выполняться вместо или после INSERT, DELETE или UPDATE. С помощью триггеров можно отменять транзакции, модифицировать данные таблиц, проверять вводимые данные на соответствие нескольким критериям.

Использование триггеров 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 имяТаблицы|Представления
{{FOR|AFTER|INSTEAD OF} {[INSERT], [DELETE], [UPDATE]}}
AS
операторыSQL
```

Изменение 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, при модификации данных данные помещаются и в таблицу deleted и в таблицу inserted.

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. Представленный 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 таблицы Склад
```

```
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.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создать триггер DML, проверяющий наличие диагноза при добавлении записи в таблицу Прием. При ошибке должно выводиться сообщение «Введен неверный номер диагноза». При создании триггера использовать конструкцию INSTEAD OF, иначе сработают ограничения таблиц.
2. Создать триггер DML: Зарплата врачей должна попадать в один из интервалов значений зарплаты, которые установлены для разных категорий сотрудников. Например, для 1 категории – не менее 30000, для 2 категории – не менее 20000, для 3-й категории – не менее 15000. Примечание: необходимо наличие таблицы Зарплата. Примерный перечень полей таблицы Зарплата: Код(РК), НомерВрача(FK), ДатаНачисления, Сумма. При неверном вводе пользователю должно выводиться сообщение об ошибке.
3. Создать триггер DML: При операциях вставки и обновления
 - А) присваивать полю значение по умолчанию в зависимости от значения других полей.
 - Б) переводить первую букву фамилии в верхний регистр.
4. Создать триггер, который будет разрешать изменение некоторого столбца таблицы всем, кроме владельца dbo.
5. Создать таблицу Аудит (Код, пользователь, время, операция), в которую записывать время доступа к конкретной таблице (на выбор) конкретного пользователя, название производимой им операции – U,D или I).
6. Создать таблицу Аудит_XML(Код, имяТаблицы, операция, стараяЗапись, новаяЗапись), в которую в формате XML записывать все изменения, производимые в таблице с помощью операции Update.

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?

Практическая работа №4

КОПИРОВАНИЕ БАЗ ДАННЫХ, ИМПОРТ ЭКСПОРТ ДАННЫХ В СРЕДЕ MS SQL SERVER EXPRESS СРЕДСТВАМИ MANAGEMENT STUDIO

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Получение практических навыков выполнения операций импорта и экспорта данных в среде MS SQL SERVER.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Копирование баз данных

Под копированием подразумевается перенос базы данных с одного сервера на другой. Данные баз данных в среде MS SQL SERVER располагаются в отдельных файлах, одноименных базе данных. Но они не могут быть скопированы как обычные файлы в файловой системе, так как их защищает от копирования служба сервера.

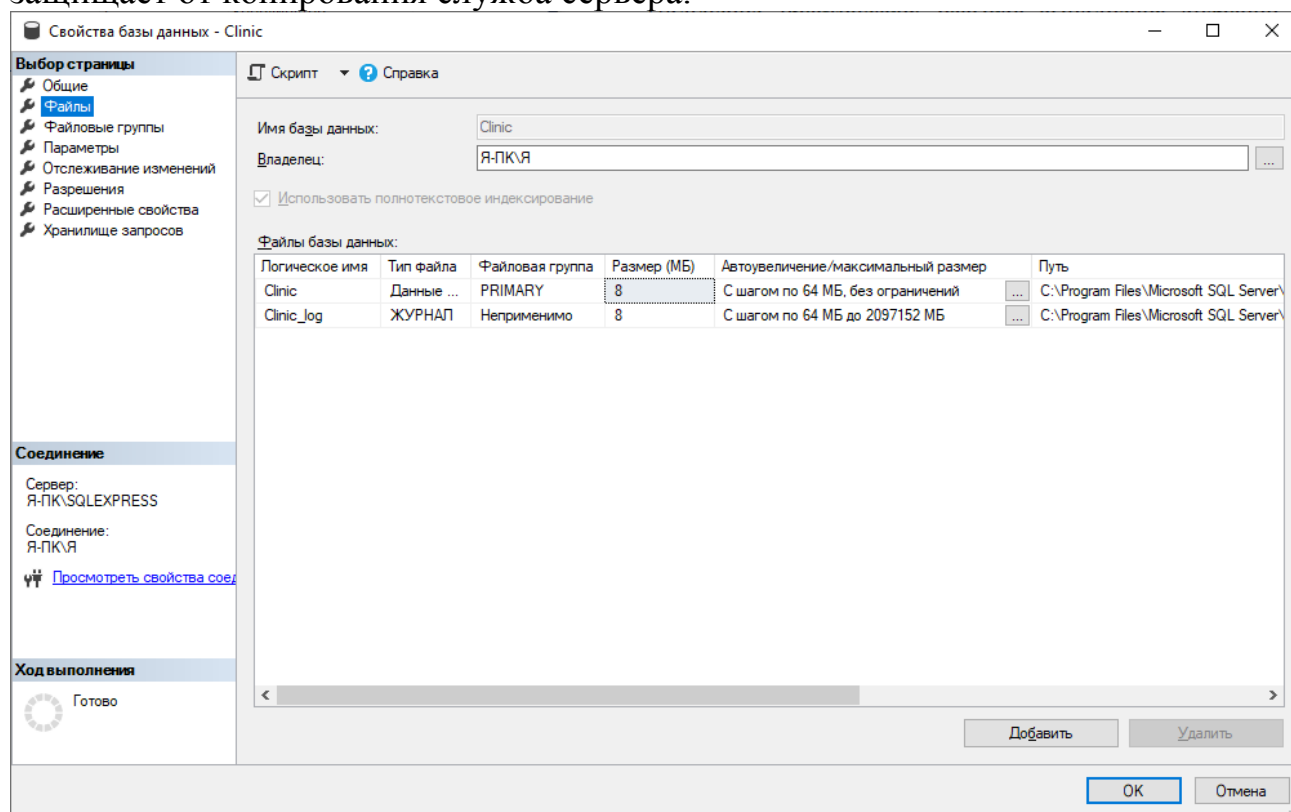


Рисунок 9 – Файлы базы данных Clinic

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

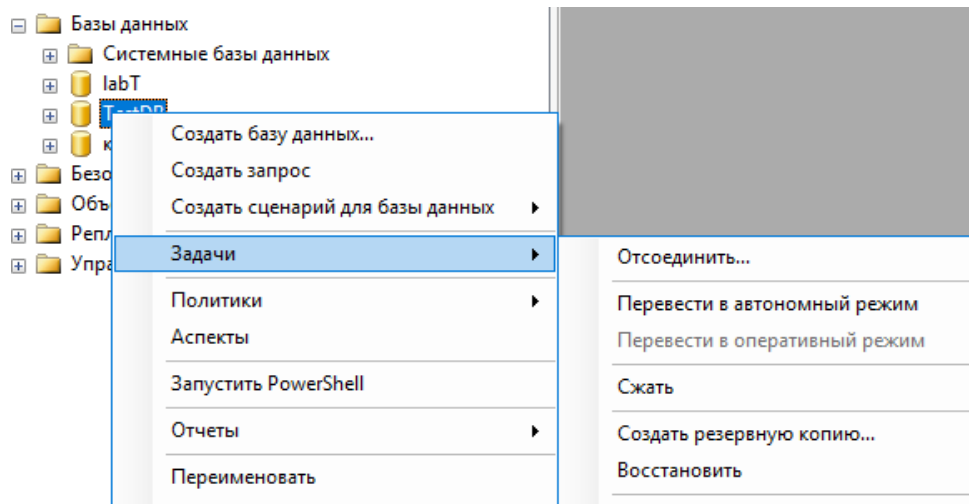


Рисунок 10 – Задачи работы с базой данных, вызываемых из контекстного меню базы данных

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:

- 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» (определяют место, в которое будет производиться копирование).
- В параметрах назначения необходимо выбрать файл, в котором будет создаваться резервная копия. По умолчанию этот файл одноименный 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 резервная копия базы будет создаваться в двух файла.
- При создании нового файла резервной копии можно указать любой путь для расположения резервной копии и любое имя файла.

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

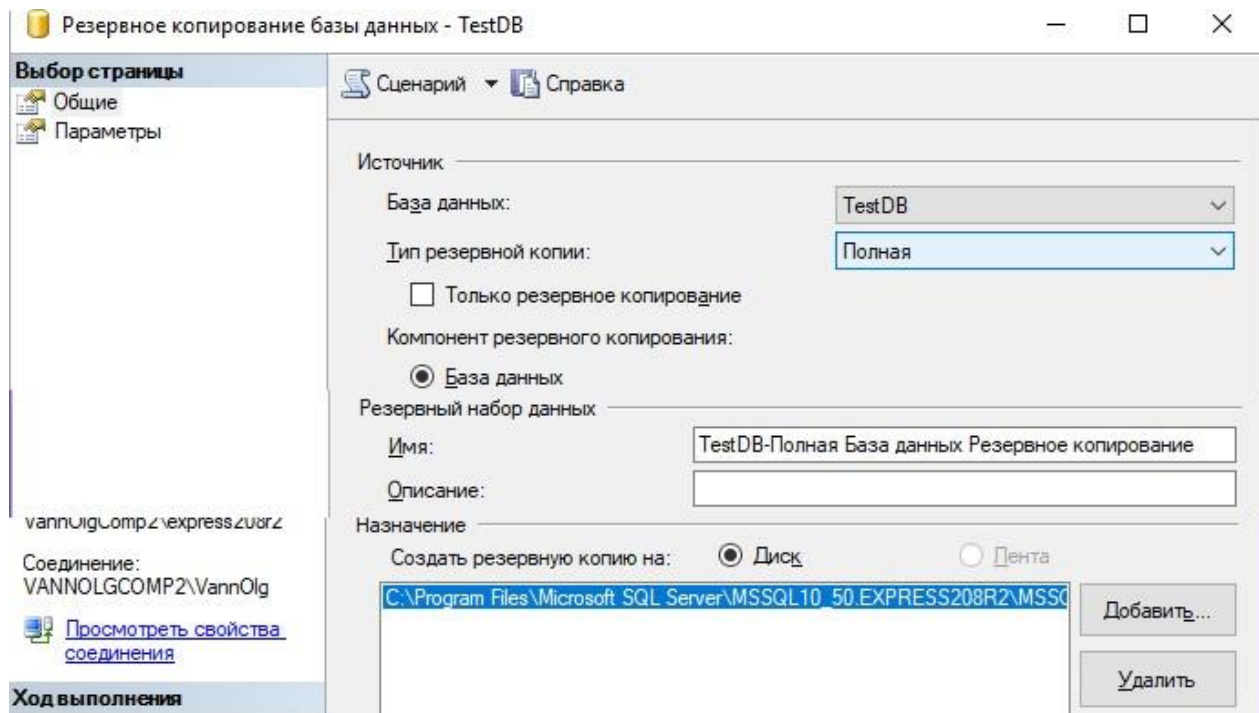


Рисунок 11 – Диалоговое окно создания резервной копии базы данных

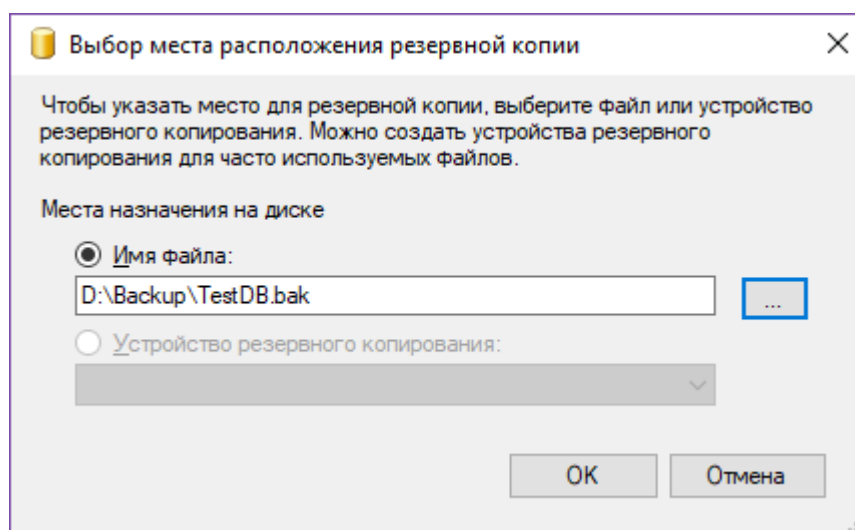


Рисунок 12 – Диалоговое окно добавления нового файла резервной копии

Восстановление базы данных

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

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.

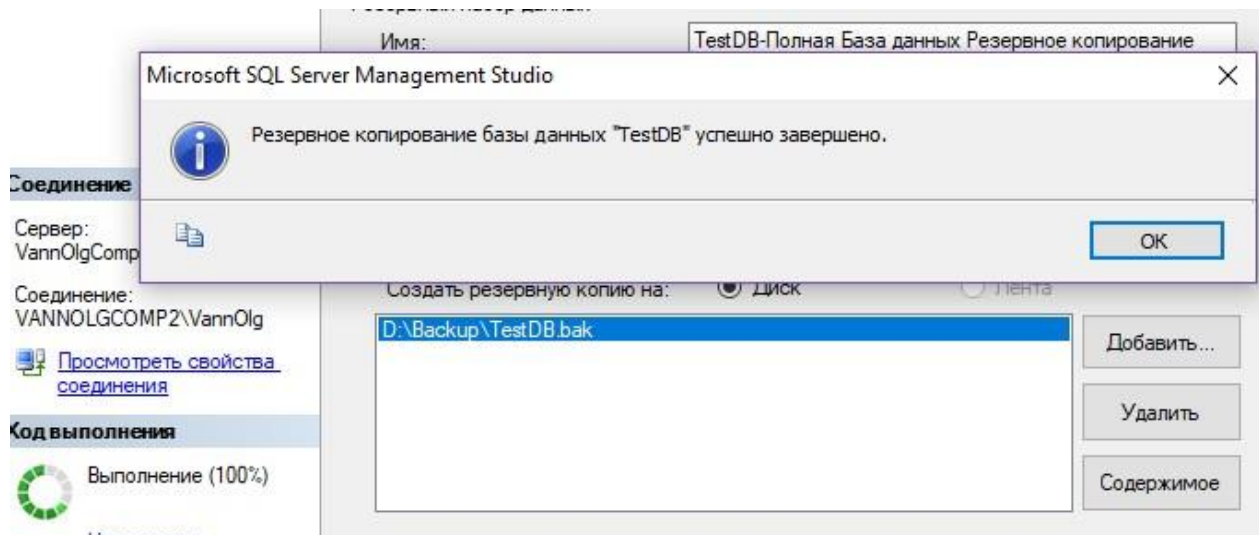


Рисунок 13 – Успешное завершение создания резервной копии

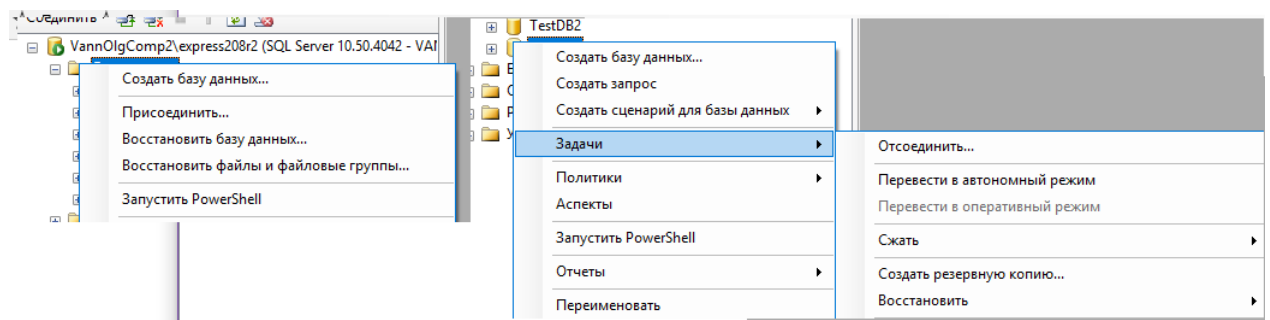


Рисунок 14 – Восстановление базы данных из файла .bak в новую базу или в существующую

Импорт и экспорт данных

Для импорта и экспорта данных в SQL Server доступны разнообразные методы. Сюда входят инструкции Transact-SQL, программы командной строки и мастера. Кроме того, можно импортировать и экспортировать данные в разных форматах. Эти форматы включают неструктурированные файлы, файлы Excel, основные типы реляционных баз данных и форматы различных облачных служб.

Программа bcp (bcp.exe) представляет собой инструмент командной строки, использующий API-интерфейс программы массового копирования (Bulk Copy Program - BCP). Программа bcp выполняет следующие задачи:

- массовый экспорт данных из таблицы SQL Server в файл данных;
- массовый экспорт данных из запроса;
- массовый импорт данных из файла данных в таблицу SQL Server ;
- создание файлов форматирования.

Синтаксис программы bcp:

bcp [database_name.] schema.{table_name | view_name | "query"}


```

{in data_file | out data_file | queryout data_file | format nul}
[-a packet_size]
[-b batch_size]
[-c]
[-C { ACP | OEM | RAW | code_page } ]
[-d database_name]
...
[-T]
[-U login_id]
[-v]
[-V (80 | 90 | 100 | 110 | 120 | 130 ) ]
[-w]
[-x]

```

Мастер импорта неструктурированных файлов позволяет легко скопировать данные из неструктурированного файла (CSV-файл, TXT-файл) в новую таблицу в базе данных. Мастер импорта неструктурированных файлов поддерживает файлы форматирования с разделителями-запятыми и с фиксированной шириной. Мастер создан на основе интеллектуальной платформы Program Synthesis using Examples (PROSE). PROSE анализирует шаблоны данных во входном файле и определяет имена столбцов, типы, разделители и т. д. Платформа запоминает структуру файла и выполняет все действия по обработке данных.

Мастер импорта и экспорта SQL Server предоставляет простой способ копирования данных из источника в целевой объект. Мастер использует службы SQL Server Integration Services (SSIS) для копирования данных. Этапы работы мастера представлены на рисунке 15.

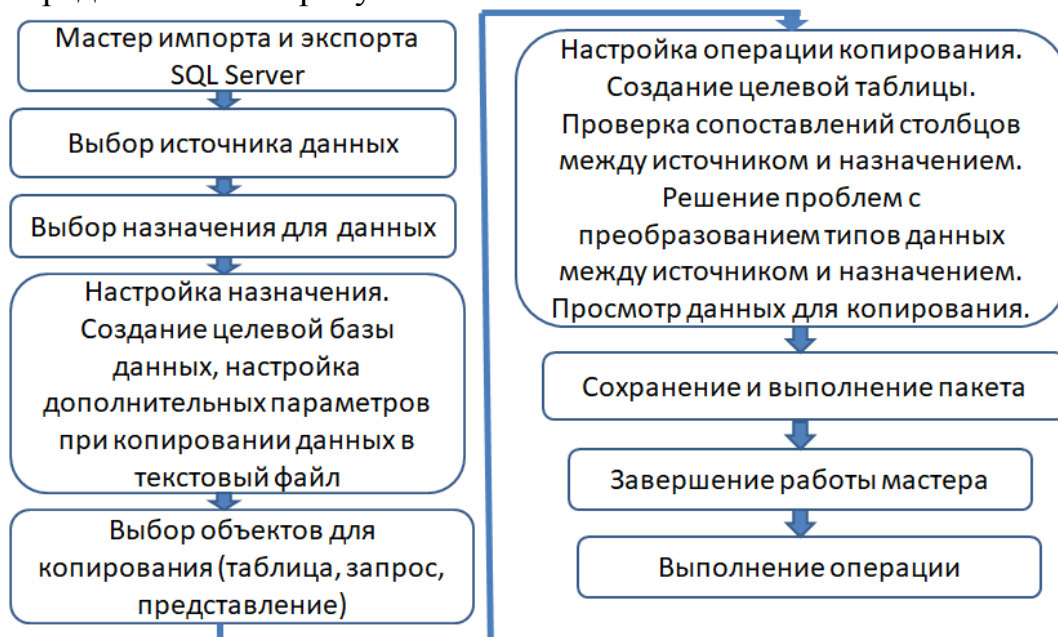


Рисунок 15 – Этапы работы мастера

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. В файле Excel создайте таблицу из 6 столбцов. В каждый столбец занесите однотипные данные.
2. Осуществите импорт данных из файла Excel с помощью мастера импорта и экспорта SQL Server.
3. Осуществите экспорт данных из запроса в таблицу Excel.
4. Создайте текстовый файл с разделителями. Занесите в него данные (10 строк, 5 столбцов). Осуществите импорт данных с помощью импорта неструктурированных файлов в SQL.
5. Экспортируйте данные из любой таблицы в текстовый файл.
6. Скопируйте данные из одной таблицы в другую.

КОНТРОЛЬНЫЕ ВОПРОСЫ

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

Практическая работа №5 КОПИРОВАНИЕ БАЗ ДАННЫХ СРЕДСТВАМИ КОМАНД SQL

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

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

Задачами работы является изучение команд DDL, предназначенных для создания объектов БД, команд модификации данных, обеспечивающих создание копии базы данных.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Создание копии БД с использованием средств администрирования, рассмотренное в прошлой работе, позволяет создавать копии 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 невозможно осуществить с помощью средств создания и восстановления резервной копии. Для копирования данных на более позднюю версию сервера или на другой сервер могут быть использованы команды подязыков SQL DDL и DML. В данном случае копирование будет заключаться в создании таблиц и других объектов данных идентичных исходной базе и заполнения данных в таблицы через команды добавления данных.

Для упрощения работы по написанию команд создания таблиц можно воспользоваться предоставляемой средой SSMS возможностью автоматической генерации данных команд через контекстное меню.

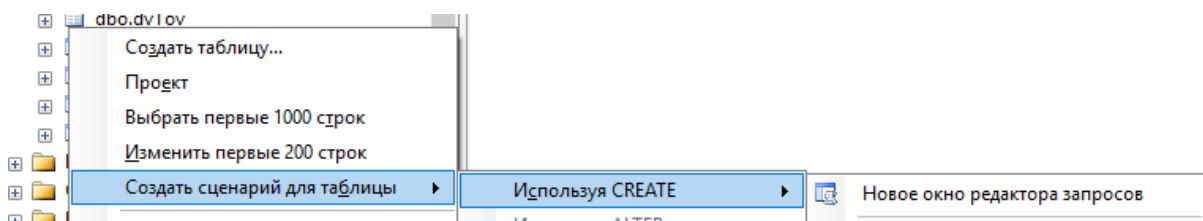


Рисунок 16 – Вызов автоматической генерации команд создания таблиц

Код, который будет сгенерирован при этом, приведен 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

```

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

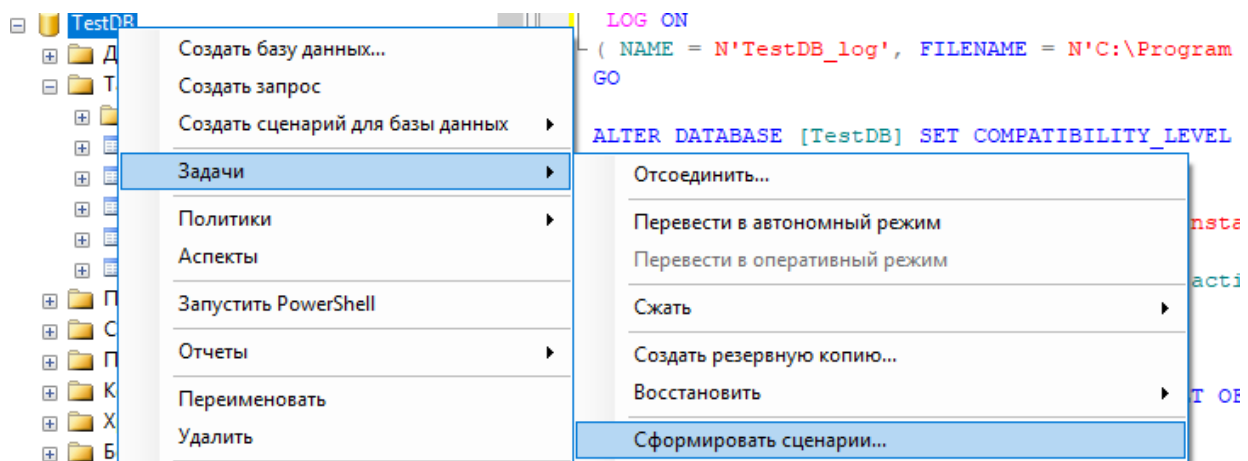


Рисунок 17 – Вызов автоматической генерации сценариев создания объектов базы данных

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

Заполнение таблиц данными производится через команды INSERT. Генерация шаблонов этих команд также может быть выполнена автоматически.

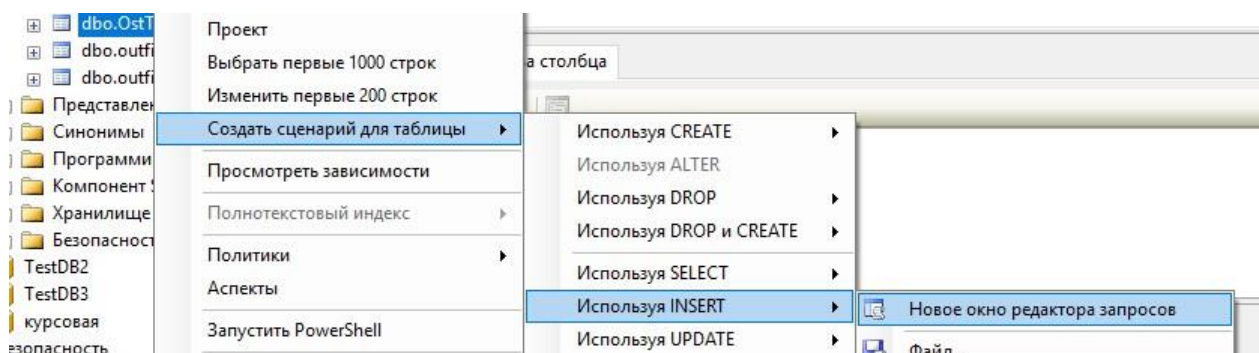


Рисунок 18 – Вызов автоматической генерации команд вставки данных в существующую таблицу

Копирование данных из одной таблицы в другую можно осуществить с

помощью команды SELECT INTO. Инструкция SELECT...INTO создает новую таблицу в файловой группе и вставляет в нее результирующие строки из запроса.

```
SELECT * INTO new_table FROM data_source [ ON filegroup ]
```

Для того чтобы создать таблицу в другой базе данных в этом же экземпляре службы SQL Server, необходимо определить new_table в качестве полного имени в форме database.schema.table_name.

Пример:

```
SELECT * INTO dbo.NewProducts
FROM Production.Product
WHERE ListPrice > $25
AND ListPrice < $100;
GO
```

Копирование данных из одной таблицы в другую также можно осуществить с помощью команды INSERT INTO ... SELECT, которая используется, чтобы добавить импортированные данные в существующую таблицу, а не создавать новую.

Инструкция INSERT INTO <target_table> SELECT <columns> FROM <source_table> может эффективно перенести большое количество строк из одной таблицы (например, промежуточной) в другую таблицу с минимальным протоколированием. Минимальное протоколирование может повысить производительность выполнения инструкции и снизить вероятность того, что во время операции будет заполнен весь журнал транзакций.

Для минимального протоколирования этой инструкции необходимо выполнение следующих требований:

- Модель восстановления базы данных настроена на простое или неполное протоколирование.
- Целевой таблицей является пустая или непустая куча.
- Целевая таблица не используется в репликации.
- Для целевой таблицы используется указание TABLOCK.

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 отдельного 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 отчет по работе, включающий отображение в виде копий экрана этапов выполнения работы.

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?

Практическая работа №6

УСТАНОВКА И НАСТРОЙКА СЕРВЕРА MS SQL SERVER EXPRESS

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

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

Задачами работы является изучение технической документации, установка MS SQL Server Express, настройка MS SQL Server Express.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

MS SQL Server доступен в различных вариациях. Прежде всего, это MS SQL Server Enterprise – полный выпуск, нацеленный на использование в реальных проектах. Именно он используется на различных хостингах и серверах баз данных. Однако он доступен только в платной версии (не считая триального периода).

Для простых приложений также может хватить и выпуска Express: он бесплатный. К тому же у него есть преимущество - его можно ставить в качестве реального сервера и использовать в реальных задачах, однако он имеет урезанный функционал по сравнению с полной версией. 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=101064>

Microsoft® SQL Server® 2019 Express

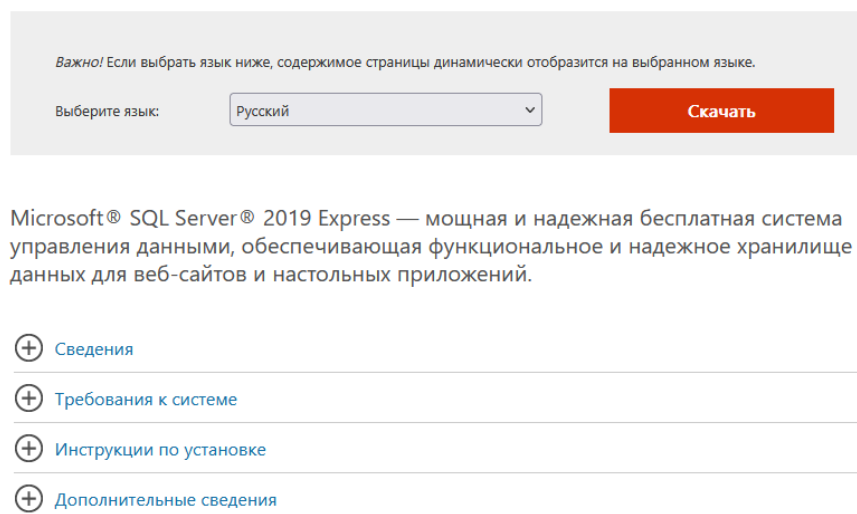


Рисунок 19 – Источник для скачивания версии

Установка выполняется в виде следующих шагов:

1 шаг. Запуск скачанного файла установки

2 шаг: Выбор типа установка (обновление или установка). Здесь выберем первый пункт "Новая установка изолированного экземпляра SQL...". Далее с помощью последовательности шагов необходимо будет установить опции установки.



Рисунок 20 – Мастер установки MS SQL Server 2019

3 шаг: Подтверждение лицензионного соглашения.

4 шаг: Выбор компонентов. На этом этапе предлагается выбрать компоненты для установки. Здесь отметим только компонент «Службы ядра СУБД».

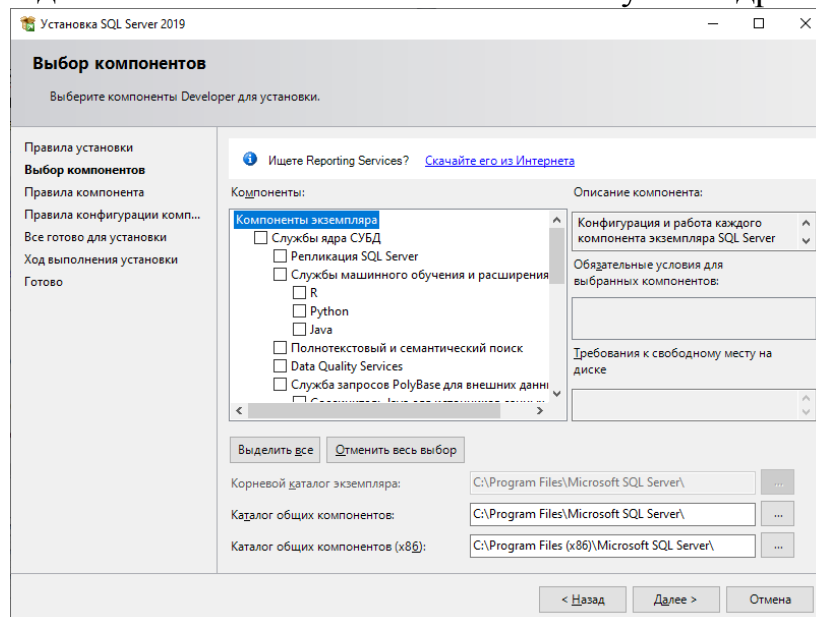


Рисунок 21 – Выбор компонентов

5 шаг: Настройки экземпляра. На этом этапе необходимо указать название и ID запускаемой сущности SQL Server. Для имени указываем опцию «Экземпляр по умолчанию», а для ID устанавливаем MSSQLSERVER. Это будет то имя экземпляра, по которому мы сможем обращаться к серверу из внешних приложений.

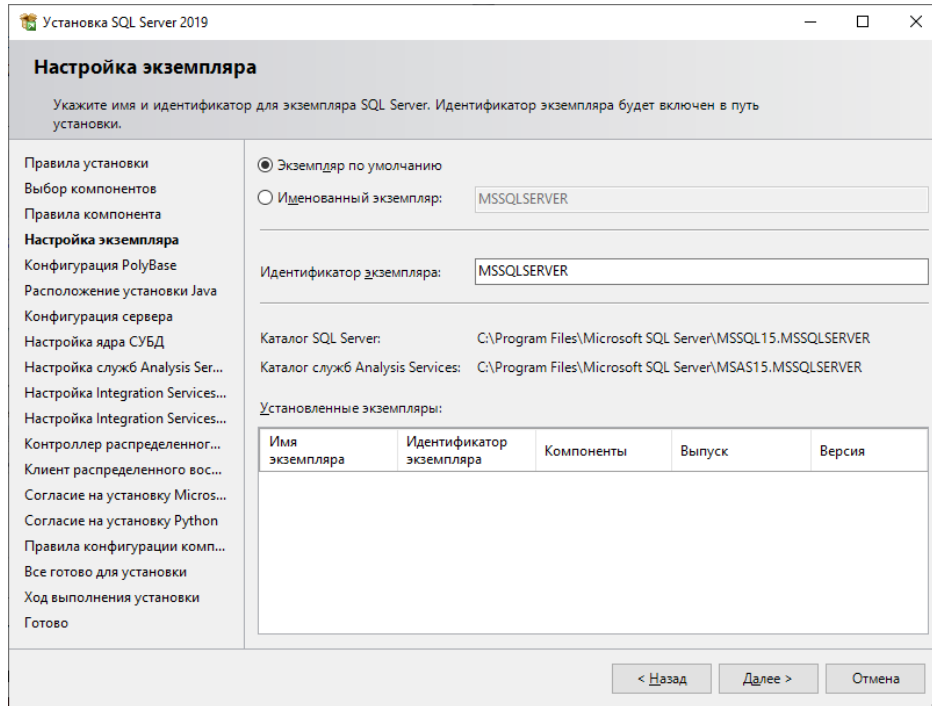


Рисунок 22 – Задание имени и идентификатора MS SQL Server

6 шаг: Настройка ядра СУБД. С помощью кнопки «Добавить текущего пользователя» добавим текущего пользователя в качестве администратора для сервера.

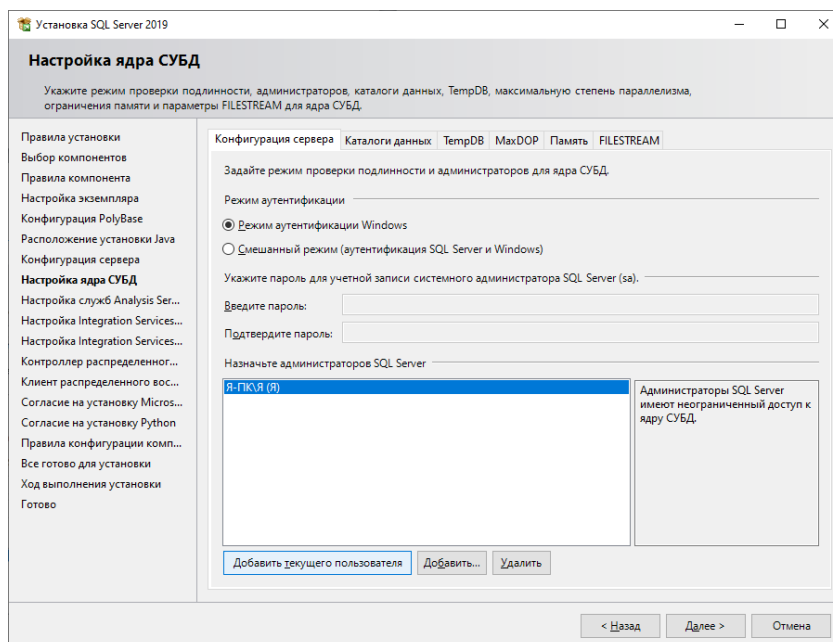


Рисунок 23 – Установка администратора для MS SQL Server 2019

На всех последующих шагах оставим настройки по умолчанию и на самом последнем экране для установки нажмем на кнопку «Установить».

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

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

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие версии MS SQL Server существуют?
2. Как установить экземпляр SQL Server?

Практическая работа №7

УСТАНОВКА И НАСТРОЙКА СЕРВЕРА БД MYSQL

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

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

Задачами работы является изучение технической документации, установка сервера MY SQL, настройка сервера MY SQL.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

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 удаленные 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.

В зависимости от используемой платформы (Windows, Linux, MacOS X) используются различные способы установки сервера MySQL.

MS Windows. Необходимо скачать дистрибутив <https://dev.mysql.com/downloads/mysql/5.5.html#downloads>. Запустить скачанный установщик и следовать инструкциям мастера.

Linux. Необходимо установить пакет сервера MySQL с помощью пакетного менеджера системы (apt, yum и др.).

Mac OS X. Два варианта установки:

а) Необходимо установить пакет сервера MySQL с помощью пакетного менеджера brew, MacPorts.

б) Необходимо скачать дистрибутив. Запустить скачанный установщик и следовать инструкциям мастера.

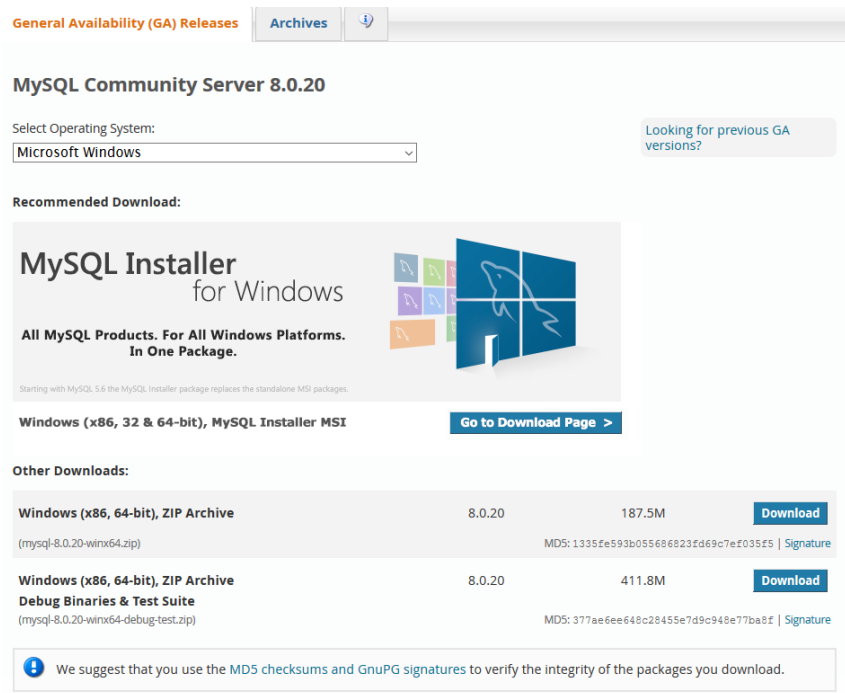


Рисунок 24 – Установщик MySQL

Затем необходимо установить среду MySQL Workbench. Для этого необходимо установить скачанный <https://dev.mysql.com/downloads/workbench/> дистрибутив среды. После скачивания необходимо установить среду с помощью мастера установки или пакетного менеджера.

Установка с помощью MySQL Community Downloads (<https://dev.mysql.com/downloads/file/?id=474803>) начинается со скачивания файла mysql-installer-community-5.7.21.0.msi и запуска установщика.

Затем выбирается вариант установки из предложенных (рекомендуется Developer Default или Custom).

После выбора устанавливаемых компонентов происходит их установка.

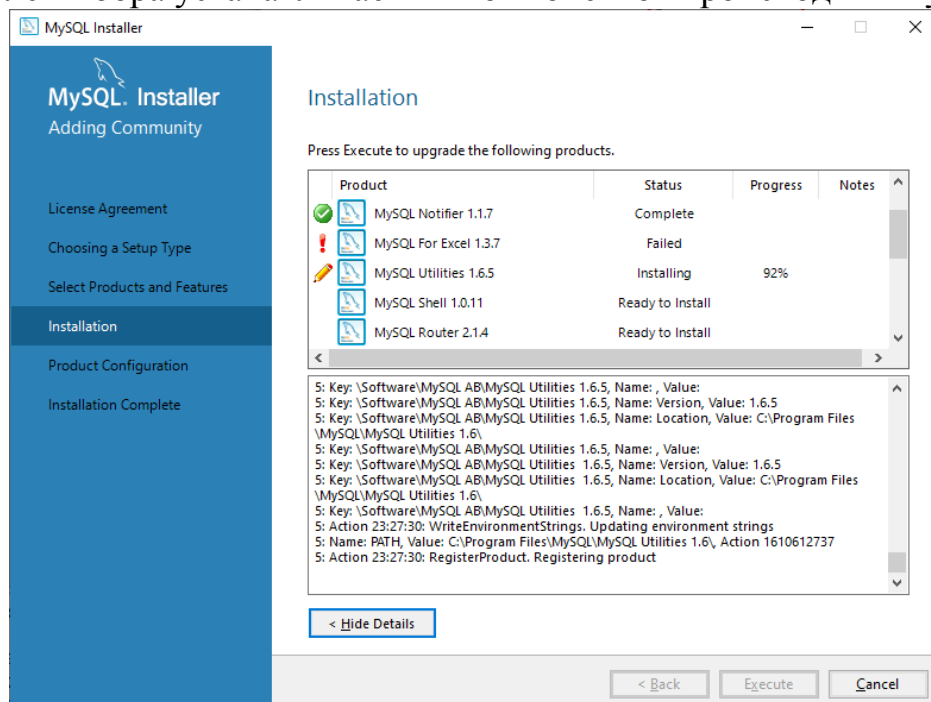


Рисунок 25 – Установка компонентов

Далее необходимо настроить сервер MySQL. На странице «Настройка» нажмите кнопку «Далее».

На первой странице конфигурации сервера MySQL (1/3) установите следующие параметры: тип конфигурации сервера, выберите вариант **«Компьютер для разработки»**, включите поддержку сети TCP/IP. Задайте 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. Затем выберите флажок **«Показать расширенные параметры»** для отображения дополнительной страницы конфигурации для настройки расширенных параметров для экземпляра сервера (если требуется).

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.

На странице учетных записей пользователя MySQL нажмите кнопку «Добавить пользователя» для создания учетной записи пользователя. В диалоговом окне «Сведения о пользователе MySQL» введите имя пользователя, затем выберите роль базы данных и пароль (например, !phpruser). Нажмите кнопку «ОК».

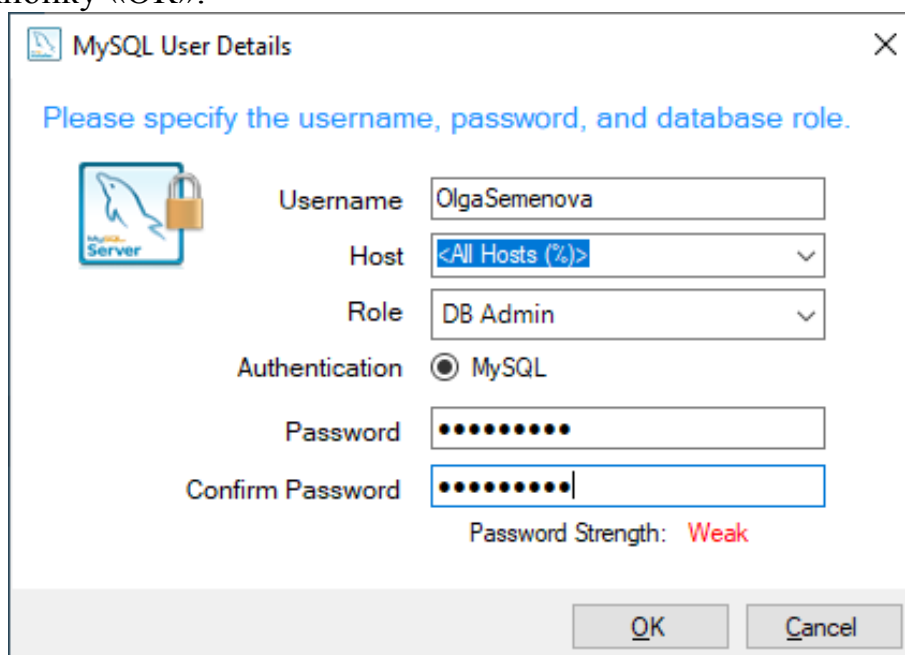


Рисунок 26 – Диалоговое окно «Сведения о пользователе MySQL»

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.

После успешного завершения настройки на панели «Завершение» появляется информационное сообщение. Нажмите кнопку «Завершить». Для проверки успешности настройки запустите диспетчер задач. Если MySQLd-nt.exe присутствует в списке «Процессы», сервер базы данных запущен.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Установить и настроить экземпляр сервера MySQL на домашнем компьютере. Подготовить отчет по выполненной работе.

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. Что такое пользователь «Root»?

Практическая работа №8 ПЕРЕНОС БАЗЫ ДАННЫХ НА ДРУГОЙ ТИП СЕРВЕРА

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цель работы – получить практические навыки переноса данных между серверами баз данных различного типа.

Задачами работы является изучение технической документации, перенос базы данных с сервера MS SQL Server на сервер MySQL.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Перенос данных между различными типами серверов может быть сделан административными средствами. 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 быть большое количество). Для автоматизации этого процесса можно выполнить выгрузку данных из исходной базы данных в некоторый файл, из которого возможна загрузка данных в целевую базу 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».

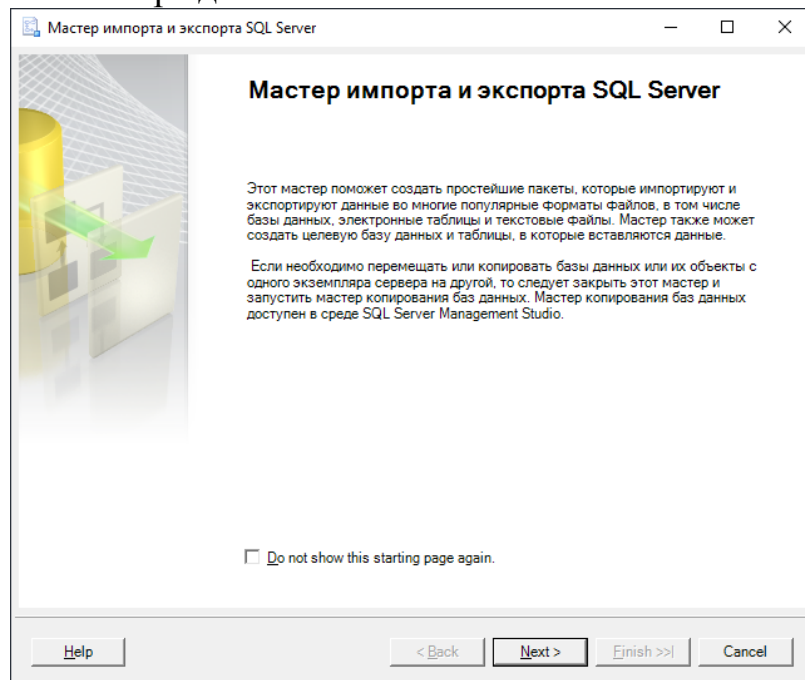


Рисунок 27 – Окно импорта экспорта данных

Далее необходимо указать источник данных, имя сервера, проверку подлинности, текущую базу данных.

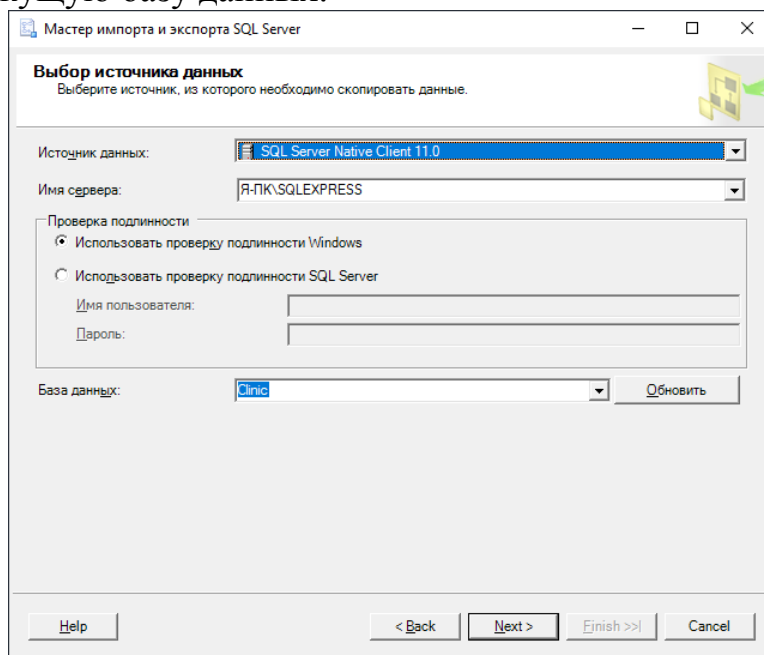


Рисунок 28 – Выбор места назначения при экспорте данных в текстовый файл

Затем выбирается тип файла и указывается путь к нему.

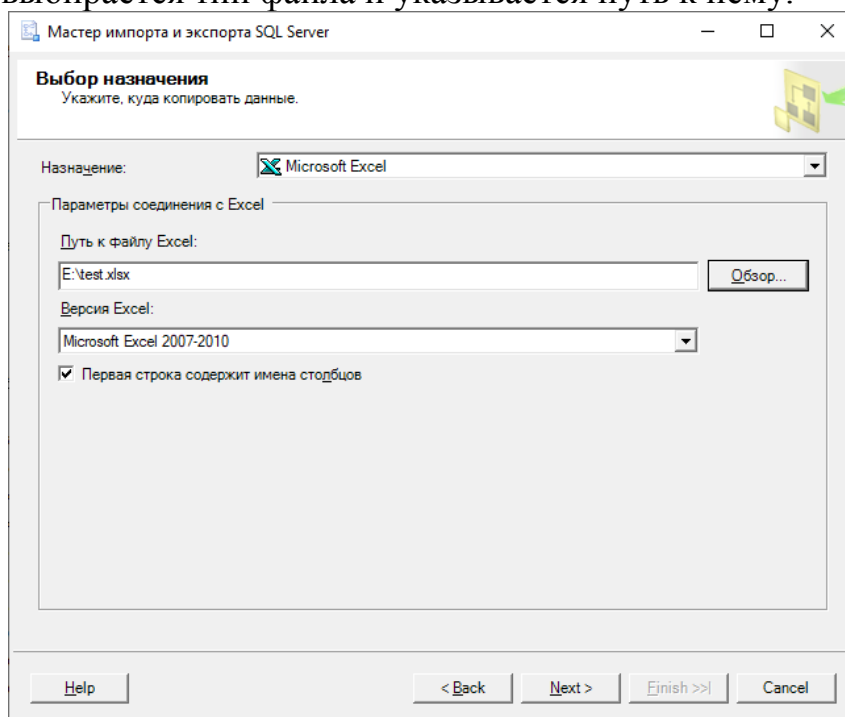


Рисунок 29 – Выбор места назначения при экспорте данных в текстовый файл

Затем выбираются таблицы, данные из которых будут экспортированы, проводится сопоставление типов данных между источником и приемником. Затем выводится окно, отображающее успешность проведения операции экспорта для каждого выбранного объекта.

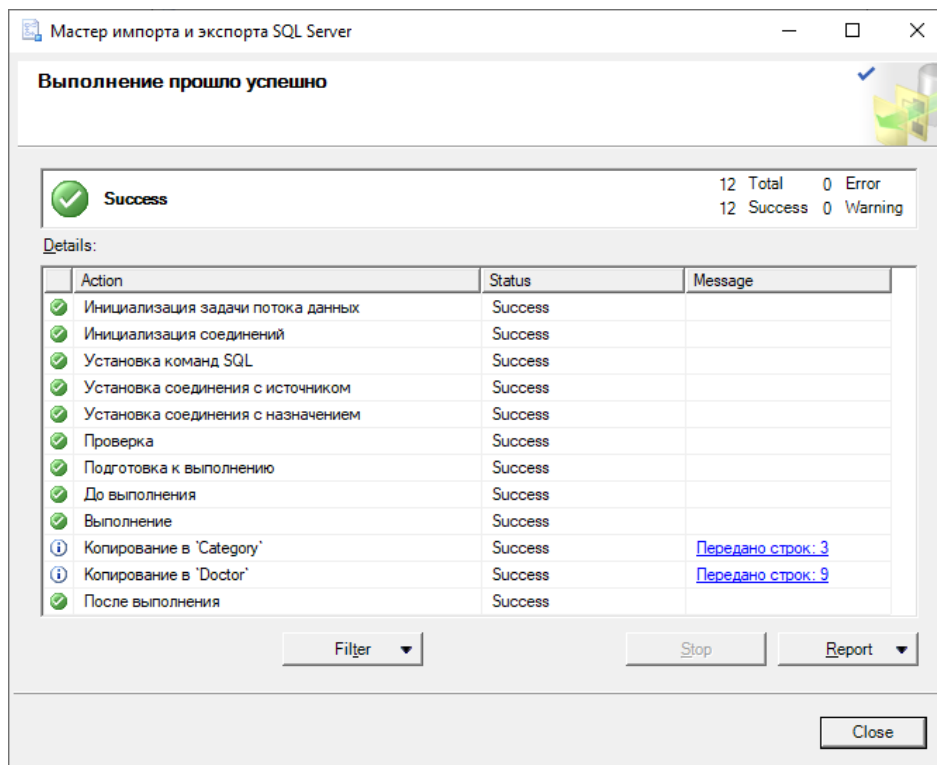


Рисунок 30 – Выбор места назначения при экспорте данных в текстовый файл

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 среде MySQL, в которую предполагается перенос данных.
3. Скорректировать исходные скрипты в соответствии с особенностями диалекта MySQL.
4. Создать необходимые объекты 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 текстовых файлов в таблицы целевой базы данных.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каким образом можно автоматически сгенерировать команды для создания и заполнения базы данных?
2. Какие методы создания копии баз данных вы знаете?
3. Каким образом можно выгрузить данные из таблицы БД в текстовый файл?
4. Какие команды для загрузки данных из текстовых файлов можно использовать в среде MySQL?

Практическая работа №9

СОЗДАНИЕ МЕХАНИЗМОВ СЕРВЕРА ДЛЯ ОБСЛУЖИВАНИЯ БАЗЫ ДАННЫХ

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

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

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

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, также 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 задания агента SQL и выполнять ежедневно.

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

Как известно, в базе данных msdb хранится история резервных копий баз данных. Теперь представим сервер, у которого баз данных более 50 и каждые 10-15 минут проходит создание резервной копии файла журнала транзакций.

Очистка истории резервного копирования осуществляется с помощью системной хранимой процедуры sp_delete_backuphistory:

sp_delete_backuphistory [@oldest_date =] «oldest_date»

где «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.

Системная хранимая процедура sp_clean_db_free_space удаляет остаточные данные, оставляемые на страницах базы данных процедурами изменения данных в SQL Server. sp_clean_db_free_space очищает все страницы во всех файлах базы данных.

```
sp_clean_db_free_space
[ @dbname = ] 'database_name'
[ , [ @cleaning_delay = ] 'delay_in_seconds' ] [;]
```

где @dbname = 'database_name' – имя очищаемой базы данных. Аргумент dbname имеет тип sysname и не может иметь значение null; @cleaning_delay = 'delay_in_seconds' – интервал задержки между операциями очистки страниц. Применение задержки помогает уменьшить нагрузку на систему ввода-вывода. delay_in_seconds имеет тип int и значение по умолчанию 0.

Управление и оптимизация ресурсов SQL Server производятся на основе параметров конфигурации с применением среды SQL Server Management Studio или системной хранимой процедуры sp_configure.

Параметры конфигурации могут вступать в силу:

- немедленно после установки параметра и выполнения инструкции RECONFIGURE (или, в некоторых случаях, RECONFIGURE WITH OVERRIDE). Повторная настройка определенных параметров аннулирует определенные параметры в кэше планов, что приводит к компиляции новых планов.
- после выполнения вышеуказанных действий и перезапуска экземпляра SQL Server.

Примеры параметров конфигурации: проверка подлинности автономной базы данных, язык по умолчанию, Database Mail XPs, show advanced options, max server memory, min memory per query, query wait, user connections, user options и др. (около 50 параметров конфигурации).

Разрешения на выполнение хранимой процедуры sp_configure без параметров или только с первым параметром по умолчанию предоставляются всем пользователям. Для выполнения процедуры sp_configure с обоими параметрами для изменения параметра конфигурации или запуска инструкции RECONFIGURE необходимо иметь разрешение ALTER SETTINGS на уровне сервера. Разрешение ALTER SETTINGS неявным образом предоставлено предопределенным ролям сервера sysadmin и serveradmin.

Системная хранимая процедура sp_configure отображает или изменяет глобальные параметры конфигурации текущего сервера.

```
sp_configure [ [ @configname = ] 'option_name'
[ , [ @configvalue = ] 'value' ] ]
```

где [@configname =] 'option_name' – имя параметра конфигурации.

[[@configvalue =] 'value' – это новый параметр конфигурации.

Пример: Активировать расширенные хранимые процедуры компонента Database Mail:

```
sp_configure 'show advanced options', 1;  
GO  
RECONFIGURE;  
GO  
sp_configure 'Database Mail XPs', 1;  
GO  
RECONFIGURE  
GO
```

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.
6. Удалить остаточные данные, оставленные на страницах базы данных процедурами изменения данных sp_clean_db_free_space
7. Удалить записи резервных наборов данных, которые старше указанной даты, дату задать самостоятельно (sp_delete_backuphistory)
8. Отобразить все глобальные параметры конфигурации текущего сервера (sp_configure). Отобразить выбранные параметры конфигурации сервера ('default language', 'Database Mail XPs', 'query wait')

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?

Практическая работа №10 РАБОТА С ЖУРНАЛОМ АУДИТА БД

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 отображения событий, происходящих с сервером и с его окружением.
- Ознакомиться с основными средствами просмотра данных этих журналов, изучить их возможности.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

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 – 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).

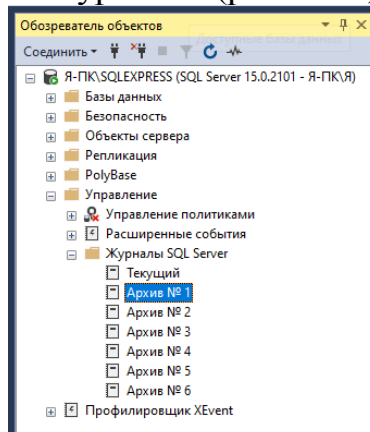


Рисунок 31 – Открытие окна просмотра журналов

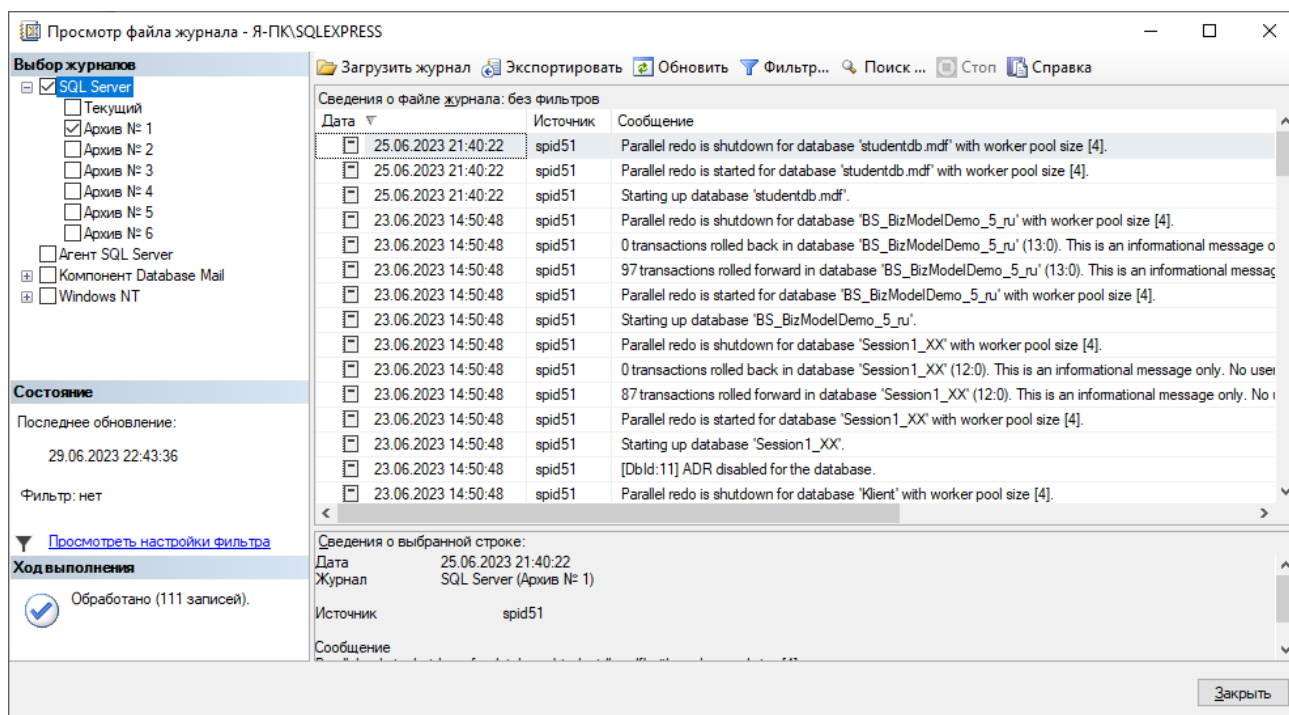


Рисунок 32 – Окно просмотра журналов

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, происходящих 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.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

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

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?

Практическая работа №11 МОНИТОРИНГ НАГРУЗКИ СЕРВЕРА

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, sp-monitor;
- ознакомиться с использованием системных процедур sp_server_diagnostics, sp_who и sp_who2.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

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.

Для просмотра фактической активности нужно разрешение VIEW SERVER STATE. Для просмотра раздела ввода-вывода в файл данных монитора активности, кроме разрешения на VIEW SERVER STATE, необходимо иметь также разрешения на CREATE DATABASE, ALTER ANY DATABASE или VIEW ANY DEFINITION.

Для вызова инструкции KILL для процесса пользователь должен быть членом предопределенных ролей сервера sysadmin или processadmin.

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 ожидающих 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.

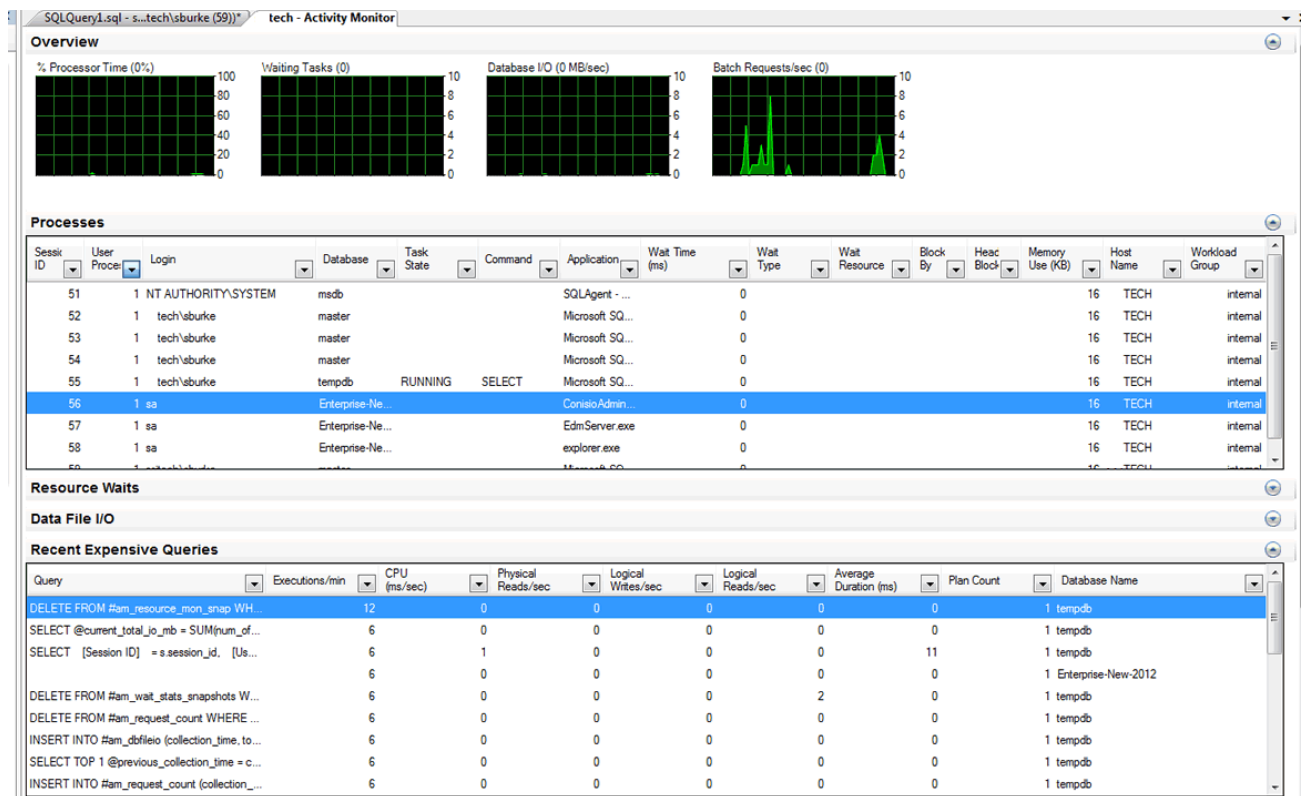


Рисунок 33 – Activity Monitor

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 нем отсутствует возможность гибкой настройки счетчиков производительности, указания пороговых значений и нет возможности сбора исторических данных.

Системная хранимая процедура `sp_monitor` отображает статистику о Microsoft SQL Server. SQL Server с помощью набора функций отслеживает объем проделанной работы. При выполнении `sp_monitor` отображаются текущие значения, возвращаемые этими функциями, и показывается, насколько они изменились с момента последнего выполнения процедуры. Результат выводится в виде таблицы.

Синтаксис: `sp_monitor`

Системная хранимая процедура `sp_server_diagnostics` возвращает диагностические данные и сведения о работоспособности SQL Server для выявления потенциальных неполадок. Для выполнения процедуры необходимо разрешение `VIEW SERVER STATE` на сервере.

`sp_server_diagnostics [@repeat_interval =] 'repeat_interval_in_seconds'`

Аргументы:

[@repeat_interval =] 'repeat_interval_in_seconds' - Указывает интервал времени, в течение которого хранимая процедура будет запускаться повторно для отправки сведений о работоспособности.

repeat_interval_in_seconds имеет тип int и значение по умолчанию 0. Допустимыми значениями для параметра являются 0 и любые значения, которые больше или равны 5 (чтобы вернуть полные данные, хранимая процедура должна работать не менее 5 секунд).

Если этот параметр не указан или задано значение 0, то хранимая процедура один раз вернет данные, а затем завершит работу.

В следующем примере выходные данные процедуры sp_server_diagnostics записываются в таблицу в режиме без повторения:

```
CREATE TABLE SpServerDiagnosticsResult
```

```
(
    create_time DateTime,
    component_type sysname,
    component_name sysname,
    state int,
    state_desc sysname,
    data xml
```

```
);
```

```
INSERT INTO SpServerDiagnosticsResult
```

```
EXEC sp_server_diagnostics;
```

Системная хранимая процедура sp_spaceused – выводит место на диске, зарезервированное и используемое объектами БД (таблицей, индексированным представлением или очередью компонента Компонент Service Broker в текущей базе данных).

Системная хранимая процедура sp_who – предоставляет сведения о текущих пользователях, сеансах и процессах в экземпляре Microsoft SQL Server.

Синтаксис:

```
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.

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

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Запустить мониторинг активности.
2. Выявить выполняемые процессы, время ожидания ресурсов и другие отображаемые параметры.
3. Отобразить статистику о Microsoft SQL Server (sp_monitor)
4. Используя выходные данные процедуры sp_server_diagnostics получить диагностические данные и сведения о работоспособности SQL Server.
5. Определить место на диске, зарезервированное и используемое указанной таблицей (sp_spaceused).
6. Получить сведения обо всех текущих пользователях, об отдельном текущем пользователе по имени его входа, об активных процессах (sp_who).

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_spaceused, sp_monitor, sp_server_diagnostics?
6. Какие параметры запуска могут быть использованы для данных процедур?

Практическая работа №12

УСТАНОВКА И НАСТРОЙКА СЕРВЕРА БД ORACLE

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.

ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

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.

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 учетной записи.



Рисунок 34 – Форма ввода электронной почты

2. Задание опций установки

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



Рисунок 35 –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.



Рисунок 36 – Выбор класса системы

4. Выбор базовых настроек установки

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

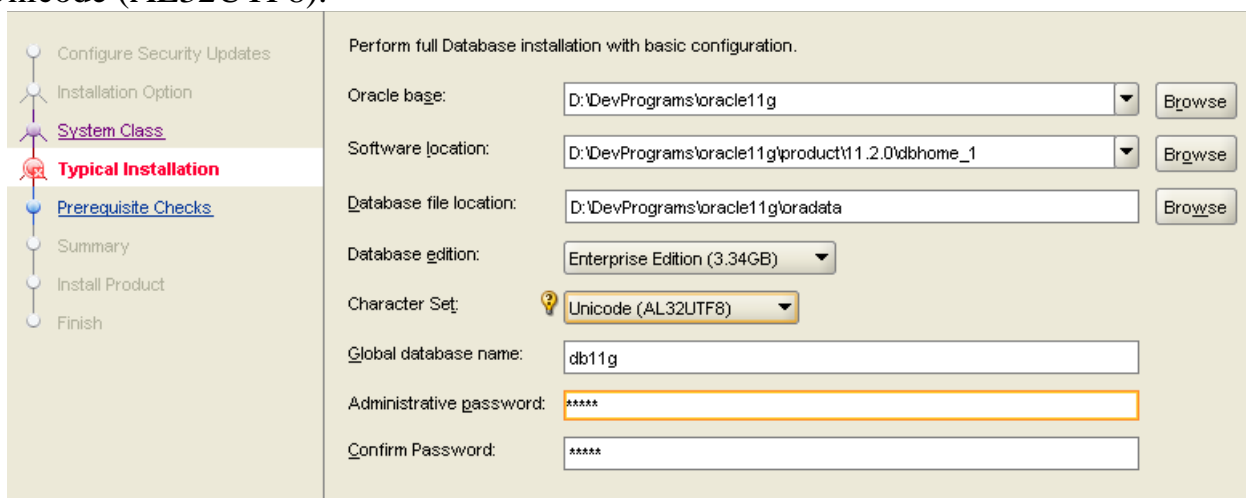


Рисунок 37 – Задание базовых настроек установки

5. Непосредственно выполнение установки

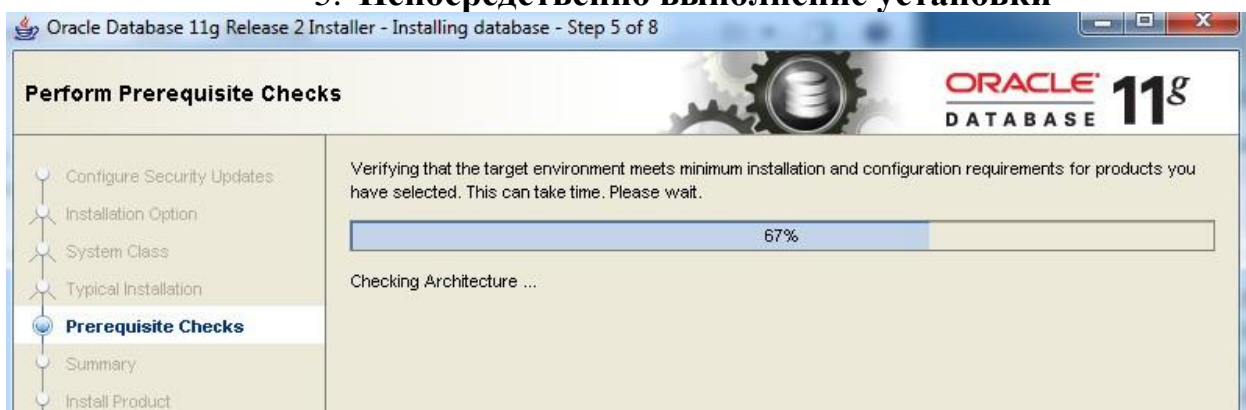


Рисунок 38 – Процесс установки

6. Отображение результатов установки

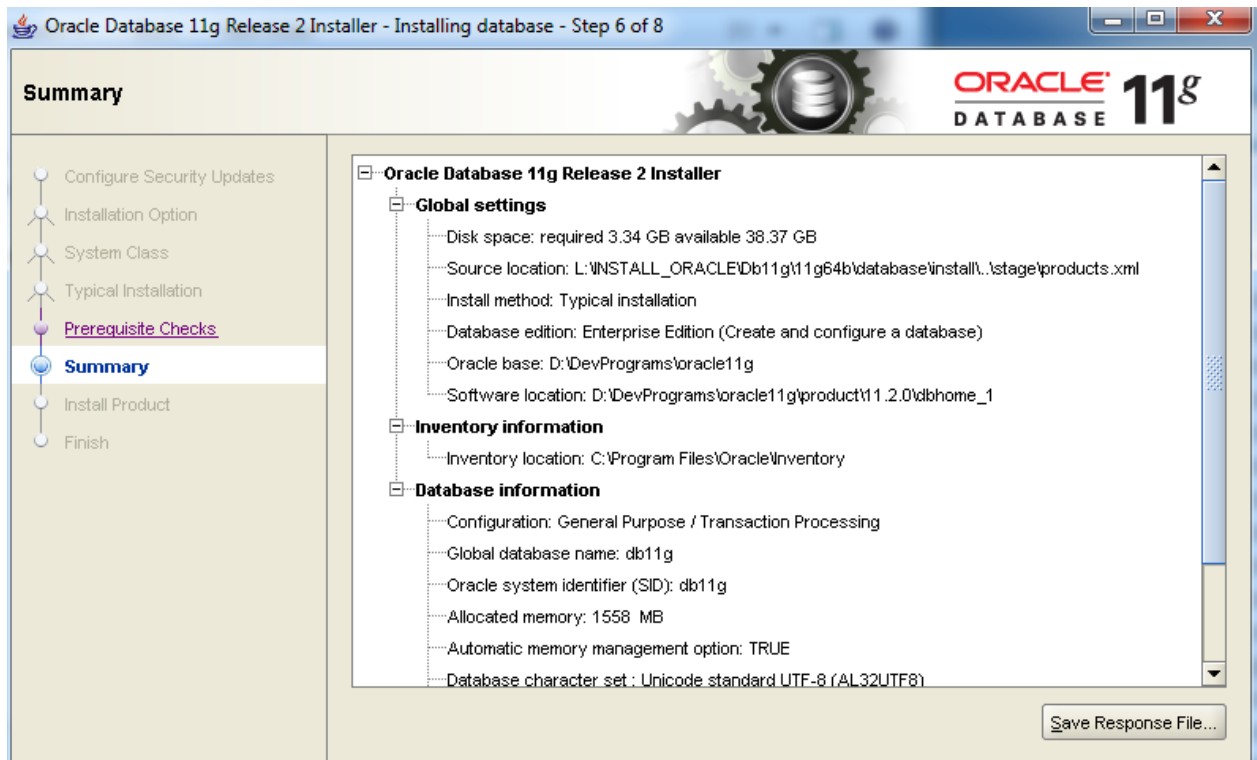


Рисунок 39 – Отображение результатов установки

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

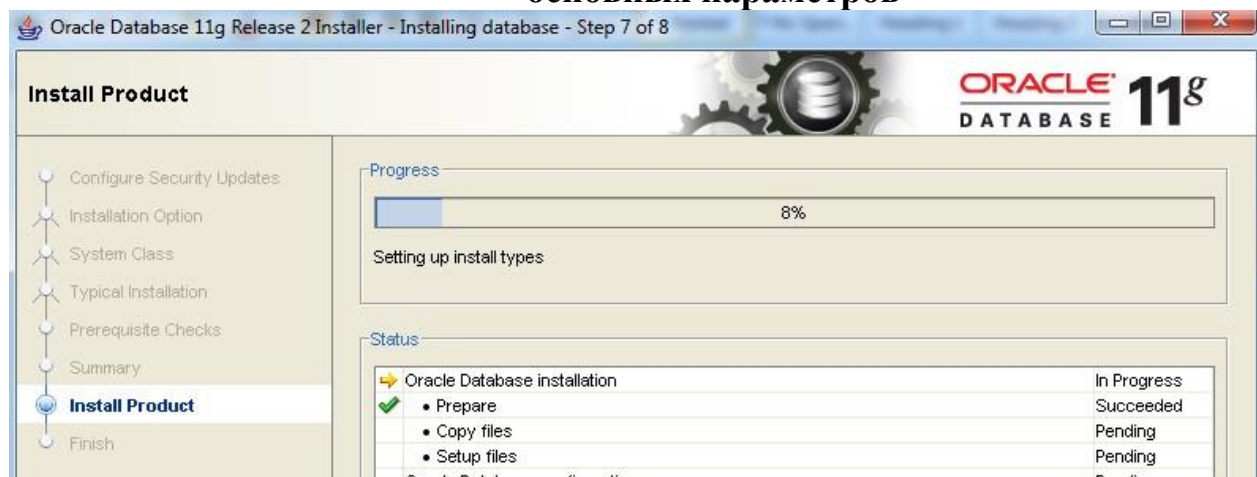


Рисунок 40 – Создание базы данных

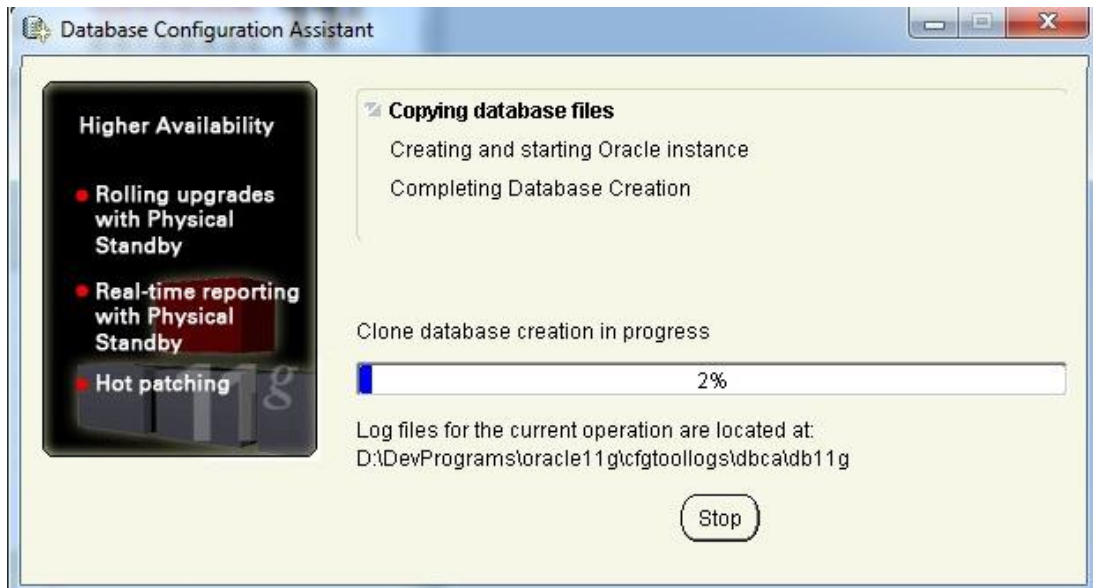


Рисунок 41 – Копирование файлов базы данных

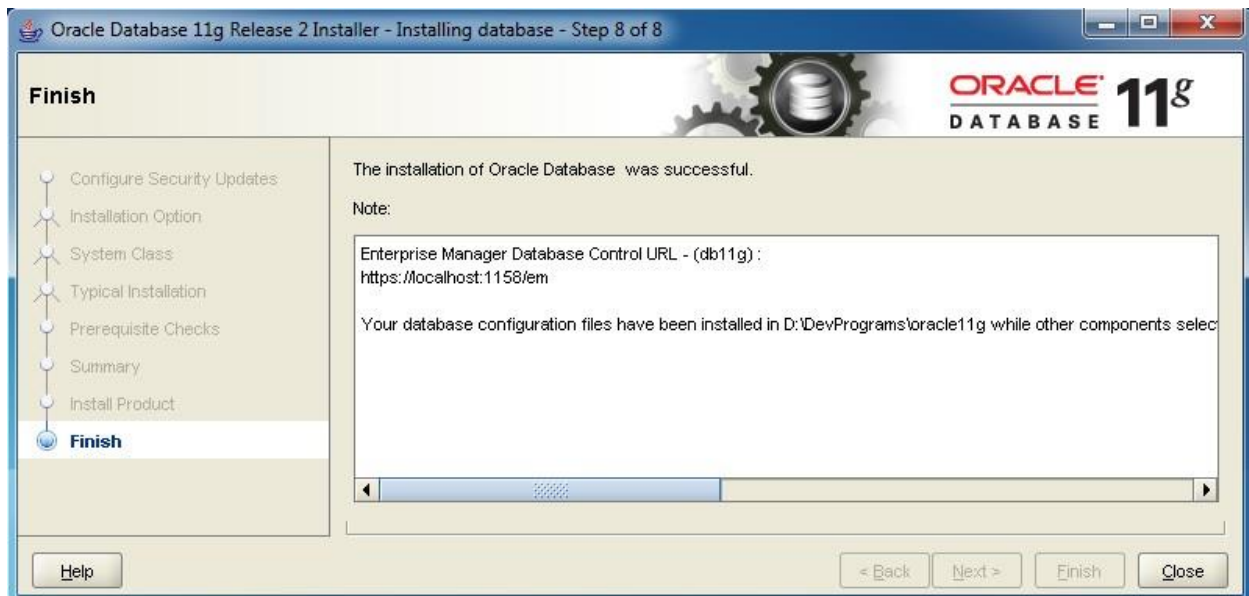


Рисунок 42 – Завершение установки

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 брандмауэр).

После установки СУБД Oracle отображается в виде системных служб.

Описание служб Oracle

1. Oracle DB Console

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

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.

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

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.

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 используются 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.

Аттестационной работой по курсу «Управление и автоматизация баз данных» является курсовой проект, поэтому в качестве самостоятельной работы выступают этапы выполнения курсового проекта.

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 БД в рамках работы над КП
5	Выполнение индивидуальных заданий по теме «Администрирование баз данных и серверов» в рамках работы над КП

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](https://msdn.microsoft.com/ru-ru/library/e80y5yhx(v=vs.110).aspx) - 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

СОДЕРЖАНИЕ

Практическая работа №1. ПОСТРОЕНИЕ БАЗЫ ДАННЫХ В СРЕДЕ ОДНОЙ ИЗ СУБД	3
Практическая работа №2. ПОСТРОЕНИЕ СХЕМЫ И СЛОВАРЯ БАЗЫ ДАННЫХ	10
Практическая работа №3. ИЗУЧЕНИЕ КОМАНД АДМИНИСТРИРОВАНИЯ ДАННЫХ ДЛЯ СРЕДЫ ОДНОЙ ИЗ СУБД	12
Лабораторная работа №1. РАЗРАБОТКА ТРЕБОВАНИЙ И КОНФИГУРИРОВАНИЕ КОРПОРАТИВНОЙ СЕТИ.....	18
Лабораторная работа №2. РАЗРАБОТКА МЕХАНИЗМОВ СЕРВЕРА БАЗ ДАННЫХ ХРАНИМЫЕ ПРОЦЕДУРЫ	21
Лабораторная работа №3. РАЗРАБОТКА МЕХАНИЗМОВ СЕРВЕРА БАЗ ДАННЫХ. ТРИГГЕРЫ	26
Практическая работа №4. КОПИРОВАНИЕ БАЗ ДАННЫХ, ИМПОРТ ЭКСПОРТ ДАННЫХ В СРЕДЕ MS SQL SERVER EXPRESS СРЕДСТВАМИ MANAGEMENT STUDIO	29
Практическая работа №5. КОПИРОВАНИЕ БАЗ ДАННЫХ СРЕДСТВАМИ КОМАНД SQL	35
Практическая работа №6. УСТАНОВКА И НАСТРОЙКА СЕРВЕРА MS SQL SERVER EXPRESS.....	39
Практическая работа №7. УСТАНОВКА И НАСТРОЙКА СЕРВЕРА БД MYSQL.....	43
Практическая работа №8. ПЕРЕНОС БАЗЫ ДАННЫХ НА ДРУГОЙ ТИП СЕРВЕРА	47
Практическая работа №9. СОЗДАНИЕ МЕХАНИЗМОВ СЕРВЕРА ДЛЯ ОБСЛУЖИВАНИЯ БАЗЫ ДАННЫХ	50
Практическая работа №10. РАБОТА С ЖУРНАЛОМ АУДИТА БД..	53
Практическая работа №11. МОНИТОРИНГ НАГРУЗКИ СЕРВЕРА .	56
Практическая работа №12. УСТАНОВКА И НАСТРОЙКА СЕРВЕРА БД ORACLE	60
СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ	66
УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ	67