

ЛАБОРАТОРНАЯ РАБОТ 2. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ С ПОМОЩЬЮ СУБД MICROSOFT SQL SERVER

Для выполнения данной лабораторной работы Вам понадобится установленная СУБД Microsoft SQL Server версии не ниже 2014. Процесс установки представлен в разделе 5 данного методического указания. Взаимодействие с СУБД Microsoft SQL Server осуществляется через графическую оболочку Microsoft SQL Server Management Studio.

ЗАДАНИЕ

- 1) Изучить пользовательский интерфейс SQL Server Management Studio
- 2) Создать БД с помощью мастера и с помощью запроса (в отчете отобразить создание с помощью обоих методов)
- 3) Создать таблицы в соответствии с моделью, созданной в Лабораторной работе №1 (одни таблицы создать с помощью мастера, другие с помощью запроса)
- 4) Создать связи между таблицами
- 5) Средствами SQL Server Management Studio создать диаграмму БД (полученную диаграмму необходимо отобразить в отчете)
- 6) Заполнить таблицы данными, выполнить запросы на изменение и удаление данных
- 7) Сделать выводы по работе

ХОД РАБОТЫ

1) Установить SQL Server версии не ниже 2014 если не установлен **ИЛИ** установить SQL Server Management Studio и подключится к серверу. Данные для подключения вы сможете найти на [сайте преподавателя](#).

2) Осуществить подключение к БД, используя логин: student и пароль, указанный при установке сервера;

3) Используя инструмент SQL Management Studio написать запросы для создания БД, а также таблиц;

(Внимание! Связи между сущностями указывать при создании таблицы используя конструкцию FOREIGN KEY)

4) Используя инструмент SQL Management Studio написать запросы на модификацию, удаление и добавления данных в таблицы. Примеры запросов указаны в конце данного методического пособия;

5) Написать отчет о проделанной работе, оформленный согласно требованиям, включающий в себя:

- титульный лист;
- цель работы;

- выполненные задачи;
- ход работы (в ходе работе представить SQL запросы на создание БД и таблиц, а также запросы на добавление, удаление и модификацию данных в таблицах);
- заключение

ОГЛАВЛЕНИЕ

ЗАДАНИЕ	1
ХОД РАБОТЫ	1
1. ПРИМЕР СОЗДАНИЯ НОВОЙ БАЗЫ ДАННЫХ.....	4
1.1. Создание новой базы данных с помощью мастера	4
1.2. Язык запросов SQL	11
1.2.1. Создание новой БД с помощью запроса	13
2. СОЗДАНИЕ ТАБЛИЦ	15
2.1. Пример создания таблиц	15
2.1.1. Создание таблиц с помощью мастера.....	16
2.1.2. Создание таблиц с помощью запроса.....	19
2.2. Связывание таблиц.....	21
2.2.1. Создание диаграммы базы данных	21
2.2.2. Связывание таблиц с помощью конструкции FOREIGN KEY	25
3. ЯЗЫК ЗАПРОСОВ SQL	28
3.1. Работа с данными	28
3.1.1. Добавление данных в таблицу	29
3.1.2. Удаление отдельных столбцов и отдельных строк из таблицы	31
3.1.3. Изменение данных в таблице	32
ЗАКЛЮЧЕНИЕ	33

1. ПРИМЕР СОЗДАНИЯ НОВОЙ БАЗЫ ДАННЫХ

1.1. Создание новой базы данных с помощью мастера

Создание любой БД начинается с создания файла данных. Рассмотрим этот процесс в Microsoft SQL Server на примере создания простой БД по учёту успеваемости студентов. Перед запуском убедитесь, что служба SQL Server запущена, открыв раздел «Службы», через меню Пуск.

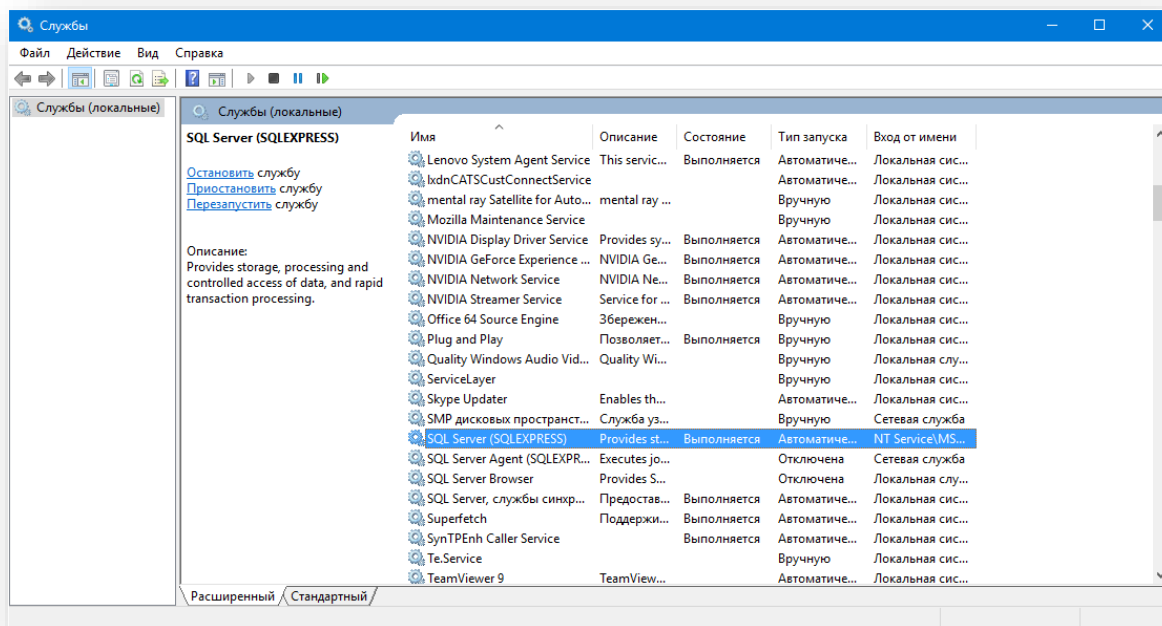


Рис. 2.1. Запуск службы SQL Server

Для начала необходимо запустить среду разработки SQL Server Management Studio. Для этого в меню «Пуск» выбираем пункт Программы\Microsoft SQL Server 20XX\SQL Server Management Studio

После запуска среды разработки появится окно подключения к серверу Соединение с сервером (рисунок 2.2).

Для подключения к серверу необходимо ввести в поле «Имя сервера»: 109.123.145.31, проверка подлинности должна быть выбрана «Проверка подлинности SQL Server», «Имя входа» и «Пароль»: student.

В этом окне необходимо нажать кнопку Соединить.

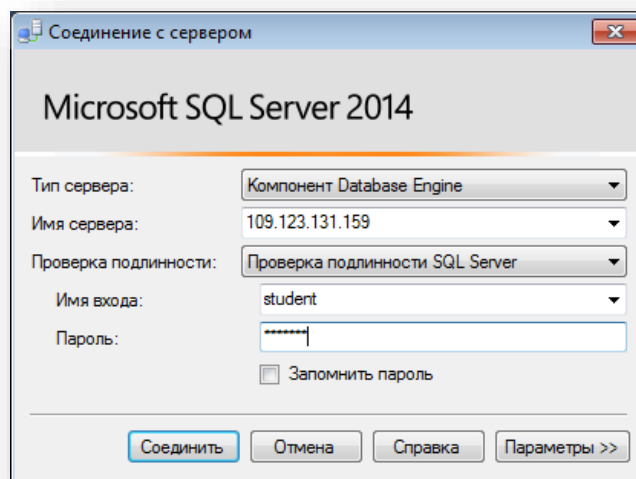


Рисунок 2.2 – Соединение с сервером

Если в процессе установки Microsoft SQL Server 20XX был задан логин и пароль подключения к серверу, то перед нажатием кнопки «Соединить», в выпадающем списке Проверка подлинности нужно выбрать Проверка подлинности SQL Server, а затем необходимо ввести заданные при установке логин и пароль.

После нажатия кнопки «Соединить» появится окно среды разработки SQL Server Management Studio (рисунке 2.3).

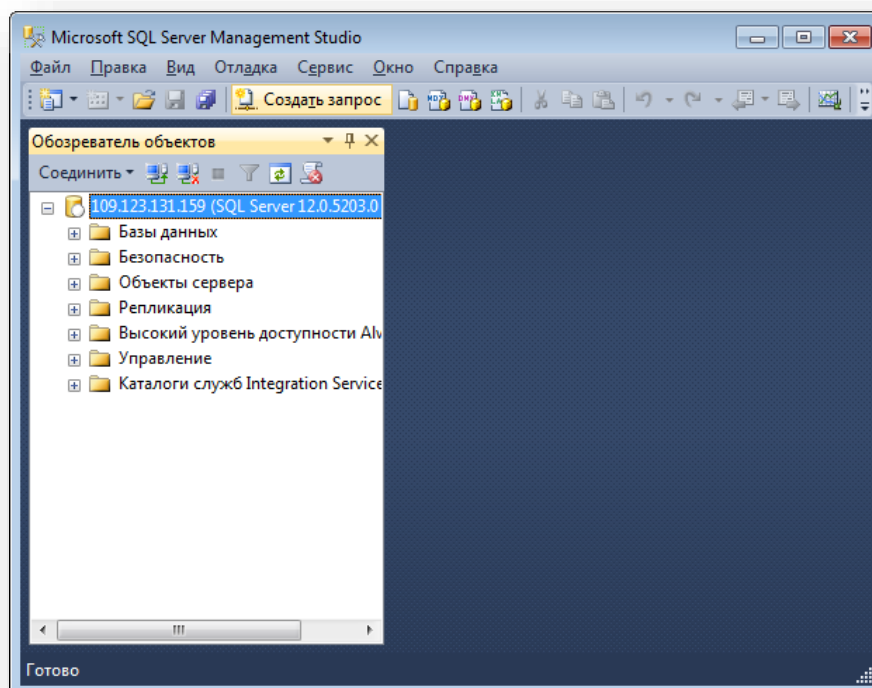


Рисунок 2.3 – Рабочее окно Server Management Studio

В самом верху расположено меню, которое содержит полный набор команд для управления сервером и выполнения различных операций.

Ниже располагается панель, которая содержит кнопки для выполнения наиболее часто производимых операций. Внешний вид данной панели зависит от выполняемой операции.

Ниже расположена Панель обозревателя объектов – это панель с древовидной структурой, отображающая все объекты сервера, а также позволяющая производить различные операции, как с самим сервером, так и с БД. Обозреватель объектов является основным инструментом для разработки БД.

В обозревателе объектов сами объекты находятся в папках. Чтобы открыть папку необходимо щёлкнуть по знаку «+» слева от изображения папки.

Теперь перейдём непосредственно к созданию файла данных. Для этого в обозревателе объектов щёлкните правой кнопкой мыши на папке Базы данных (рисунок 2.4) и в появившемся меню выберите пункт «Создать базу данных...».

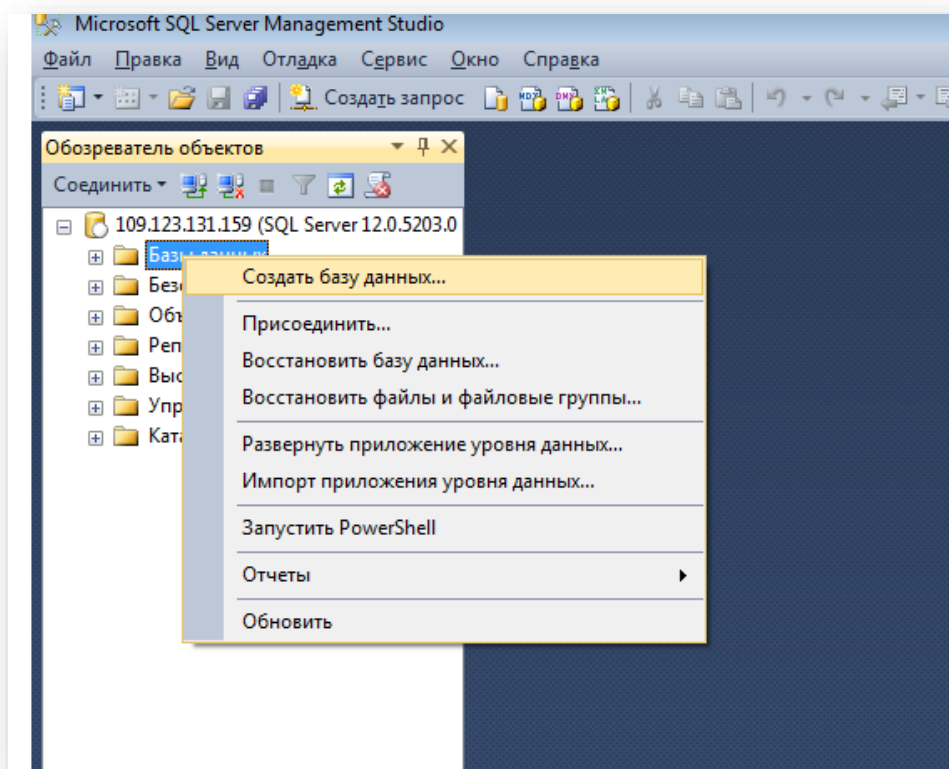


Рисунок 2.4 – Создание базы данных

Появится окно настроек параметров файла данных новой базы данных Создание базы данных (рисунок 2.5). В левой части окна настроек имеется список «Выбор страницы». Этот список позволяет переключаться между группами настроек.

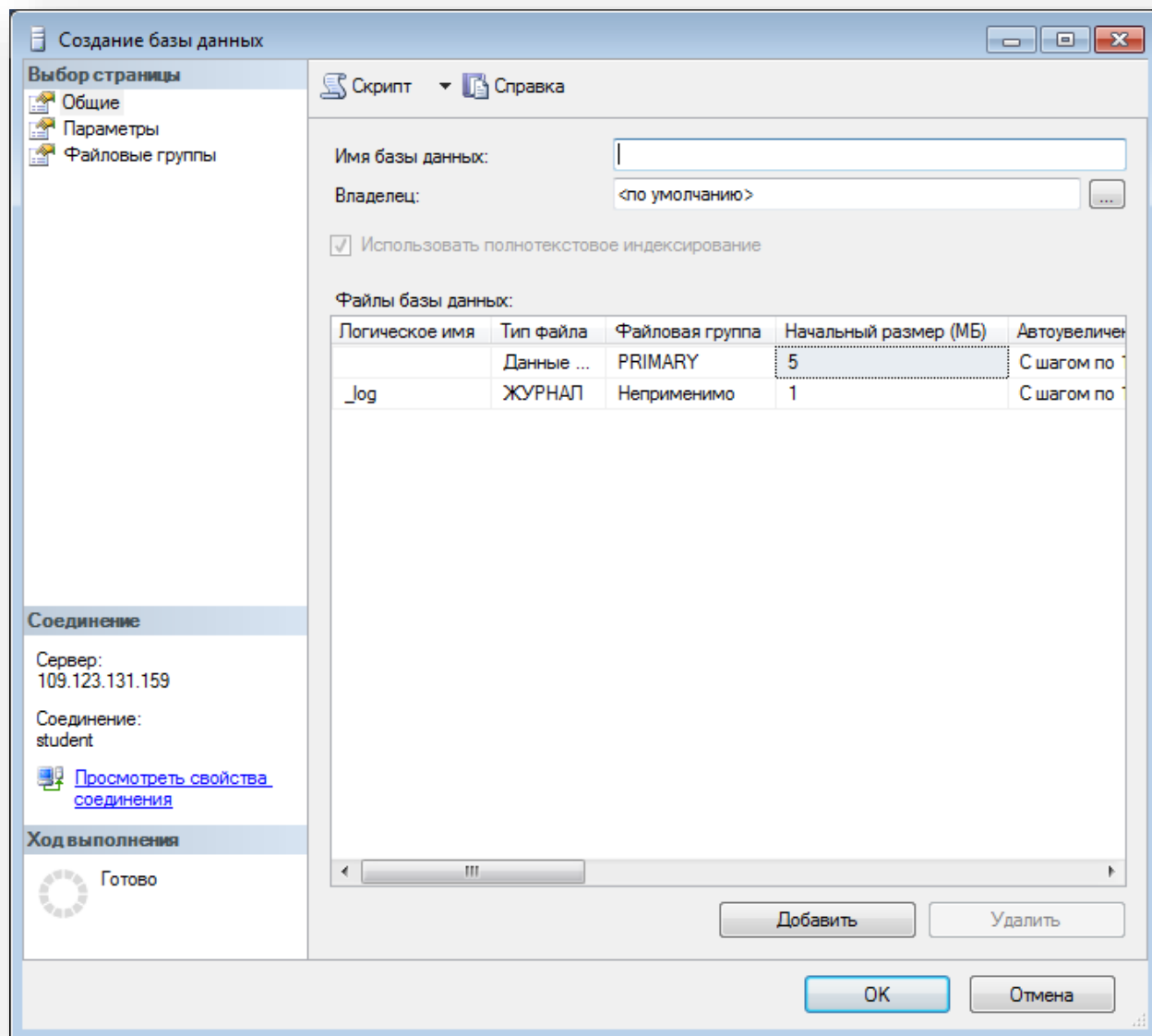


Рисунок 2.5 – Окно «Создание базы данных», страница «Общие»

Для начала настроим основные параметры. Для выбора основных настроек нужно щёлкнуть мышью по пункту «Общие» в списке «Выбор страницы». В правой части окна «Создание базы данных» появятся основные настройки (рисунок 2.5):

Верхней части окна расположено два параметра: «Имя базы данных» и «Владелец». Необходимо задать параметр «Имя базы данных», в котором должно содержаться название группы, фамилия и название базы данных в таком формате 8BM64_Ivanov_databasename. Параметр Владелец оставьте без изменений. Под вышеприведёнными параметрами в виде таблицы располагаются настройки файла данных и журнала транзакций. Таблица имеет следующие столбцы:

– **Логическое имя** – логическое имя файла данных и журнала транзакций. По этим именам будет происходить обращение к вышеприведённым файлам в БД. Можно заметить, что файл данных имеет то же имя что и БД, а имя файла журнала транзакций составлено из имени БД и суффикса «_log».

– **Тип файла** – этот параметр показывает, является ли файл файлом данных или журналом транзакций.

– **Файловая группа** – группа файлов, показывает к какой группе файлов относится файл. Группы файлов настраиваются в группе настроек Файловая группа.

– **Начальный размер** (МБ) – начальный размер файла данных и журнала транзакций в мегабайтах.

– **Авторасширение** – как только файл заполняется информацией его размер автоматически увеличивается на величину, указанную в параметре Авторасширение. Увеличение можно задавать как в мегабайтах так и в процентах. Здесь же можно задать максимальный размер файлов. Для изменения этого параметра надо нажать кнопку «...». В нашем случае (рисунок 2.5) размер файлов не ограничен. Файл данных увеличивается на 1 мегабайт, а файл журнала транзакций на 10%.

– **Путь** – путь к папке, где хранятся файлы. Для изменения этого параметра также надо нажать кнопку «...».

– **Имена файлов** – по умолчанию имена файлов аналогичны логическим именам. Однако файл данных имеет расширение .mdf, а файл журнала транзакций – расширение .ldf. В рассматриваемом случае все основные настройки без изменений были оставлены без изменений. Теперь перейдём к другим второстепенным настройкам файла данных. Для доступа к этим настройкам необходимо щёлкнуть мышью по пункту Параметры в списке Выбор страницы. Появится следующее окно (рисунок 2.5). В правой части окна мы видим следующие настройки:

– **Параметры сортировки** – этот параметр отвечает за обработку текстовых строк, их сравнение, текстовый поиск и т.д. Рекомендуется оставить его как. При этом данный параметр будет равен значению, заданному на вкладке Параметры сортировки, при установке сервера.

– **Модель восстановления** – данный параметр отвечает за информацию, предназначенную для восстановления БД, хранящуюся в файле транзакций. Чем полнее модель восстановления, тем больше вероятность восстановления данных при сбое системы или ошибках пользователей, но и больше размер файла журнала транзакций. При наличии места на диске, рекомендуется оставить этот параметр в значении Простая.

– **Уровень совместимости** – определяет совместимость файла данных с более ранними версиями сервера. Если планируется перенос данных на другую, более раннюю версию сервера, то её необходимо указать в этом параметре.

– *Другие параметры* – данные параметры являются необязательными для изменения. В нашем случае все параметры в разделе Другие параметры, рекомендуется оставить как на рисунке 2.6.

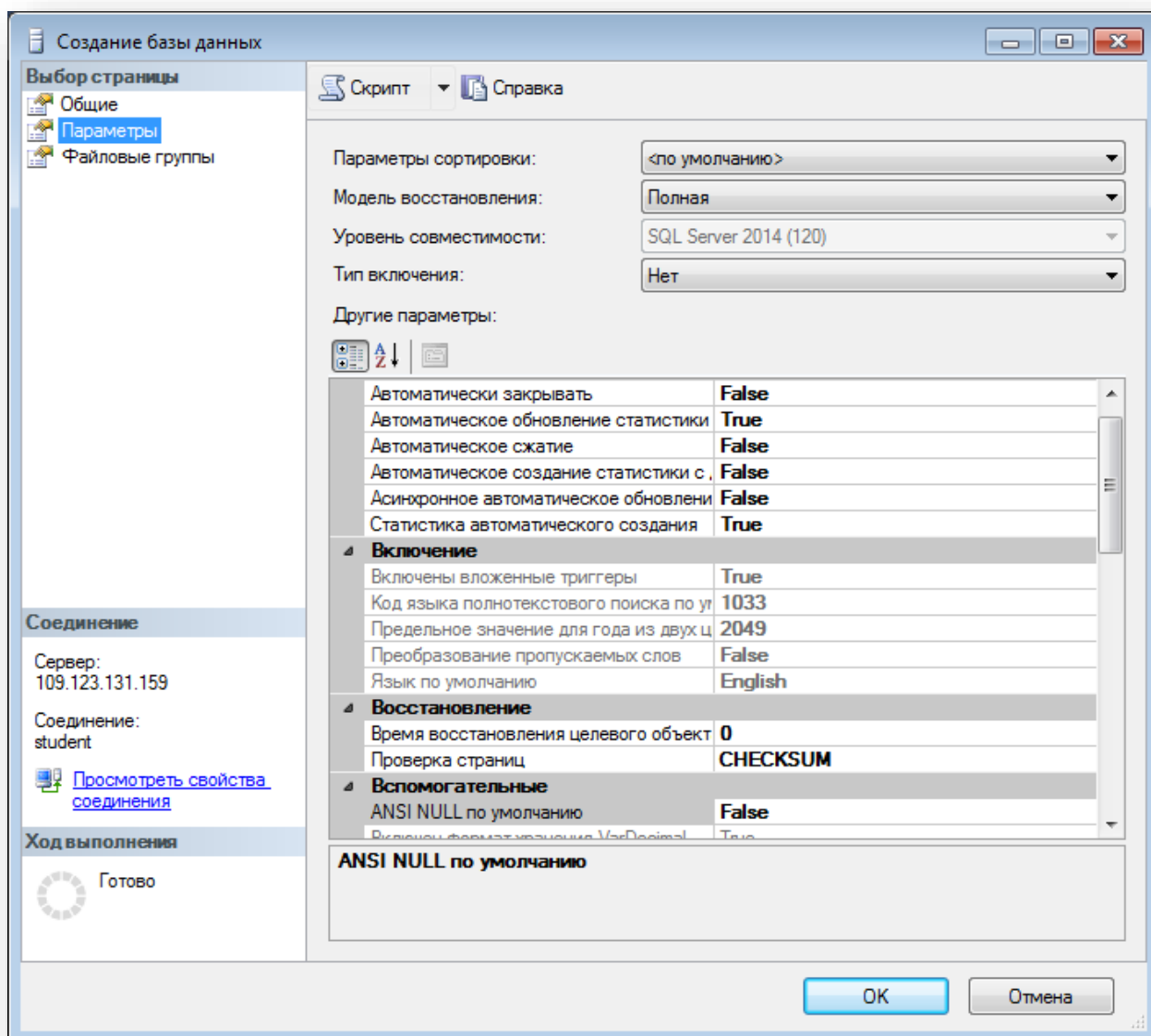


Рис. 2.6 – Окно «Создание базы данных», страница «Параметры»

Наконец рассмотрим последнюю группу настроек Файловые группы. Данная группа настроек отвечает за группы файлов (рисунок 2.6). Группы файлов представлены в таблице Строки в правой части окна (Рис.2.6).

Данная таблица имеет следующие столбцы:

- *Имя* – имя группы файлов.
- *Файлы* – количество файлов, входящих в группу.

– **Только для чтения** – файлы в группе будут только для чтения. То есть, их можно только просматривать, но нельзя изменять.

– **По умолчанию** – группа по умолчанию. Все новые файлы данных будут входить в эту группу. В рассматриваемой БД нет необходимости добавлять новые группы файлов. Поэтому оставим группу настроек Файловые группы без изменений.

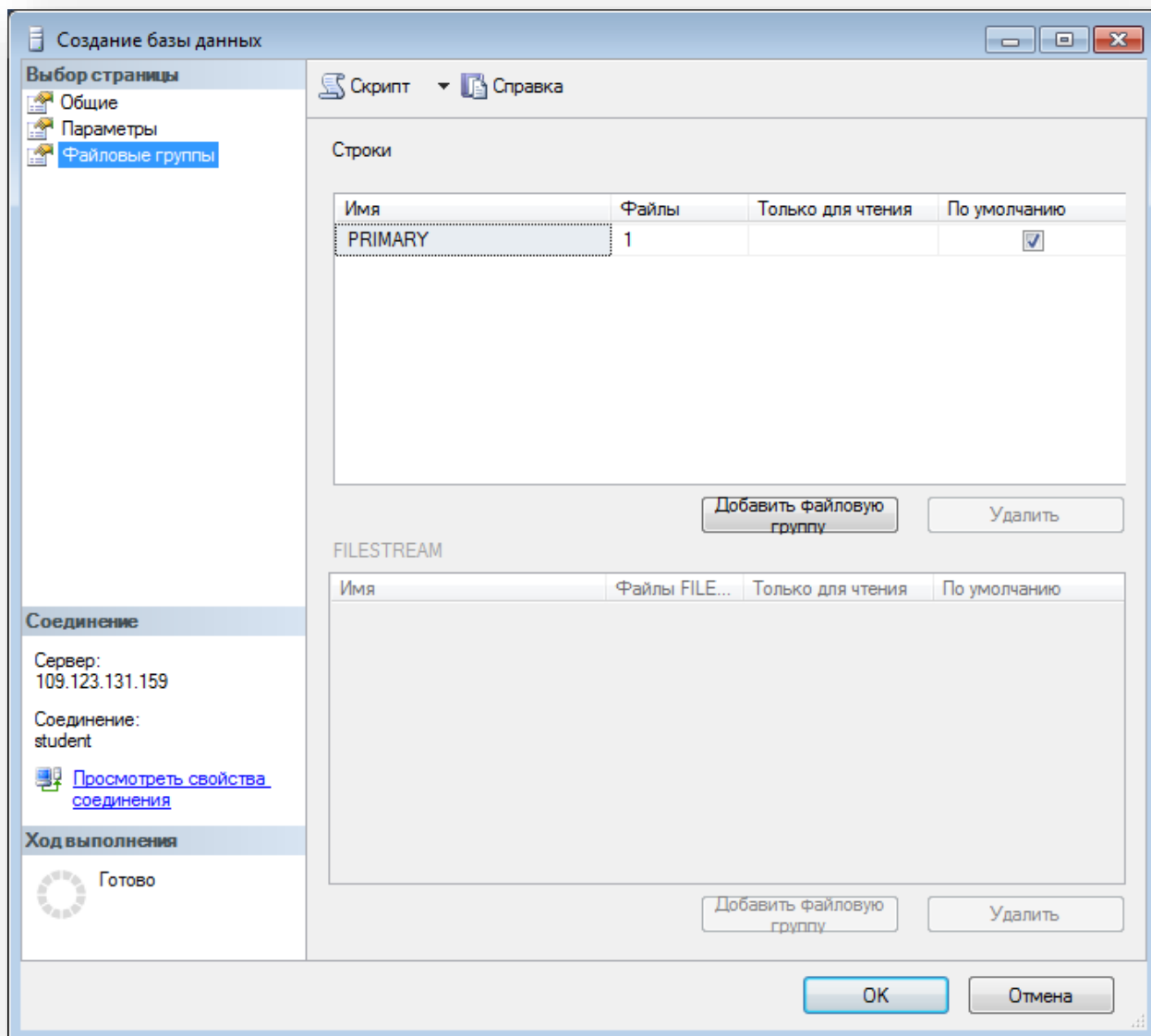


Рис. 2.7 – Окно Создание базы данных, страница Файловые группы

На этом мы заканчиваем настройку свойств наших файлов. Для принятия всех настроек и создание фала данных и журнала транзакций нашей БД в окне «Создание базы данных» нажмём кнопку ОК.

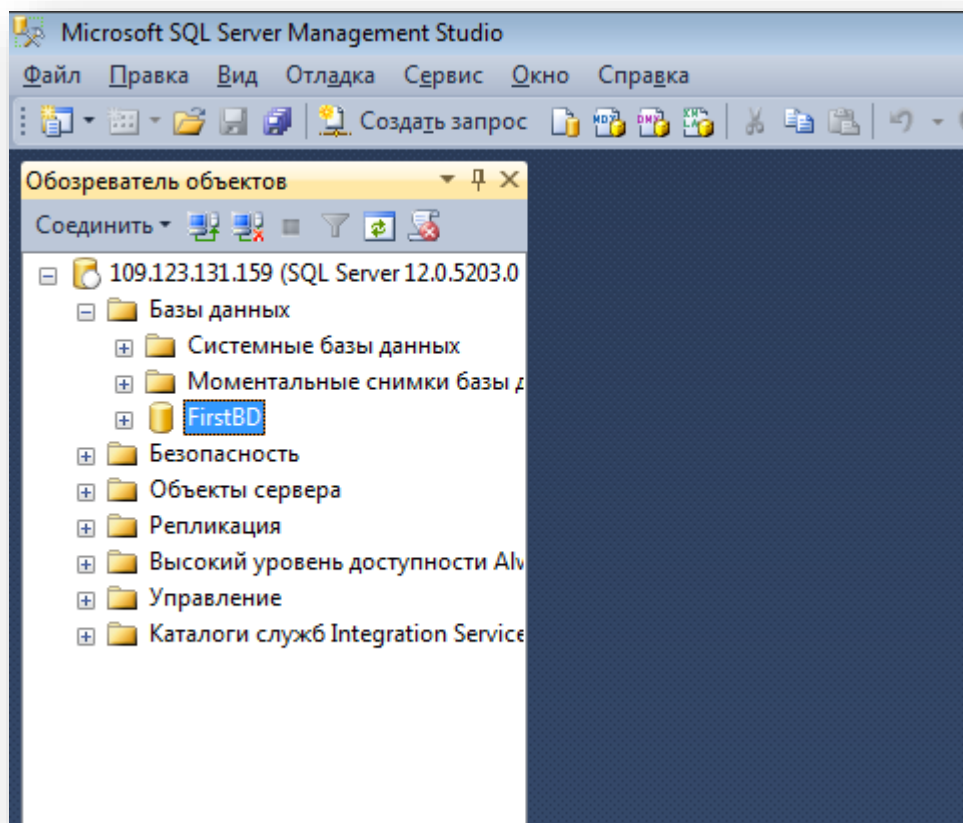


Рис. 2.8 – Результат создания БД

Произойдёт возврат в окно среду разработки SQL Server Management Studio. На панели обозревателя объектов в папке «Базы данных» появиться новая база данных «FirstBD».

1.2. Язык запросов SQL

Появление и развитие языка SQL связано с созданием теории реляционных БД. Математической основой языка SQL является реляционная алгебра и реляционное исчисление.

Прообраз языка возник в 1970 году в лаборатории Санта-Тереза фирмы IBM. В настоящее время популярность SQL настолько велика, что разработчики нереляционных СУБД снабжают свои системы SQL-интерфейсом.

SQL сочетает в себе возможности языка определения данных, языка манипулирования данными и языка запросов. При этом он реализует и основные функции реляционных СУБД.

SQL не является языком программирования в традиционном представлении. На нем пишутся не программы, а запросы к БД, поэтому этот язык называют языком

запросов, языком *декларативным*, а не процедурным. Это означает, что с его помощью можно сформулировать, что необходимо получить, однако нельзя указать, как это следует сделать. В отличие от процедурных языков программирования (Си, Паскаль), в языке SQL отсутствуют алгоритмические конструкции, операторы цикла, условные переходы и т.д.

Язык SQL выходит в состав 4 семейств языков и выполняет их роли.

SQL является языком DDL (Data Definition Language – язык описания данных) так как реализует следующие функции:

- CREATE – создание объектов в БД;
- ALTER – изменение структуры БД;
- DROP – удаление объектов из БД;
- TRUNCATE – удаление всех записей из БД;
- COMMENT – добавление комментариев;
- RENAME – изменение имени объекта БД.

SQL является языком DML (Data Manipulation Language – язык манипулирования данными) так как реализует следующие функции:

- SELECT – извлечение данных из БД;
- UPDATE – обновление данных в БД;
- DELETE – удаление данных из БД;
- INSERT – вставка данных в БД;
- MERGE – вставка и обновление;
- CALL – вызов PL/SQL или Java подпрограммы;
- EXPLAIN PLAN – создание планов выполнения запросов;
- LOCK TABLE – для блокирования таблиц.

SQL является языком DCL (Data Control Language – язык управления данными) так как реализует следующие функции:

- GRANT – предоставляет права доступа пользователям к БД;
- REVOKE – отмена прав доступа.

SQL является языком TCL (Transaction Control Language – язык транзакций) так как реализует следующие функции:

- COMMIT – сохранение работы;
- SAVEPOINT – определение точки транзакции;
- ROLLBACK – восстановление БД на оригинал с последним COMMIT;
- SET TRANSACTION – изменение параметров транзакций.

Запрос в языке SQL состоит из одного или нескольких операторов, следующих один за другим и разделенных точкой с запятой. Каждая последовательность операторов языка SQL реализует определенное действие над БД. Оно осуществляется за несколько шагов, на каждом из которых над таблицами выполняются определенные действия.

Каждый оператор SQL начинается с ключевого слова, которое определяет, что делает этот оператор (SELECT, INSERT, DELETE).

В операторе содержатся предложения, содержащие сведения о том, над какими данными производятся операции. Каждое предложение начинается с ключевого слова, такого как FROM, WHERE и др.

Структура предложения зависит от его типа: ряд предложений содержит имена полей или таблиц, некоторые могут включать дополнительные ключевые слова, константы или выражения. Для примера приведем структуру запроса SELECT на рисунке 2.9.

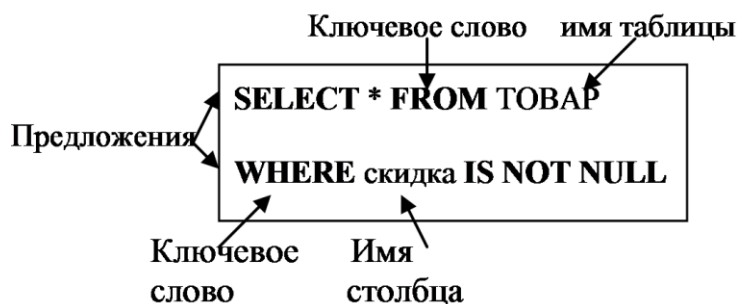




Рисунок 2.9 – структура запроса SELECT

1.2.1. Создание новой БД с помощью запроса

Новую БД можно создать, используя стандартные команды языка T-SQL. Все команды языка T-SQL набираются на вкладке нового запроса (SQLQuery). Для того чтобы создать новый запрос на панели инструментов необходимо нажать кнопку  Создать запрос. Для выполнения команд языка T-SQL на панели инструментов необходимо нажать кнопку  Выполнить или на вкладке нового запроса набрать команду GO.

Для создания нового файла данных используется команда CREATE DATABASE, которая имеет следующий синтаксис:

```
CREATE DATABASE [Имя БД] ON PRIMARY
(
    NAME = <Логическое имя>,
    FILENAME = <Имя файла>,
```

```

SIZE = <Нач.размер>,
MAXSIZE = <Макс.размер>,
FILEGROWTH = <Шаг> )
LOG ON
(
NAME = <Логическое имя>,
FILENAME = <Имя файла>,
SIZE = <Нач.размер>,
MAXSIZE = <Макс.размер>,
FILEGROWTH = <Шаг> )
)

```

Пример: Создать БД «publishing», расположенную в файле D:\publishing.mdf и имеющую начальный размер файла данных 5 Мб., максимальный размер файла данных неограничен. и шаг увеличения файла данных равный 1 Мб. Файл журнала транзакций данной БД имеет имя publishing_log и расположен в файле D:\publishing_log.ldf. Данный файл имеет начальный размер равный 1 Мб., максимальный размер равный 2 Гб. и шаг увеличения равный 100 Кб.

```

CREATE DATABASE [publishing] ON PRIMARY
(
NAME = 'publishing',
FILENAME = 'D:\publishing.mdf' ,
SIZE = 5120KB ,
MAXSIZE = UNLIMITED,
FILEGROWTH = 1024KB )
LOG ON
(
NAME = 'publishing_log',
FILENAME = 'D:\publishing_log.ldf' ,
SIZE = 1024KB ,
MAXSIZE = 2048GB ,
FILEGROWTH = 10%
)
GO

```

2. СОЗДАНИЕ ТАБЛИЦ

После создания базы данных мы пока не можем вносить в нее какие-либо данные т.к. пока еще не создана структура куда мы можем их поместить и работать с ними. Для создания этой структуры необходимо создать в БД таблицы-сущности, которые были спроектированы в Лабораторной работе №1.

2.1. Пример создания таблиц

Вся информация в базе данных хранится в таблицах. Таблицы состоят из записей. Запись – это строка в таблице. Вся информация обрабатывается по записям. Каждая запись состоит из полей. После это столбец таблицы. Каждое поле имеет три характеристики:

Имя поля – используется для обращения к полю;

Значение поля – определяет информацию, хранимую в поле;

Тип данных поля – определяет, какой вид информации можно хранить в поле.

В SQL сервер используются следующие типы данных:

- **Битовые типы данных**, которые содержат последовательности нулей и единиц: *Binary(n)* и *Varbinary(n)*, где *n* длина. Содержимое полей типа *Binary* всегда равно *n*, разница заполняется пробелами. *Varbinary* размер поля равен *n* или большему;
- **Целочисленные типы данных** – типы данных для хранения целых чисел (в скобках указан диапазон значений типа данных): *Tinyint* (0-255), *Smallint* (±32000), *Int* (±2000000000), *Bigint* (±2⁶³);
- **Типы данных для хранения дробных чисел**: *Real* семь знаков после запятой, *Float(m)* может хранить числа из *m* знаков, максимальное *m*=38, *Decimal(m n)* дробные числа с *m* знаков до запятой и *n* после;
- **Специальные типы данных**: *Bit* – логический тип данных является заменой логическому типу *Boolean* в Visual Basic, *Text* - тип для хранения больших объемов текста, одно поле может хранить до 2 Гб текста, *Image* – тип данных для хранения до 2Гб рисунков, *RowGUID* – уникальный идентификатор строки таблицы, *SQL_Variant* - аналогичен типу *Variant* в Visual Basic;
- **Типы данных даты и времени**: *Datetime* (от 1.01.1953 до 3.12. 1999). *SmallDatetime* (от 1.01.19 до 6.07 2079);
- **Денежные типы данных для хранения финансовой информации**: *Money* (±1015 и 4 знака после нуля), *Smallmoney* (± 20000,0000);
- **Автоматически обновляемые типы данных** - аналоги счетчиков, но в данной роли они не используются: *RowVersion* уникальный идентификатор строки. *TimeStamp* – закодированное дата и время создания строки.

2.1.1. Создание таблиц с помощью мастера

Перейдём теперь к созданию таблиц. Все таблицы нашей БД находятся в подпапке «Таблицы» папки «publishing» в окне обозревателя объектов (рисунок 6).

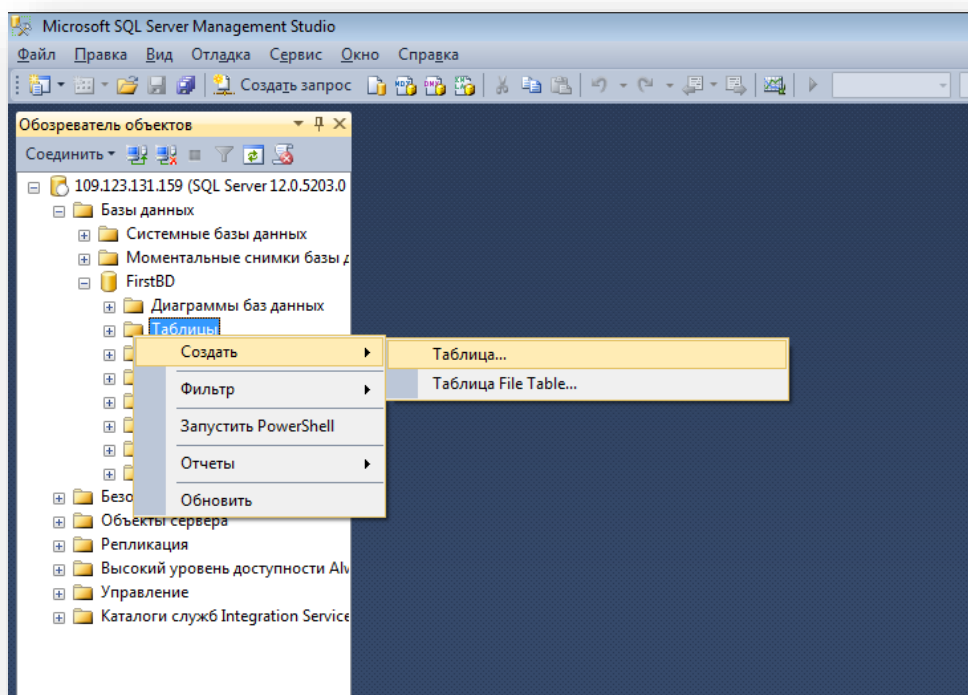


Рис. 2.9 – Обозреватель объектов

Создадим таблицу «Сотрудники» (из примера, рассматриваемого в лабораторной работе 1). Для этого необходимо кликнуть правой кнопкой мыши по папке «Таблицы» и в появившемся меню выбрать пункт «Создать таблицу». Появится окно создания новой таблицы (рисунок 2.10).

Имя столбца	Тип данных	Разрешит...
▶		<input type="checkbox"/>

Рисунок 2.10 – Создание новой таблицы

В правой части окна расположена таблица определения полей новой таблицы.

Данная таблица имеет следующие столбцы:

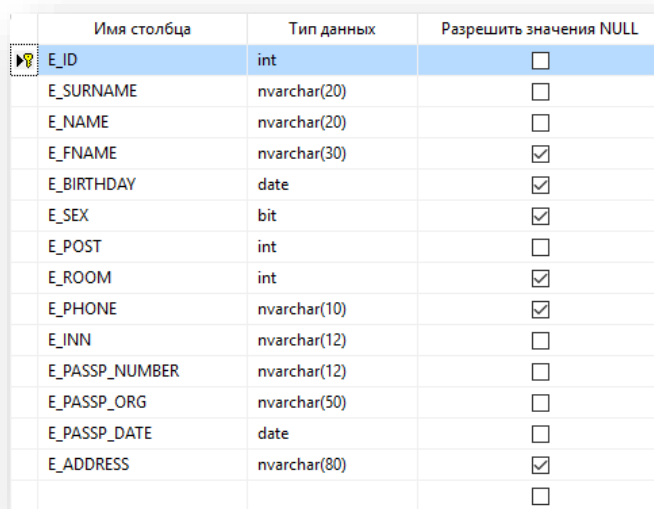
- Имя столбца должно всегда начинаться с буквы и не должно содержать различных специальных символов и знаков препинания. Если имя поля содержит пробелы, то оно автоматически заключается в квадратные скобки.
- Тип данных столбца.

– Разрешить значения Null. Если эта опция поля включена, то в случае не заполнения поля в него будет автоматически подставлено значение Null. То есть, поле обязательно для заполнения.

Под таблицей определения полей располагается таблица свойств выделенного поля «Свойства столбца». В данной таблице настраиваются свойства выделенного поля.

Некоторые из них будут рассмотрены ниже.

Перейдём к созданию полей и настройке их свойств. В таблице определения полей задайте значения столбцов «Имя столбца», «Тип данных» и «Разрешить значения Null», как показано на рисунке 2.11.



Имя столбца	Тип данных	Разрешить значения NULL
E_ID	int	<input type="checkbox"/>
E_SURNAME	nvarchar(20)	<input type="checkbox"/>
E_NAME	nvarchar(20)	<input type="checkbox"/>
E_FNAME	nvarchar(30)	<input checked="" type="checkbox"/>
E_BIRTHDAY	date	<input checked="" type="checkbox"/>
E_SEX	bit	<input checked="" type="checkbox"/>
E_POST	int	<input type="checkbox"/>
E_ROOM	int	<input checked="" type="checkbox"/>
E_PHONE	nvarchar(10)	<input checked="" type="checkbox"/>
E_INN	nvarchar(12)	<input type="checkbox"/>
E_PASSP_NUMBER	nvarchar(12)	<input type="checkbox"/>
E_PASSP_ORG	nvarchar(50)	<input type="checkbox"/>
E_PASSP_DATE	date	<input type="checkbox"/>
E_ADDRESS	nvarchar(80)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Рисунок 2.11 – Пример создания таблицы «Сотрудники»

Из рисунка 2.11 следует, что таблица Сотрудники (Employees) имеет следующий набор столбцов:

- E_ID –идентификатор сотрудника (первичный ключ, уникальный);
- E_SURNAME – фамилия сотрудника (обязательное, строка 20 символов);
- E_NAME – имя сотрудника (обязательное, строка 20 символов);
- E_FNAME – отчество (не обязательно, строка 30 символов);
- E_BIRTHDAY – дата рождения (дата);
- E_SEX – пол (логический тип данных);
- E_POST – идентификатор должности, служит для связи с таблицей «должности» (posts), в которой хранятся все должности (внешний ключ – связи между таблицами будут рассмотрены ниже);
- E_ROOM – номер комнаты (составной внешний ключ, служит для связи с таблицей «Комнаты»;

– E_TEL – номер телефона (составной внешний ключ, служит для связи с таблицей «Комнаты»;

- E_INN – идентификационный номер налогоплательщика;
- E_PASSP_NUMBER – номер паспорта (строка 12 символов, обязательное);
- E_PASSP_ORG – кем выдан паспорт(строка, 50 символов, обязательной);
- E_PASSP_DATE – дата выдачи паспорта (дата, обязательное);
- E_ADDRESS – адрес проживания (строка 80 символов).

Так как, поле E_ID будет являться первичным полем связи, то мы должны сделать его числовым счётчиком. То есть данное поле должно автоматически заполняться числовыми значениями. Более того, оно должно быть ключевым. Для этого выделите поле, просто щёлкнув по нему мышкой в таблице определения полей. В таблице свойств столбца отобразятся свойства поля E_ID. Разверните группу свойств *Спецификация идентификатора* Свойство (*Идентификатор*) установите в значение *Да*. Задайте свойства *Начальное значение идентификатора* и *Шаг приращения идентификатора* равными 1 (рисунок 2.12).

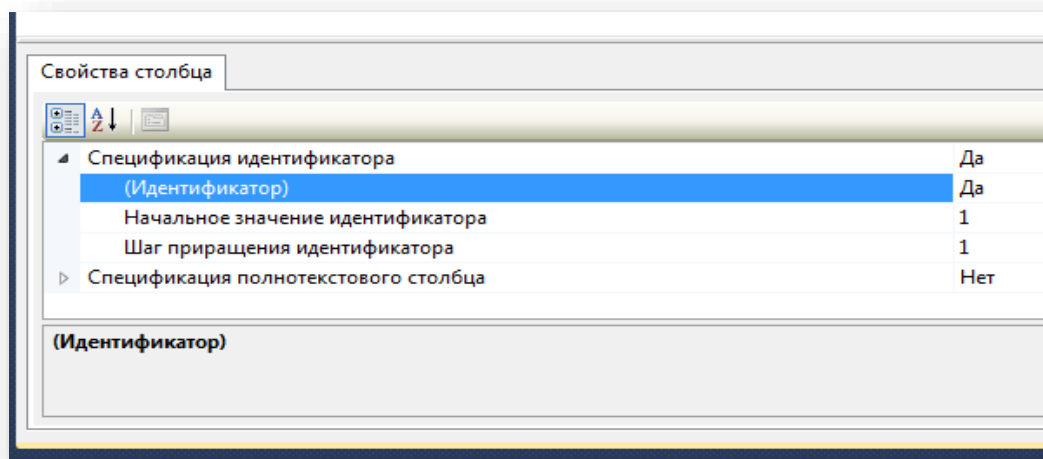



Рис. 2.12 – Свойства столбца E_ID


Эти настройки показывают, что значение поля E_ID у первой записи в таблице будет равным 1, у второй – 2, у третьей 3 и т.д.

Теперь сделаем поле E_ID ключевым полем. *Ключевое поле – это необходимый элемент реляционных баз данных, он позволяет однозначно идентифицировать запись в таблице, другими словами, позволяет присвоить каждой записи индивидуальный номер т.к. требуется, что бы все записи были уникальны.*

Для создания ключевого поля выделите поле, а затем на панели инструментов нажмите кнопку с изображением ключа . В таблице определения полей, рядом с

полем E_ID появится изображение ключа  SID, говорящее о том, что поле ключевое.

На этом настройку таблицы «Сотрудники» можно считать за данным этапом завершённой.

Закройте окно создания новой таблицы, нажав кнопку закрытия  в верхнем правом углу окна, над таблицей определения полей.

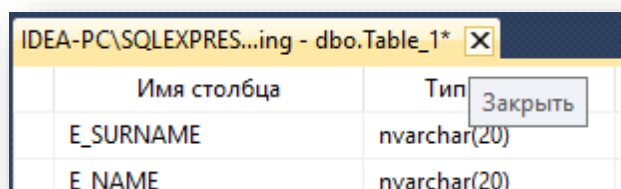


Рис. 2.13 – Заккрытие настройки таблицы

Появится окно с запросом о сохранении таблицы. В этом окне необходимо нажать Да. Появится окно Выбор имени, предназначенное для определения имени новой таблицы.

В этом окне задайте имя новой таблицы как «employees» и нажмите кнопку ОК. Таблица «employees» отобразится в обозревателе объектов в папке Таблицы базы данных «publishing».

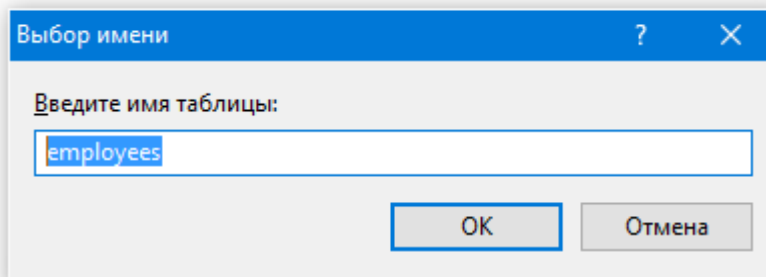


Рис. 2.14 – Ввод имени таблицы

В обозревателе объектов таблица «employees» отображается как dbo.employees. Префикс dbo обозначает, что таблица является объектом базы данных (Data Base Object). В дальнейшем при работе с объектами базы данных префикс dbo можно опускать.

2.1.2. Создание таблиц с помощью запроса

Для создания таблиц в SQL Server в первую очередь необходимо сделать активной ту БД, в которой создается таблица. Для этого можно в новом запросе набрать команду: `USE <Имя БД>`, либо на панели инструментов необходимо выбрать в выпадающем списке рабочую БД. После выбора БД можно создавать таблицы.

Таблицы создаются командой:

```
CREATE TABLE <Имя таблицы>
(
  <Имя поля1> <Тип1> IDENTITY(NULL|NOTNULL),
  <Имя поля1> <Тип1> NOT NULL,
  ...
);
```

Здесь:

- <Имя таблицы> – имя создаваемой таблицы;
- <Имя поля> – имена столбцов таблицы;
- <Тип> – типы полей;
- <IDENTITY NULL|NOT NULL> – поле счётчик.

Если имя поля содержит пробел, то оно заключается в квадратные скобки. Пример создание таблицы «employees» с помощью запроса:

```
CREATE TABLE employees(
  E_ID int IDENTITY(1,1) NOT NULL,
  E_SURNAME NVARCHAR(20) NOT NULL,
  E_NAME NVARCHAR (20) NOT NULL,
  E_FNAME NVARCHAR (30) NULL,
  E_BIRTHDAY DATE NULL,
  E_SEX BIT NULL,
  E_POST INT NULL,
  E_ROOM INT NULL,
  E_PHONE NVARCHAR (10) NULL,
  E_INN NVARCHAR (12) NOT NULL,
  E_PASSP_NUMBER NVARCHAR (12) NOT NULL,
  E_PASSP_ORG NVARCHAR (50) NOT NULL,
  E_PASSP_DATE DATE NOT NULL,
  E_ADDRESS NVARCHAR (80) NULL,
  CONSTRAINT PK_employees PRIMARY KEY E_ID
);
```

По аналогии создадим все таблицы, описанные в примере рассмотренном в первой лабораторной работе.

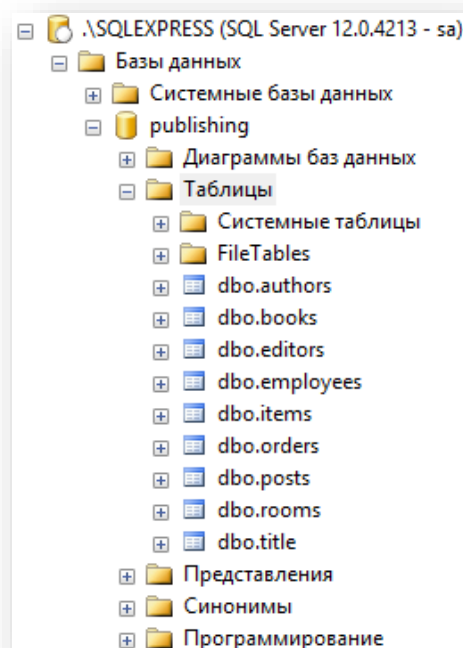


Рис. 2.15 – Результат создания всех таблиц

2.2. Связывание таблиц

2.2.1. Создание диаграммы базы данных

При работе БД должна обеспечиваться целостность данных. Под целостностью данных понимают обеспечения целостности связей между записями в таблицах при удалении записей из первичных таблиц. То есть, при удалении записей из первичных таблиц автоматически должны удаляться связанные с ними записи из вторичных таблиц.

В случае несоблюдения целостности данных со временем в БД накопится большое количество записей во вторичных таблицах, связанных с несуществующими записями в первичных таблицах, что приведёт к сбоям в работе БД и её засорению неиспользуемыми данными.

Внешний ключ – это столбец (или комбинация столбцов), совпадающий с первичным ключом не которой таблицы. Если значение внешнего ключа соответствует значению первичного ключа, становится понятно, что между объектами базы данных представленными совпадающими строками, существует логическое взаимоотношение.

Одним из главных ограничений отношения является ссылочная целостность, которая определяет, что каждое значение внешнего ключа, не являющееся NULL-значением, должно ссылаться (REFERENCES) на некоторое существующее значение первичного ключа.

Для обеспечения целостности данных в SQL Server используют диаграммы и триггеры. Диаграммы – это компоненты БД, которые блокируют удаление записей из

первичных таблиц если существуют связанные с ними записи во вторичных таблицах. Следовательно, диаграммы предотвращают нарушение целостности данных.

В БД Microsoft SQL Server 20XX все диаграммы находятся в папке Диаграммы базы данных обозревателя объектов.

Создадим диаграмму, обеспечивающую целостность данных нашей БД «publishing».

Для создания новой диаграммы в БД «publishing» щёлкните правой кнопкой мыши по папке Диаграммы базы данных и в появившемся меню выберем пункт Новая диаграмма базы данных. Сначала появится окно с вопросом о добавлении нового объекта Диаграмма. В этом окне нужно нажать кнопку Да. Затем появится окно «Добавление таблицы» предназначенное для добавления таблиц в новую диаграмму (рисунок 2.16).

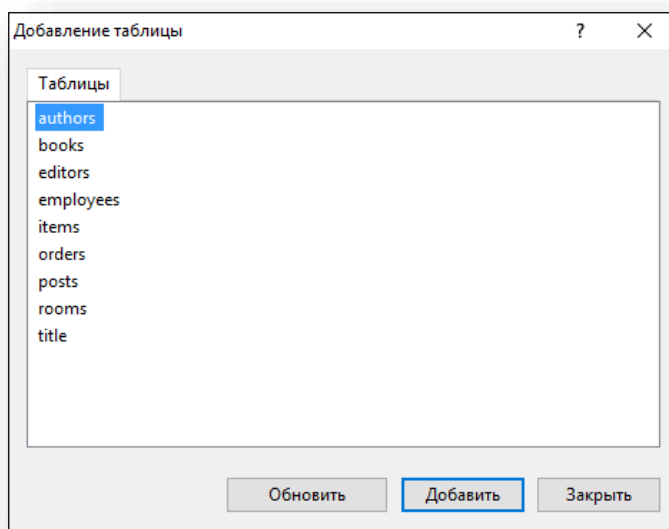


Рис. 2.16 – Окно Добавление таблицы

В окне добавления таблиц выделите все таблицы нашей БД и нажмите кнопку «Добавить». Закройте окно «Добавление таблицы» нажатием на кнопку «Закрыть». Появится окно диаграммы, где будут отображены отобранные таблицы. Обратите внимание, что на данном этапе таблицы являются не связанными, т.е. отсутствует ограничение целостности по внешнему ключу.

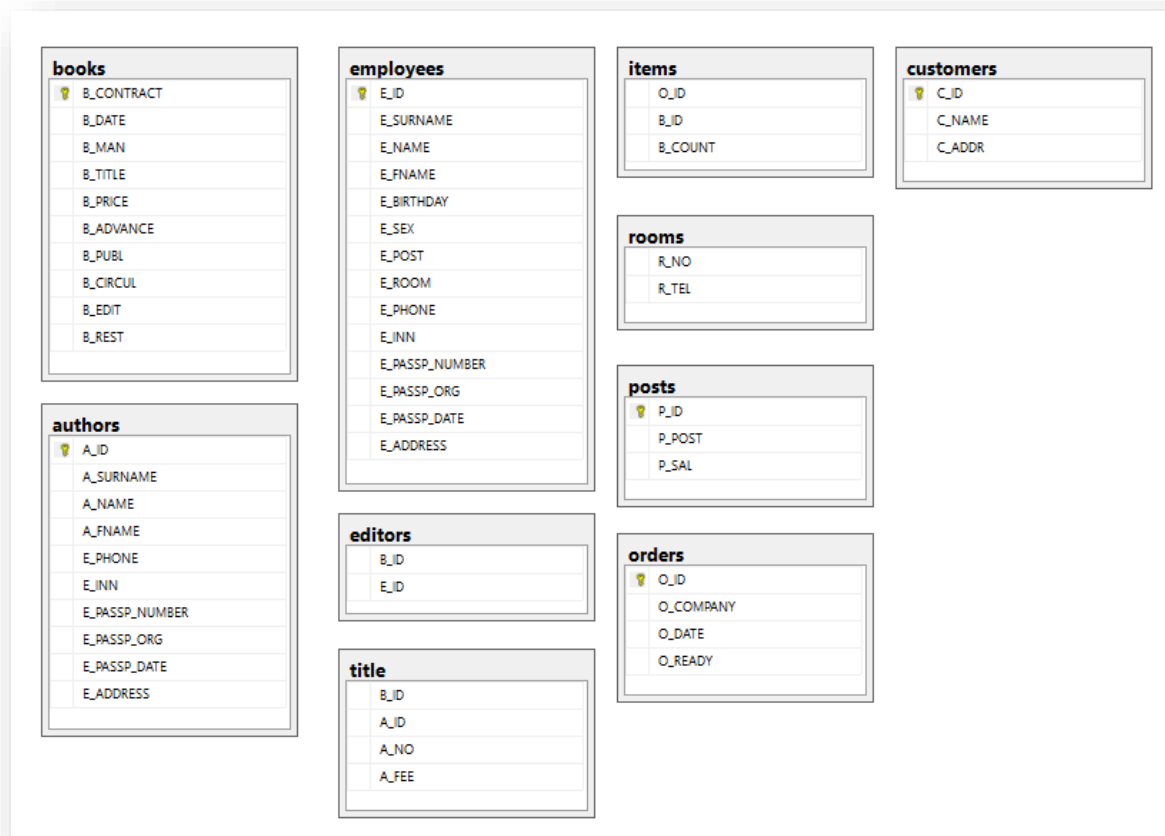


Рис. 2.17 – Сущности на диаграмме

Теперь необходимо определить связи между таблицами. Давайте организуем связь между сотрудником и должностью. Перед этим уточним, что в случае если мы захотим удалить из БД должность, то необходимо обнулить должность у сотрудника. Для этого необходимо перетащить поле P_ID из таблицы Posts на такое же поле в таблице Employees. Появится окно создания связи между таблицами «Таблицы и столбцы» (рисунок 2.18).

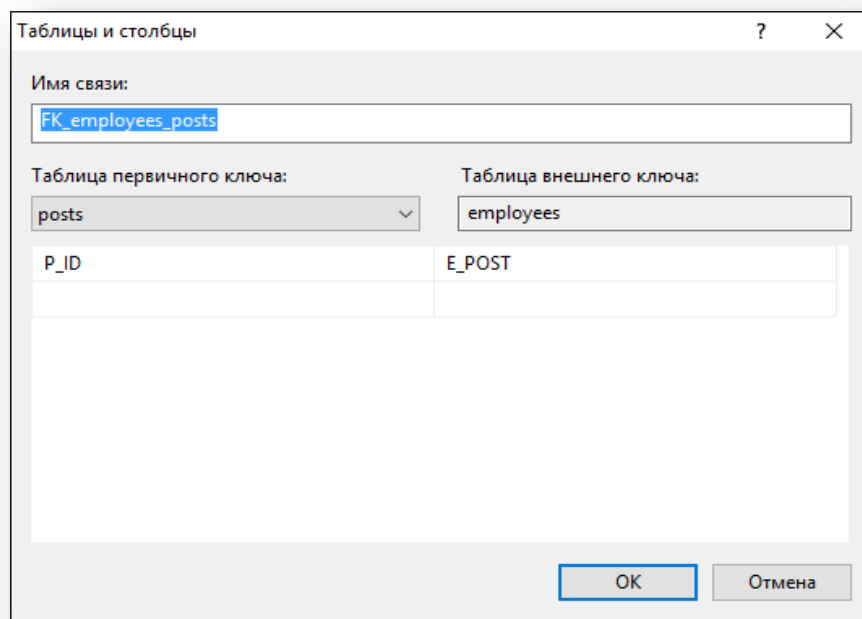


Рис. 2.18 – Окно Таблицы и столбцы

В окне создания связи нажмите кнопку ОК. Появится окно настройки свойств связи Связь по внешнему ключу (рисунок 2.19).

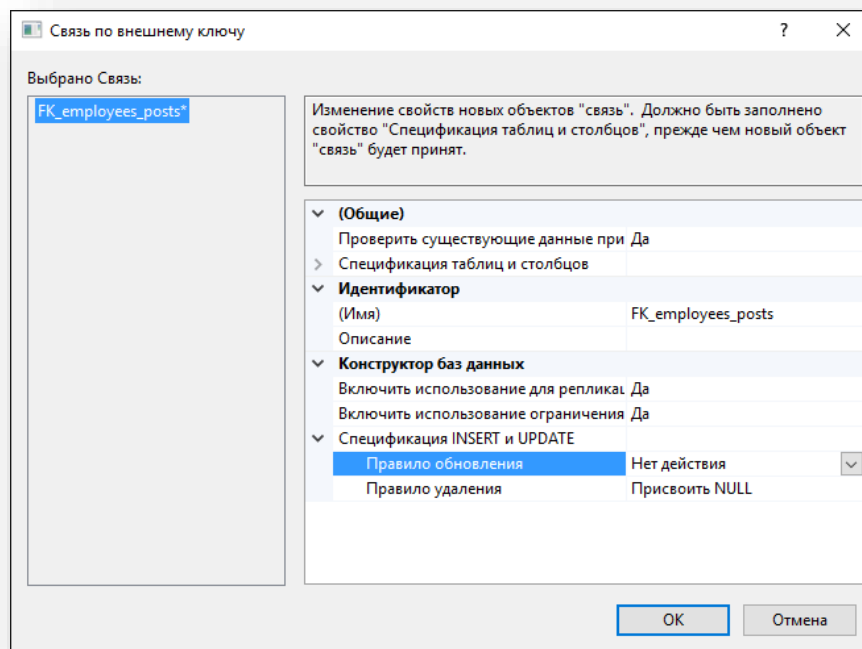


Рис. 2.19 – Окно Связь по внешнему ключу

Спецификация INSERT и UPDATE определяет, что происходит, если значение первичного ключа, на которое ссылается внешний ключ, удаляется или обновляется:

- установить NULL (SET NULL) – значение внешнего ключа заменяется NULL-значением. Это невозможно, когда внешний ключ является частью первичного ключа своей таблицы.
- установить значение по умолчанию (SET DEFAULT) – значение внешнего ключа заменяется значением столбца, используемым по умолчанию.
- каскадно (CASCADE) – удаляются или обновляются все строки внешнего ключа.
- нет действия (NO ACTION) – после обновления нельзя модифицировать значения

В диаграмме между таблицами employees и posts появится связь в виде линии (рисунок 2.20).

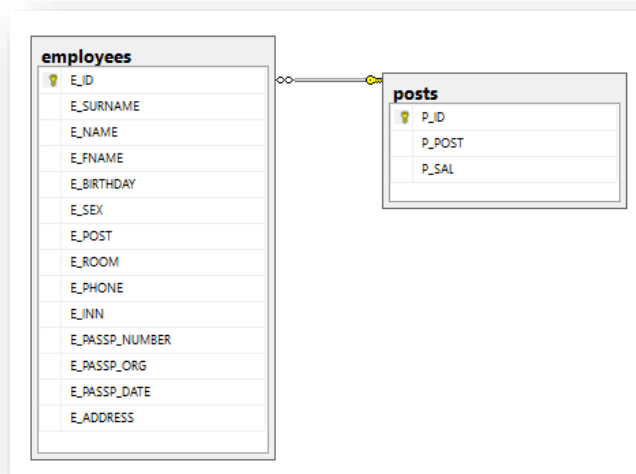


Рис. 2.20 – Связь между таблицами employees и posts

2.2.2. Связывание таблиц с помощью конструкции FOREIGN KEY

Для указания внешнего ключа таблицы нужно добавить конструкцию FOREIGN KEY в конструкцию CREATE TABLE:

```
FOREIGN KEY <имя внешнего ключа> REFERENCES <таблица первичного ключа>
ON UPDATE <операция обновления>
ON DELETE <операция удаления>
```

Имена столбцов внешних ключей указываются после ключевых слов FOREIGN KEY. В конструкции REFERENCES содержится имя таблицы того первичного ключа, на который производится ссылка. Если столбцы первичного ключа заданы в конструкции PRIMARY KEY своей таблицы, нет необходимости перечислять их имена. Если же имена

столбцов не являются частью конструкции PRIMARY KEY, необходимо перечислить столбцы первичного ключа в конструкции REFERENCES.

Тогда создание таблицы Employees с помощью запроса будет выглядеть следующим образом:

```
CREATE TABLE Employees
(
  E_ID INT IDENTITY (1, 1) NOT NULL,
  E_SURNAME NVARCHAR(20) NOT NULL,
  E_NAME NVARCHAR (20) NOT NULL,
  E_FNAME NVARCHAR (30) NULL,
  E_BIRTHDAY DATE NULL,
  E_SEX BIT NULL,
  E_POST INT NULL,
  E_ROOM INT NULL,
  E_PHONE NVARCHAR(10) NULL,
  E_INN NVARCHAR(12) NOT NULL,
  E_PASSP_NUMBER NVARCHAR (12) NOT NULL,
  E_PASSP_ORG NVARCHAR (50) NOT NULL,
  E_PASSP_DATE DATE NOT NULL,
  E_ADDRESS NVARCHAR (80) NULL,
  PRIMARY KEY E_ID,
  FOREIGN KEY (E_POST) REFERENCES
  Posts (P_ID)
);
```

FOREIGN KEY – команда, указывающая какой из полей у нас будет внешним ключом.

REFERENCES – указывает на какое поле в соединяемой таблице у нас будет ссылаться внешний ключ (после данной команды указывается название таблицы, а в скобках название поля)

ON UPDATE и **ON DELETE** – указывают какие действия будут выполняться при изменении родительского ключа или удалении строк родительской таблицы соответственно. **NO ACTION** означает, что когда родительский ключ изменяется или удаляется из БД, то никаких специальных действий не производится. **SET NULL** означает,

что при удалении родительского ключа (для ON DELETE SET NULL) или его изменении (для ON UPDATE SET NULL) столбцы дочернего ключа будут устанавливаться в значение NULL во всех строках дочерней таблицы, которые ссылаются на удаляемую/изменяемую строку родительской таблицы.

Если таблица уже создана для добавления в нее внешнего ключа необходимо воспользоваться конструкцией ALTER TABLE.

Примечание: как говорилось ранее, команды CREATE, DROP, ALTER относятся к DDL.

Добавление внешнего ключа с помощью конструкции ALTER TABLE будет выглядеть следующим образом:

ALTER TABLE Employees

ADD FOREIGN KEY (E_POST)

REFERENCES

Posts

(P_ID)

Аналогично для всех остальных таблиц необходимо проделать те же самые действия. В итоге схема отношений будет выглядеть следующим образом.

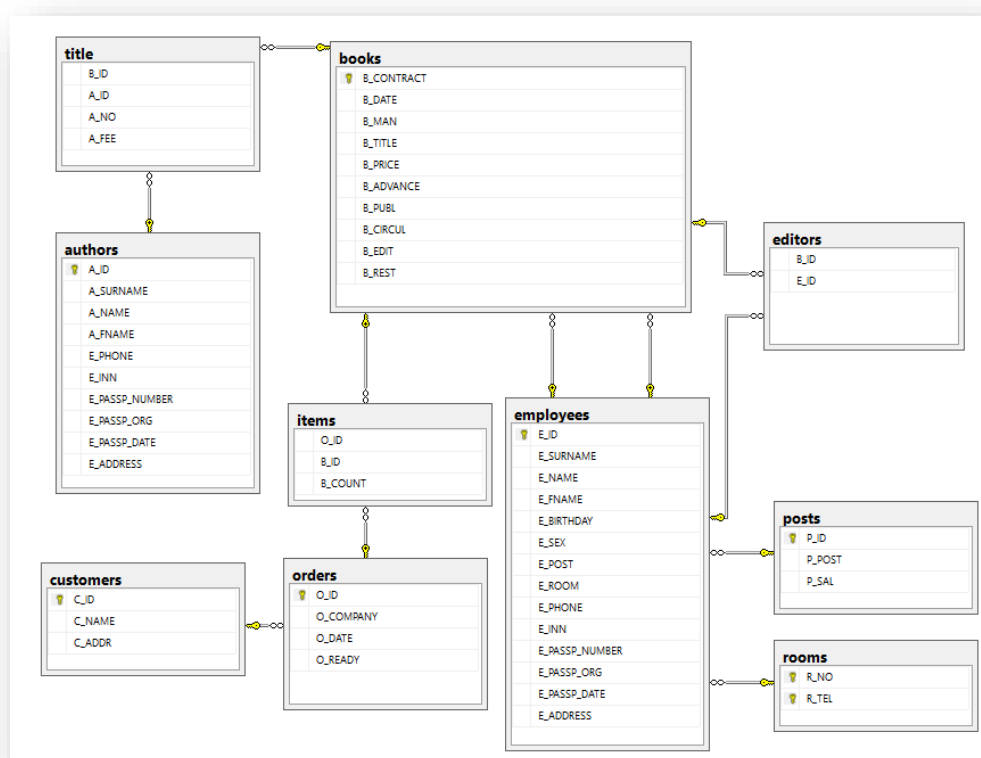


Рис. 2.21 – Схема базы данных рассматриваемого примера

3. ЯЗЫК ЗАПРОСОВ SQL

Запрос в языке *SQL* состоит из одного или нескольких операторов, следующих один за другим и разделенных точкой с запятой. Каждая последовательность операторов языка *SQL* реализует определенное действие над БД. Оно осуществляется за несколько шагов, на каждом из которых над таблицами выполняются определенные действия.

3.1. Работа с данными

Напомним, что язык *SQL* выходит в состав 4 семейств языков и выполняет их роли (Рисунок 1).

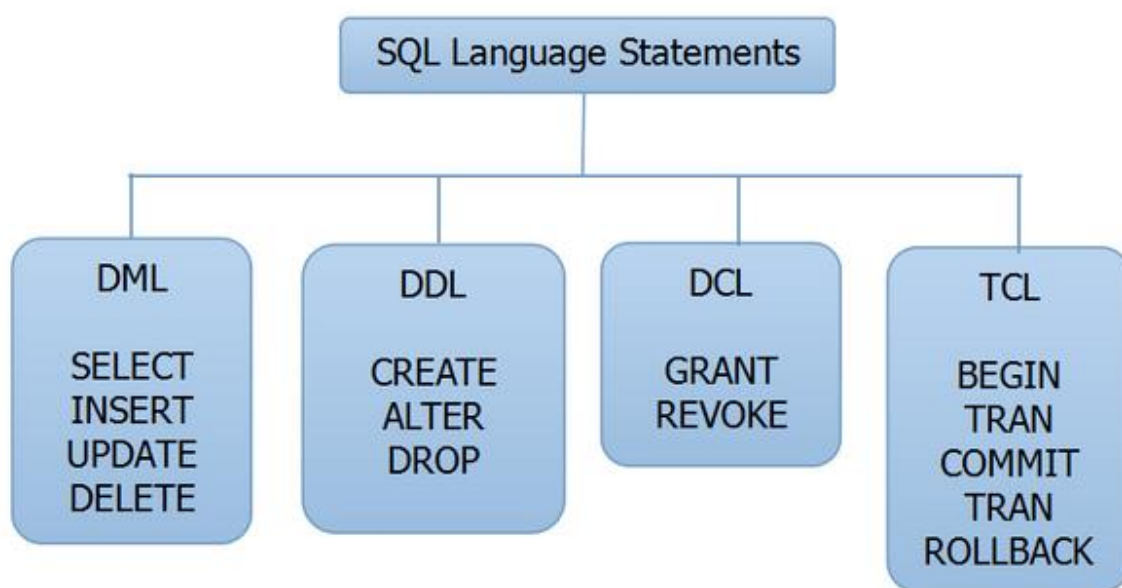


Рисунок 1 - Семейства языков

SQL является языком DML (Data Manipulation Language – язык манипулирования данными) так как реализует следующие функции:

- **SELECT** – извлечение данных из БД;
- **UPDATE** – обновление данных в БД;
- **DELETE** – удаление данных из БД;
- **INSERT** – вставка данных в БД;
- **MERGE** – вставка и обновление;
- **CALL** – вызов PL/SQL или Java подпрограммы;
- **EXPLAIN PLAN** – создание планов выполнения запросов;
- **LOCK TABLE** – для блокирования таблиц.

Каждый оператор *SQL* начинается с ключевого слова, которое определяет, что делает этот оператор (**SELECT**, **INSERT**, **DELETE**, **UPDATE**).

В операторе содержатся предложения, содержащие сведения о том, над какими данными производятся операции. Каждое предложение начинается с ключевого слова, такого как FROM, WHERE и др.

Структура предложения зависит от его типа: ряд предложений содержит имена полей или таблиц, некоторые могут включать дополнительные ключевые слова, константы или выражения. Для примера приведем структуру запроса SELECT на рисунке 2.



Рисунок 2 – структура запроса SELECT

Так как после создания таблиц БД все еще остается пустой в нее необходимо добавить данные. Данные в таблицы добавляются построчно, где каждая строка описывает новый уникальный объект системы (например: сотрудник, книга и т.д.) Для этого рассмотрим операцию заполнения таблиц начальными данными.

Для примера работы с данными заполним таблицу Employees. Для заполнения этой таблицы в обозревателе объектов щёлкните правой кнопкой мыши по таблице Employees и в появившемся меню выберите пункт «Изменить первые 200 строк». В рабочей области Microsoft SQL Server Management Studio проявится окно заполнения таблиц. Таблицы заполняются аналогично Microsoft Excel.

3.1.1. Добавление данных в таблицу

В SQL Server 20XX заполнение таблиц производится при помощи следующей команды:

```
INSERT <Имя таблицы> (<Список полей>)  
VALUES (<Значения полей>)
```

–<Имя таблицы> – таблица, куда вводим данные;

–<Список полей> – список полей, куда вводим данные, если не указываем, то подразумевается заполнение всех полей, в списке полей поля указываются через запятую;

–<Значения полей> – значения полей через запятую

Пример:

```
INSERT INTO Posts (P_POST, P_SAL)
VALUES ('Менеджер', 15000)
```

Или для добавления множества записей можно использовать конструкция из следующего примера:

```
INSERT INTO Posts VALUES
('Менеджер', 15000),
('Редактор', 20000),
('Программист', 50000),
('Главный редактор', 150000)
```

Примечание: Строковые данные, как и даты в запросах указываются в кавычках, числовые данные указываются без кавычек.

VALUES позволяет использовать один или несколько списков вставляемых значений данных. Список значений должен быть заключен в скобки.

Если нет необходимости заполнять все поля, то можно указать только те имена полей, значения которых необходимо изменить. В перечислении можно указывать поля в любом порядке, но в списке VALUES значения должны идти в том же порядке, в котором перечислены поля.

```
INSERT INTO tbPeoples (vcSurname, idPosition, vcName)
VALUES ('Смирнов', 12, 'Сергей')
```

В качестве значений можно указать константу DEFAULT, то есть будет поставлено значение по умолчанию, либо можно подставить оператор SELECT.

Также возможно добавить множество записей из текстового файла. Для этого создайте текстовый файл с именем (например: title.txt), содержащий по одной записи в каждой строке (значения столбцов должны быть разделены символами табуляции (клавиша TAB) и даны в том порядке, который был определен командой CREATE TABLE). Незаполненным полям можно присвоить значение NULL. В текстовом файле это значение представляется символами \N.

Загрузить файл title.txt в таблицу можно с помощью следующей команды:

```
LOAD DATA LOCAL INFILE "title.txt" INTO TABLE title;
```

3.1.2. Удаление отдельных столбцов и отдельных строк из таблицы

Из таблицы можно удалить все столбцы, либо отдельные записи. Это осуществляется командой

```
DELETE <Имя таблицы>  
WHERE <Условие>
```

Примечание: **WHERE** – означает условие, т.е. в данном случае мы удаляем данные, если выполняется условие, которое находится после ключевого слова **WHERE**.

Минимальная команда для удаления выглядит следующим образом:

```
DELETE <Имя таблицы>
```

например,

```
DELETE tbPeoples
```

Эта команда удаляет все строки из таблицы tbPeoples.

Либо для удаления таблицы полностью:

```
DROP TABLE <Имя таблицы>
```

Для полной очистки таблицы и сброса счетчика AUTOINCREMENT используется конструкция

```
TRUNCATE TABLE <Имя таблицы>
```

Пример использования этой конструкции представлен ниже:

```
TRUNCATE TABLE Posts;
```

Инструкция TRUNCATE TABLE выполняется быстрее, чем инструкция DELETE, и использует меньше системных ресурсов и ресурсов журнала транзакций.

Если условие указано, то удаляются записи поля, которые соответствуют условию.

Ключевое слово TOP задает количество или процент удаляемых случайных строк. Если с инструкцией DELETE применяется предложение TOP (n), то операция удаления производится над n случайно выбранных строк.

Пример: Удалить 20 случайных строк из таблицы PurchaseOrderDetail, имеющих дату ранее 1 июля 2006 г.

```
DELETE TOP (20)  
FROM Purchasing.PurchaseOrderDetail  
WHERE DueDate < '20060701';
```

Пример: Удалить записи из таблицы Posts, у которых значение поля P_SAL > 100000.

```
DELETE Posts  
WHERE P_SAL > 100000
```

Пример: Удалить записи из таблицы Posts, у которых значение поля P_Post будет содержать часть слова «джер»

```
DELETE FROM Posts WHERE P_Post LIKE '%джер';
```

Пример: Удалить записи из таблицы Posts, у которых значение поля P_ID будет между 5 и 8.

```
DELETE FROM Posts WHERE P_ID BETWEEN 5 AND 8;
```

3.1.3. Изменение данных в таблице

Для этого используется следующая команда:

```
UPDATE <Имя таблицы>
SET
<Имя поля1> = <Выражение1>,
[<Имя поля2> = <Выражение2>,,]
...
[WHERE <Условие>]
```

—<Имя поля1>, <Имя поля2> - имена изменяемых полей,

—<Выражение1>, <Выражение2> - либо конкретные значения, либо NULL, либо операторы SELECT. Здесь SELECT применяется как функция.

—<Условие> – условие, которым должны соответствовать записи, поля которых изменяем.

Пример: В таблице posts переименовать «Менеджера» в «Редактора» название

```
UPDATE Pposts
SET P_POST = 'Редактор'
WHERE P_POST = 'Менеджер'
```

Здесь командой SET в поле P_POST устанавливаем значение «Редактор» там, где по условию WHERE в этом поле значение равно «Менеджер».

Пример: Увеличить значение даты у всех строк на 1

```
UPDATE peaples
SET dayBirthday = dayBirthday + 1
```

Из примера видно, что оператор UPDATE позволяет использовать математические вычисления.

ЗАКЛЮЧЕНИЕ

В данной лабораторной работе были даны теоретические знания по работе с SQL Server Management Studio, созданию баз данных, таблиц и запросов. Данная лабораторная работа была продолжением лабораторной работы №1, где по ранее спроектированной модели была создана БД уже физически и произведена работа с данными с помощью запросов.