Министерство образования и науки

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет инфокоммуникационных технологий

Отчет

по лабораторным работам

по дисциплине **«Базы данных»**

Выполнил: **студент группы K3220 И.О. Мороз**

Проверил: **ст. преподаватель И.С. Осетрова**

Санкт-Петербург

2020

# «Создание учебной базы данных»

## Цель работы.

Создание учебной базы данных с использованием SQL запросов. Создание сценария базы данных.

## Задачи, решаемые при выполнении работы.

### Подключение к Microsoft SQL Server.

### Создание учебной базы данных.

### Создание сценария базы данных.

### Удаление учебной базы данных.

### Создание учебной базы данных в Query Editor.

## Объект исследования.

Microsoft SQL Server, учебная база данных.

## Исходные данные.

### Docker-контейнер с SQL Server, DataGrip IDE.

### Методическое пособие.

### Методическое пособие.

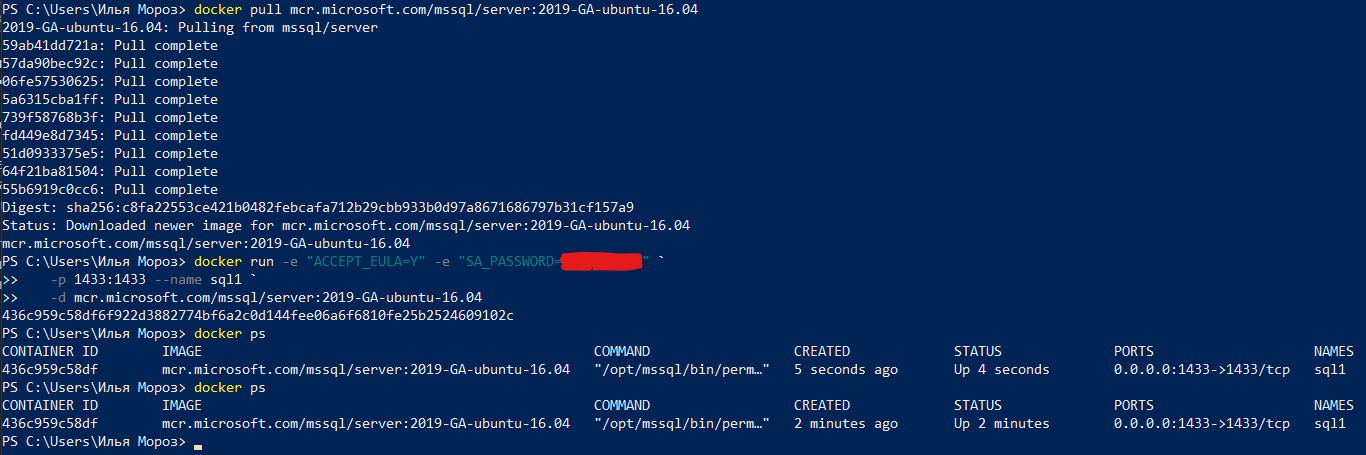
### Методическое пособие.

### Методическое пособие.

## Выполнение работы.

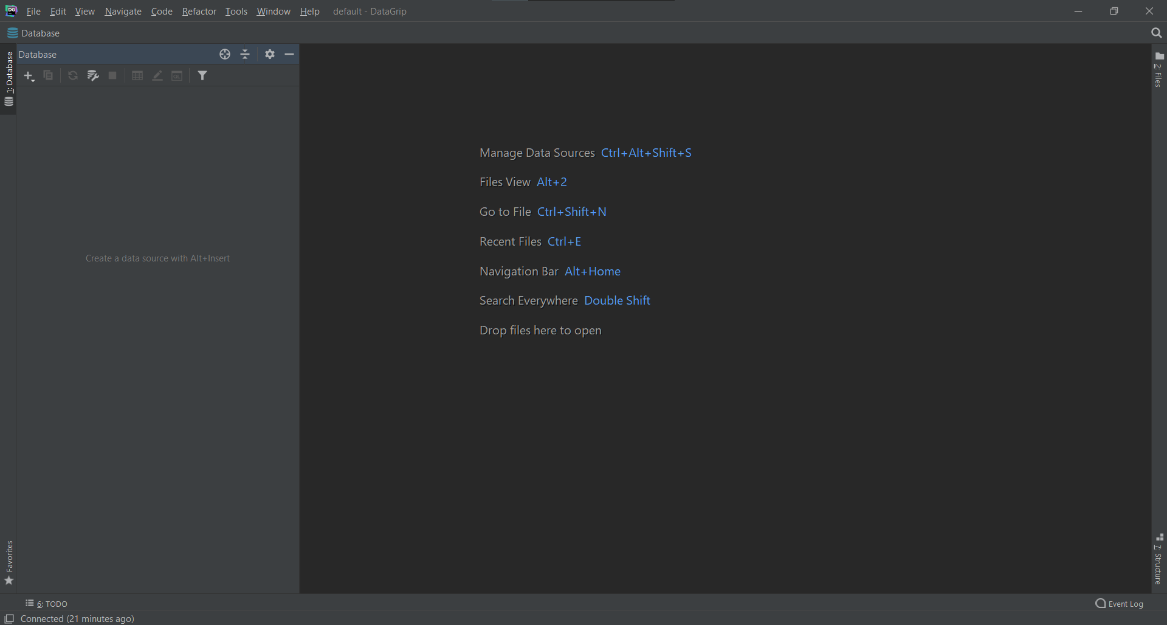
### Подключение к Microsoft SQL Server.

Перед выполнением задания было необходимо поднять SQL Server в локальной сети и подключиться к ней. Для упрощения процесса поднятия SQL Server был использован Docker контейнер от Microsoft (подробнее см. [Краткое руководство. Запуск образов контейнеров SQL Server в Docker](https://docs.microsoft.com/ru-ru/sql/linux/quickstart-install-connect-docker?view=sql-server-ver15&pivots=cs1-cmd))

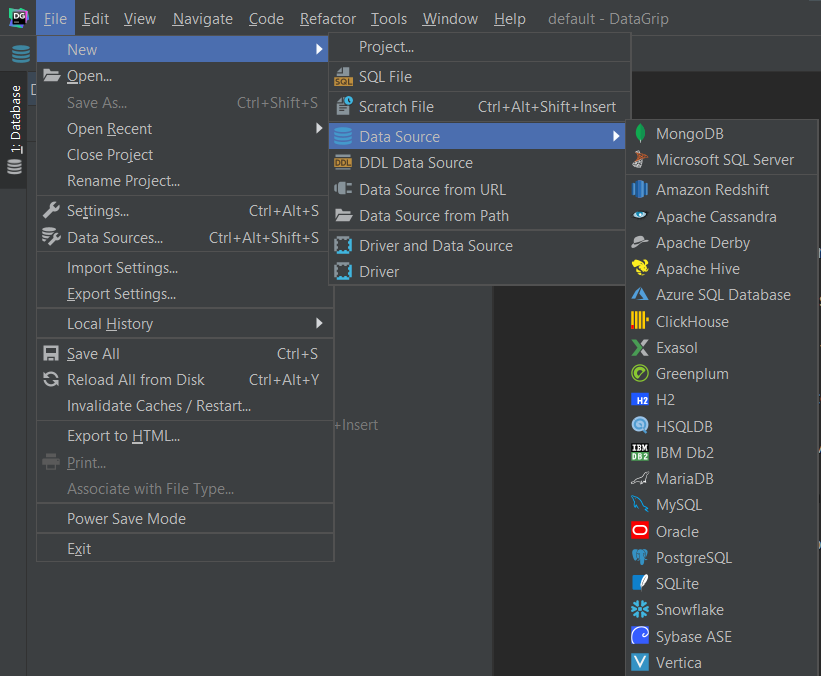


*Рисунок 1 – Запуск Docker контейнера с SQL Server*

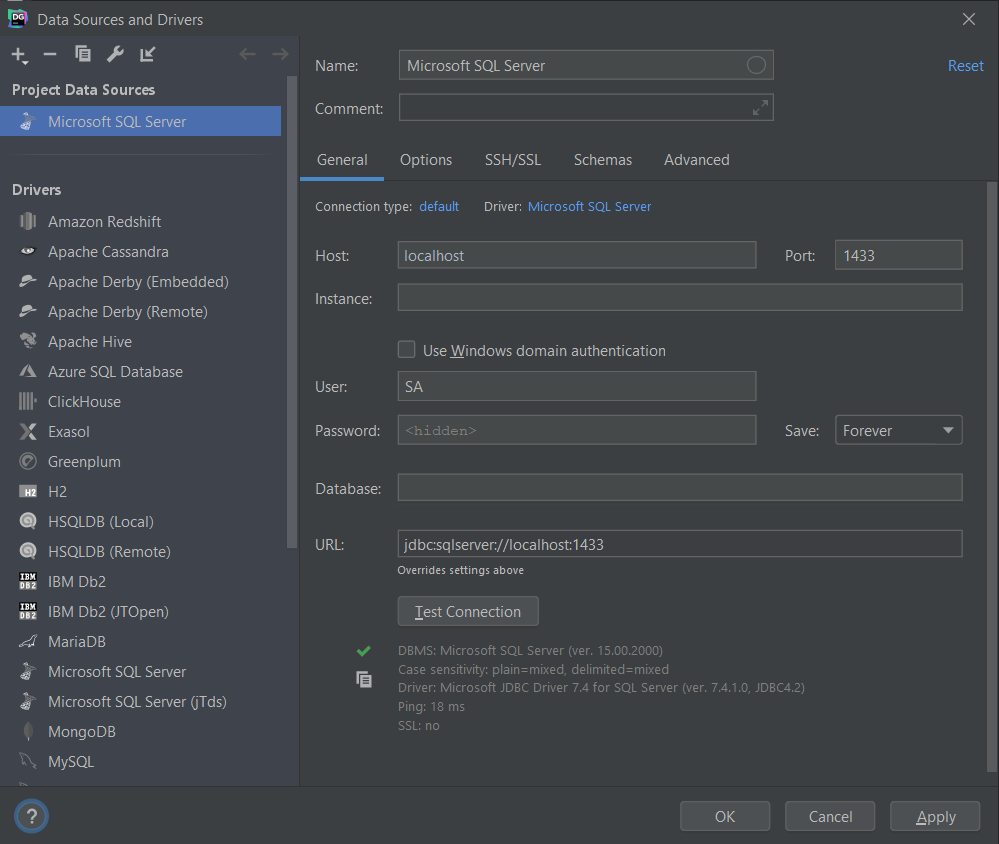
После запуска SQL Server подключение было выполнено с использованием DataGrip:



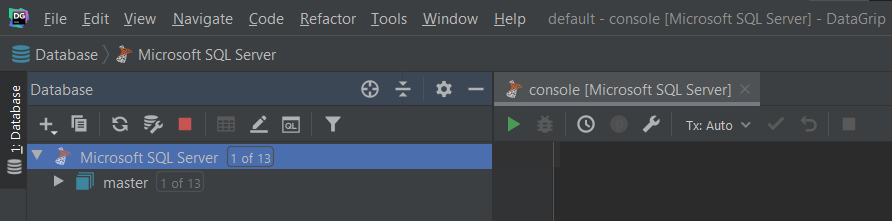
*Рисунок 2 – Окно DataGrip IDE*



*Рисунок 3 – Создание нового Data Source. File – New – Data Source – Microsoft SQL Server*



*Рисунок 4 – Настройка нового подключения*



*Рисунок 5 – Успешное подключение к SQL Server*

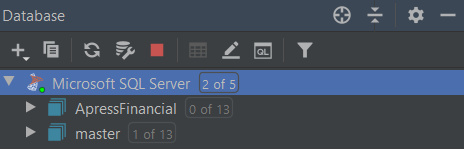
### Создание учебной базы данных.

С помощью SQL запроса была создана новая база ApressFinancial. При создании базы данных был создан вторичный файл данных с логическим именем «ApressFinancial\_act»

CREATE DATABASE ApressFinancial  
ON  
 PRIMARY  
 ( NAME = N'ApressFinancial',  
 FILENAME = N'/var/opt/mssql/data/ApressFinancial.mdf',  
 SIZE = 8192KB,  
 MAXSIZE = UNLIMITED,  
 FILEGROWTH = 1024KB ),  
 FILEGROUP [SECONDARY] DEFAULT  
 ( NAME = N'ApressFinancial\_act',  
 FILENAME = N'/var/opt/mssql/data/ApressFinancial\_act.ndf',  
 SIZE = 8192KB,  
 MAXSIZE = UNLIMITED,  
 FILEGROWTH = 1024KB )  
LOG ON  
 ( NAME = N'ApressFinancial\_log',  
 FILENAME = N'/var/opt/mssql/data/ApressFinancial\_log.ldf',  
 SIZE = 1024KB,  
 MAXSIZE = 2048KB,  
 FILEGROWTH = 10% )  
GO

*Листинг 1 – SQL запрос для создания базы данных «ApressFinancial»*

После выполнения данного SQL запроса была создана база данных.



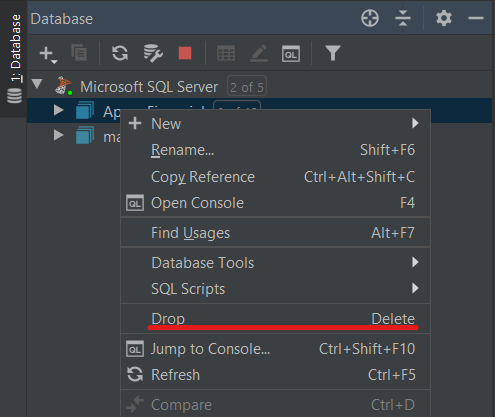
*Рисунок 6 – База данных создана*

### Создание сценария базы данных.

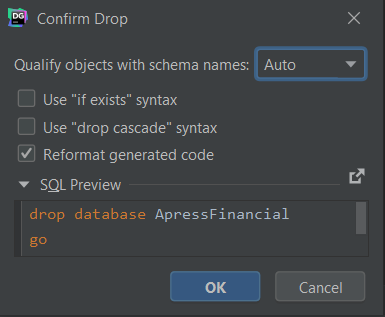
Так как DataGrip IDE поддерживает создание базы данных только с помощью SQL запроса, в предыдущем задании был создан файл сценария базы данных *create\_database.sql*. Листинг этого файла прикреплён в задании 1 (см. Листинг 1).

### Удаление учебной базы данных.

В данном задании было необходимо удалить только что созданную базу данных ApressFinancial.

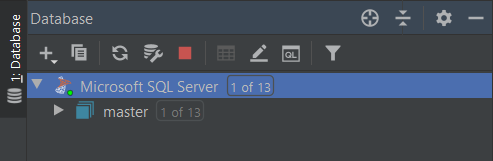


*Рисунок 7 – Команда для удаления базы данных в контекстном меню*



*Рисунок 8 – Подтверждение удаления базы данных*

База данных успешно удалена.

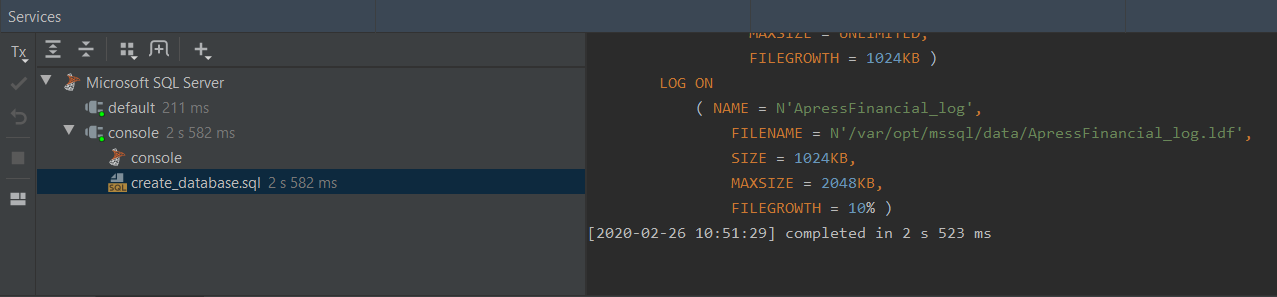


*Рисунок 9 – Структура Microsoft SQL Server после удаления базы данных*

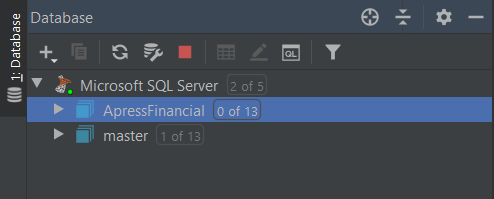
### Создание учебной базы данных в Query Editor.

Для создания учебной базы данных с помощью SQL запроса был открыт уже созданный файл *create\_database.sql*.

Файл был выполнен.



*Рисунок 10 – Результат выполнения файла*

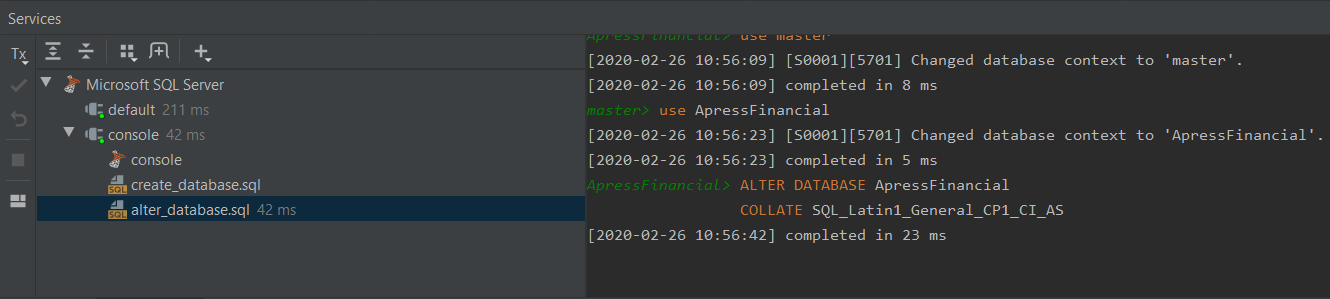


*Рисунок 11 – База данных вновь создана*

Для изменения параметров сортировки для символьных строк был создан файл SQL инструкций *alter\_database.sql*.

ALTER DATABASE ApressFinancial  
COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS  
GO

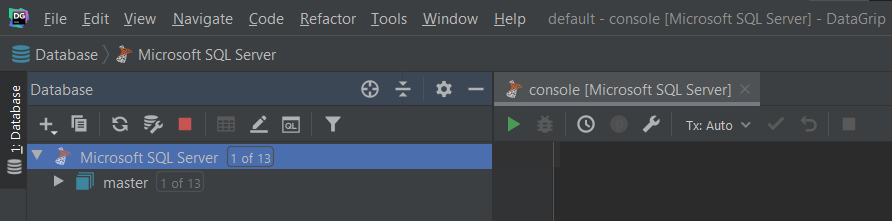
*Листинг 2 – Изменение параметров сортировки для символьных строк*



*Рисунок 12 – Результат выполнения файла*

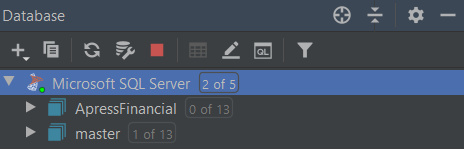
## Окончательные результаты.

### Подключение к базе данных было выполнено.



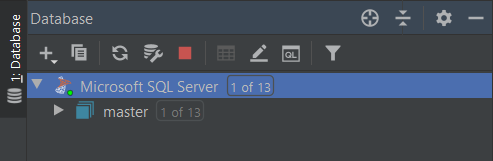
*Рисунок 13 – Успешное подключение к SQL Server*

### Была создана учебная база данных.



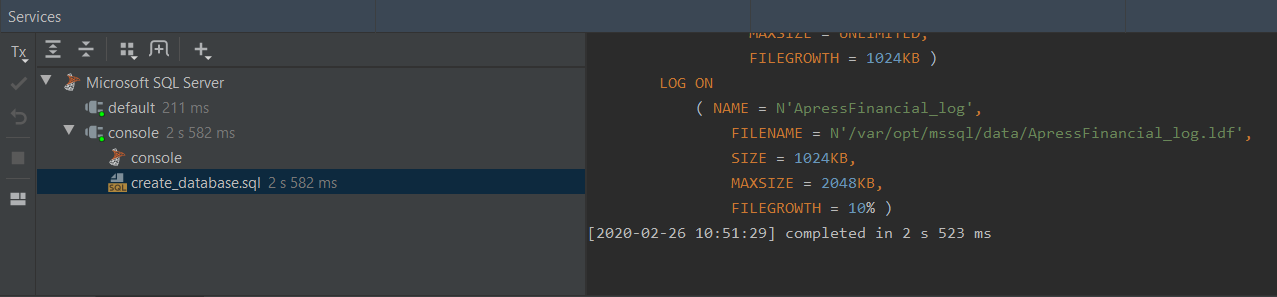
*Рисунок 14 – База данных создана*

### Учебная база данных была удалена с использованием контекстного меню.

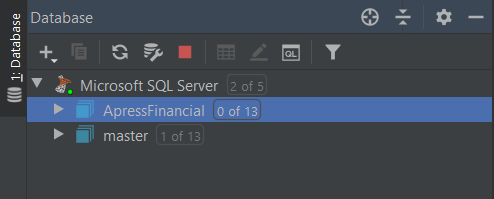


*Рисунок 15 – Структура Microsoft SQL Server после удаления базы данных*

### С помощью Query Editor был создан новый экземпляр учебной базы данных.



*Рисунок 16 – Результат выполнения файла*



*Рисунок 17 – База данных вновь создана*

## Выводы и анализ результатов работы.

В ходе данной лабораторной работы локально был запущен Microsoft SQL Server. При запуске использовался Docker контейнер. К созданному SQL Server было выполнено подключение с использованием DataGrip IDE.

При выполнении заданий были созданы файлы SQL сценариев *create\_database.sql* и *alter\_database.sql*. При выполнении этих файлов была создана учебная база данных ApressFinancial. Для данной базы данных были изменены параметры сортировки для символьных строк.

# «Безопасность: создание схем»

## Цель работы.

Создание схем для базы данных с использованием контекстного меню и Query Editor.

## Задачи, решаемые при выполнении работы.

### Создание новой схемы с использованием контекстного меню.

### Создание новой схемы с использованием Query Editor.

## Объект исследования.

Microsoft SQL Server, учебная база данных, схемы базы данных.

## Исходные данные.

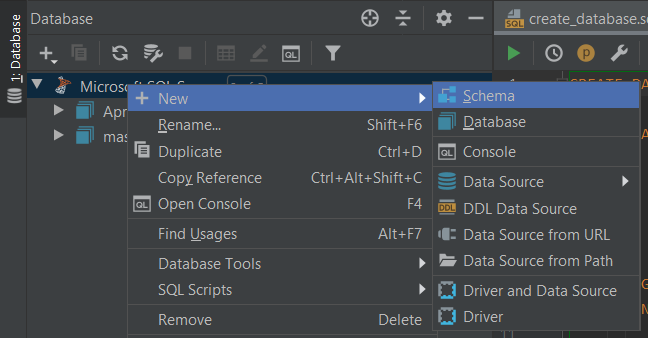
### Методическое пособие, учебная база данных.

### Методическое пособие, учебная база данных.

## Выполнение работы.

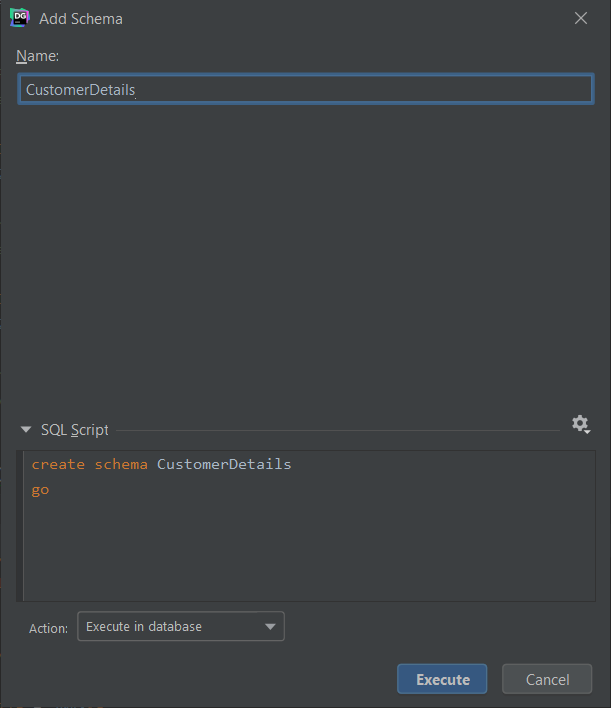
### Создание схемы в DataGrip IDE.

Для создания новой схемы необходимо нажать ПКМ на Data source, выбрать New – Schema.



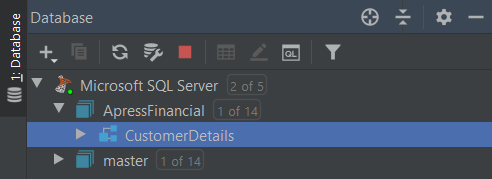
*Рисунок 18 – Контекстное меню создания новой схемы*

В появившемся окне было введено имя схемы «CustomerDetails».



*Рисунок 19 – Окно создания новой схемы*

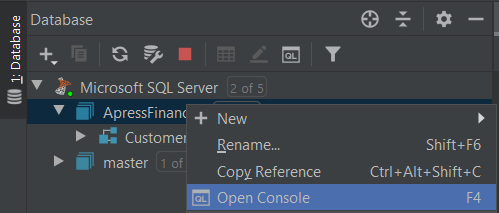
После создания новой схемы она отобразилась у базы данных ApressFinancial.



*Рисунок 20 – Только что созданная схема CustomerDetails*

### Создание схемы с помощью SQL запроса.

Для создания схемы с помощью SQL запроса необходимо открыть консоль для БД ApressFinancial.



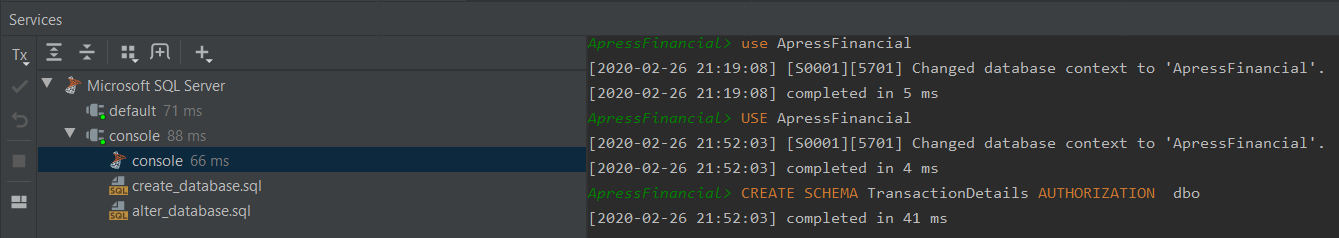
*Рисунок 21 – Открытие консоли*

В консоли был введён следующий SQL запрос.

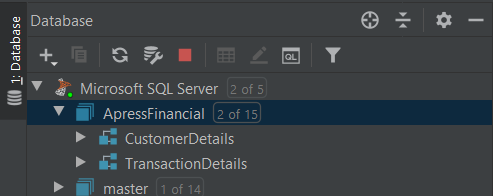
USE ApressFinancial  
GO  
  
CREATE SCHEMA TransactionDetails AUTHORIZATION dbo  
GO

*Листинг 3 – Создание схемы TransactionDetails*

Данный запрос был успешно выполнен.



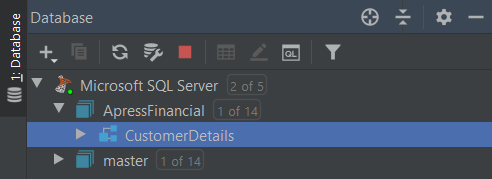
*Рисунок 22 – Успешное выполнение SQL запроса*



*Рисунок 23 – Схема появилась у базы данных*

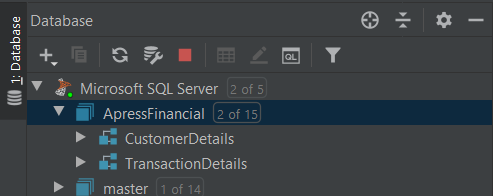
## Окончательные результаты.

### С помощью контекстного меню схема была создана.



*Рисунок 24 – Только что созданная схема CustomerDetails*

### С помощью SQL запроса схема была создана.



*Рисунок 25 – Схема появилась у базы данных*

## Выводы и анализ результатов работы.

В ходе данной лабораторной работы локально были созданы схемы базы данных с помощью контекстного меню DataGrip IDE и с помощью SQL запроса. В результате был получен файл с запросом SQL *create\_schema.sql*.

# «Проектирование и создание Таблиц»

## Цель работы.

Создание и проектирование таблиц в учебной базе данных с помощью контекстного меню и SQL запросов.

## Задачи, решаемые при выполнении работы.

### Создание таблицы с использованием контекстного меню.

### Создание таблицы в Query Editor.

### Создание таблицы с помощью шаблона.

### Изменение таблицы.

### Создание связи с помощью контекстного меню.

### Создание связи с помощью SQL запроса.

## Объект исследования.

Microsoft SQL Server, учебная база данных, таблицы базы данных.

## Исходные данные.

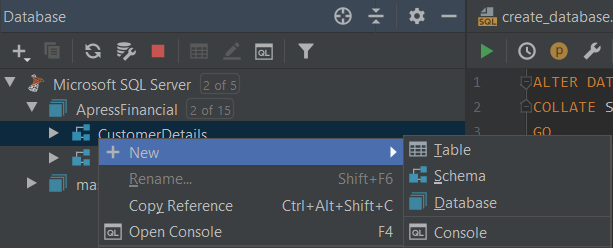
### Методическое пособие.

### Учебная база данных.

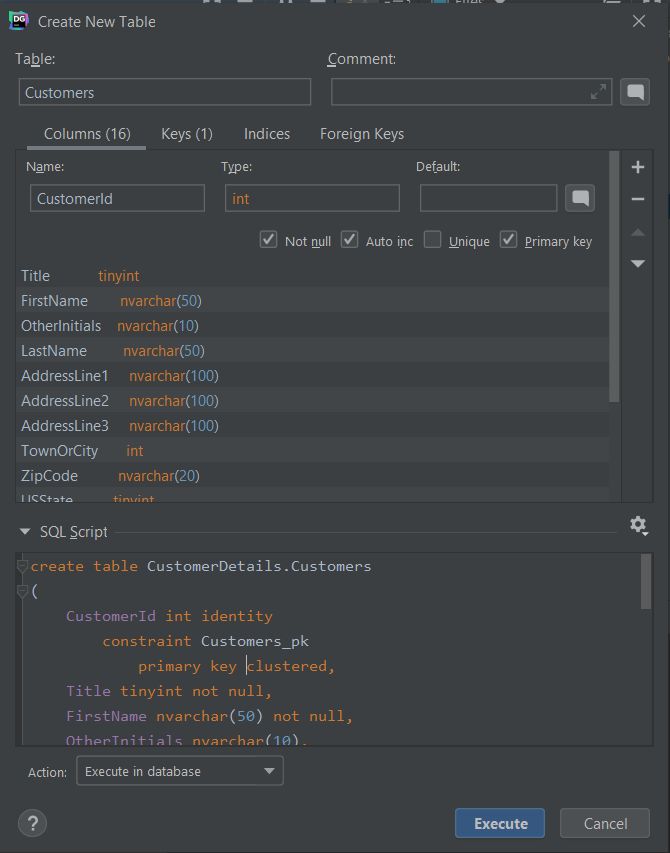
## Выполнение работы.

### Создание таблицы с использованием контекстного меню в DataGrip IDE.

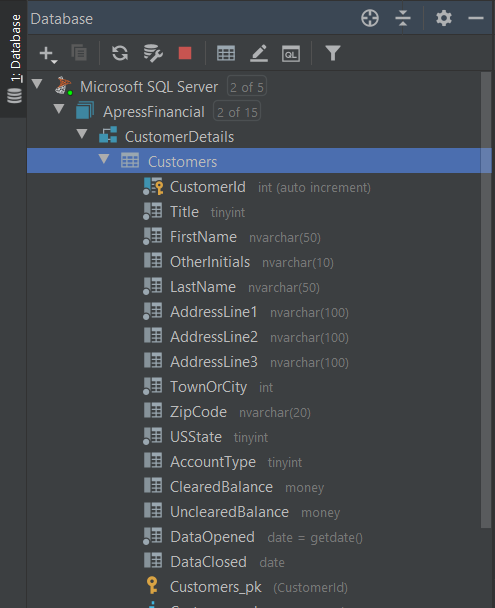
Для создания таблицы в DataGrip IDE необходимо создать схему. Так как создаётся таблица *Customers*, была использована уже существующая схема *CustomerDetails*.



*Рисунок 26 – CustomerDetails – ПКМ – New – Table*



*Рисунок 27 – Окно создания новой таблицы Customers*

**

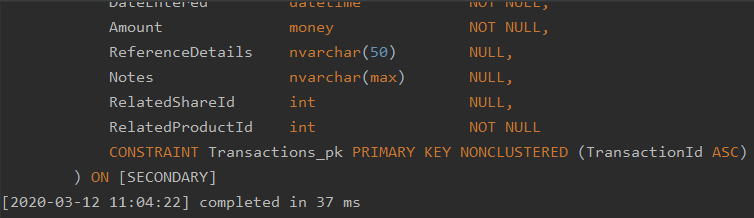
*Рисунок 28 – Таблица успешно создана*

### Создание таблицы с помощью Query Editor

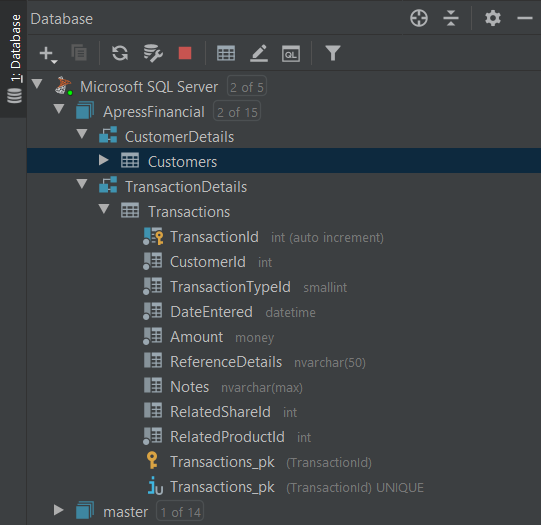
Был создан файл *create\_transactions\_table.sql* со следующим содержимым.

USE ApressFinancial  
GO  
  
CREATE TABLE TransactionDetails.Transactions  
(  
 TransactionId int IDENTITY(1,1) NOT NULL,  
 CustomerId int NOT NULL,  
 TransactionTypeId smallint NOT NULL,  
 DateEntered datetime NOT NULL,  
 Amount money NOT NULL,  
 ReferenceDetails nvarchar(50) NULL,  
 Notes nvarchar(max) NULL,  
 RelatedShareId int NULL,  
 RelatedProductId int NOT NULL  
 CONSTRAINT Transactions\_pk PRIMARY KEY NONCLUSTERED (TransactionId ASC)  
) ON [SECONDARY]  
GO

*Листинг 4 – Запрос для создания таблицы Transactions*

**

*Рисунок 29 – Результат выполнения файла в консоли*

**

*Рисунок 30 – Таблица успешно создана*

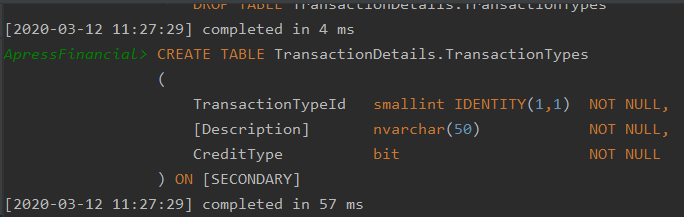
### Создание таблицы *TransactionTypes*

Так как DataGrip IDE не предусматривает создание таблиц с использованием Templates, данная таблица была создана с помощью SQL запроса.

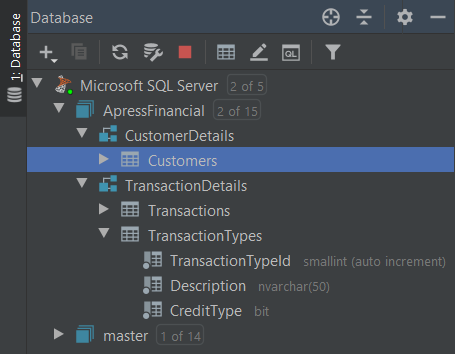
Был создан файл *create\_transactiontypes\_table.sql* со следующим содержимым.

USE ApressFinancial  
GO  
  
IF *OBJECT\_ID*('TransactionDetails.TransactionTypes', 'U') IS NOT NULL  
 DROP TABLE TransactionDetails.TransactionTypes  
GO  
  
CREATE TABLE TransactionDetails.TransactionTypes  
(  
 TransactionTypeId smallint IDENTITY(1,1) NOT NULL,  
 [Description] nvarchar(50) NOT NULL,  
 CreditType bit NOT NULL  
) ON [SECONDARY]  
GO

*Листинг 5 – Код создания таблицы TransactionTypes*



*Рисунок 31 – Результат выполнения кода в консоли*

**

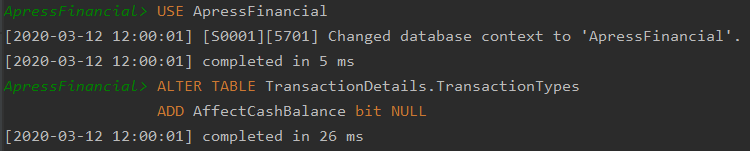
*Рисунок 32 – Таблица успешно создана*

### Изменение таблицы.

Необходимо добавить поле *AffectCashBalance* в уже существующую таблицу *TransactionTypes*. Для этого был создан файл со следующим SQL запросом.

USE ApressFinancial  
GO  
  
ALTER TABLE TransactionDetails.TransactionTypes  
ADD AffectCashBalance bit NULL  
GO

*Листинг 6 – SQL запрос создания поля AffectCashBalance*



*Рисунок 33 – Результат выполнения запроса в консоли. Поле успешно добавлено.*

Поле *AffectCashBalance* было изменено. Теперь оно обязательно должно содержать значения 0 или 1.

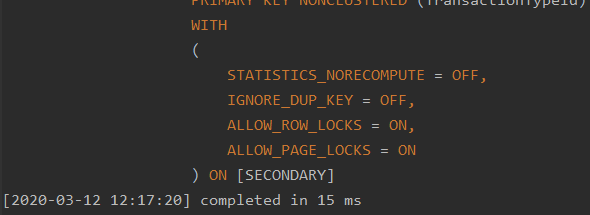
ALTER TABLE TransactionDetails.TransactionTypes  
ALTER COLUMN AffectCashBalance bit NOT NULL

*Листинг 7 – SQL запрос изменения поля AffectCashBalance*

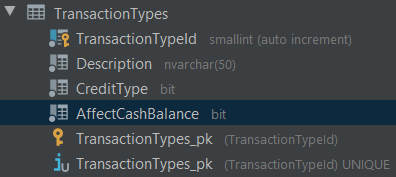
Необходимо создать Primary Key для таблицы TransactionTypes. Был создан файл со следующим содержимым.

USE ApressFinancial  
GO  
  
ALTER TABLE TransactionDetails.TransactionTypes  
ADD CONSTRAINT TransactionTypes\_pk  
 PRIMARY KEY NONCLUSTERED (TransactionTypeId)  
 WITH  
 (  
 STATISTICS\_NORECOMPUTE = OFF,  
 IGNORE\_DUP\_KEY = OFF,  
 ALLOW\_ROW\_LOCKS = ON,  
 ALLOW\_PAGE\_LOCKS = ON  
 ) ON [SECONDARY]  
GO

*Листинг 8 – SQL запрос для создания Primary Key*



*Рисунок 34 – Результат выполнения кода в консоли*



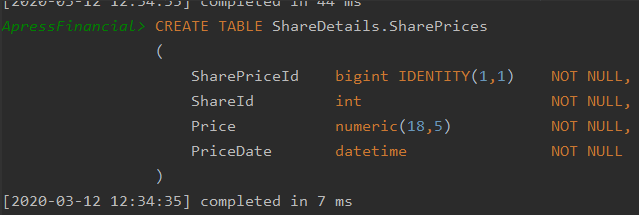
*Рисунок 35 – Таблица TransactionTypes после всех изменений*

### Создание остальных таблиц.

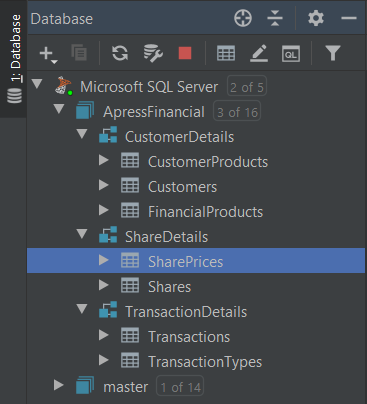
На данном шаге были созданы оставшиеся таблицы с использованием SQL запроса.

USE ApressFinancial  
GO  
  
CREATE TABLE CustomerDetails.FinancialProducts  
(  
 ProductId int NOT NULL,  
 ProductName nvarchar(50) NOT NULL  
) ON [SECONDARY]  
GO  
  
CREATE TABLE CustomerDetails.CustomerProducts  
(  
 CustomerFinancialProductId int NOT NULL,  
 CustomerId int NOT NULL,  
 FinancialProductId int NOT NULL,  
 AmountToCollect money NOT NULL,  
 Frequency smallint NOT NULL,  
 LastCollected datetime NOT NULL,  
 LastCollection datetime NOT NULL,  
 Renewable bit NOT NULL  
) ON [SECONDARY]  
GO  
  
CREATE SCHEMA ShareDetails AUTHORIZATION dbo  
 CREATE TABLE Shares  
 (  
 ShareId int IDENTITY(1,1) NOT NULL,  
 [Description] nvarchar(50) NOT NULL,  
 StockExchangeTicker nvarchar(50) NULL,  
 CurrentPrice numeric(18,5) NOT NULL  
 ) ON [SECONDARY]  
GO  
  
CREATE TABLE ShareDetails.SharePrices  
(  
 SharePriceId bigint IDENTITY(1,1) NOT NULL,  
 ShareId int NOT NULL,  
 Price numeric(18,5) NOT NULL,  
 PriceDate datetime NOT NULL  
)  
GO

*Листинг 9 – SQL запрос для создания оставшихся таблиц*

**

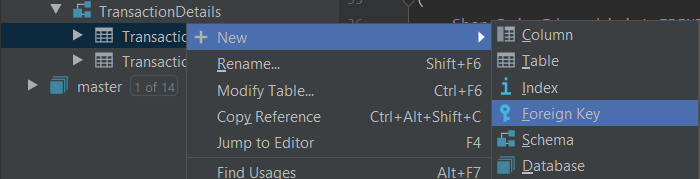
*Рисунок 36 – Результат выполнения кода в консоли*

**

*Рисунок 37 – Учебная база данных после создания новых таблиц*

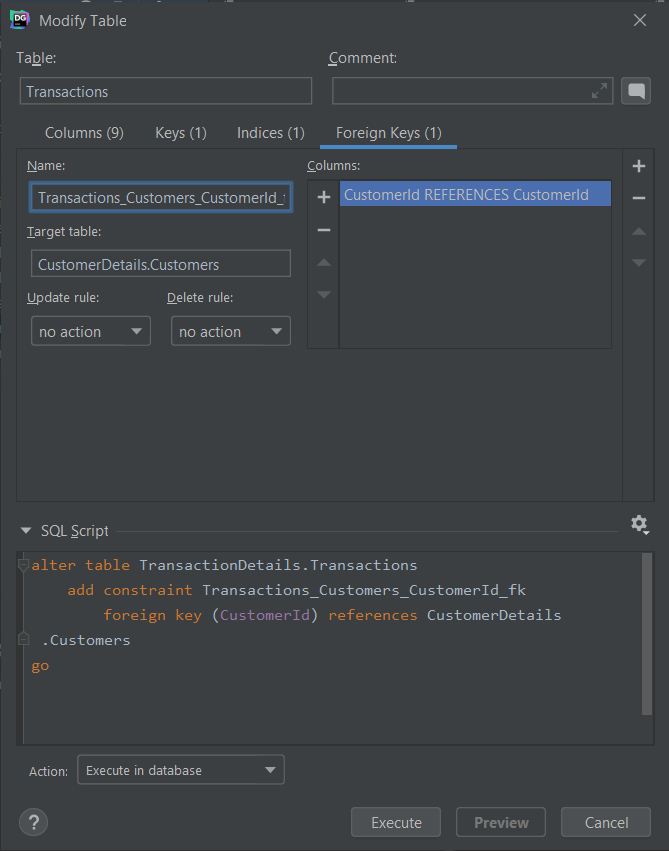
### Создание связи с помощью контекстного меню в DataGrip IDE.

Необходимо создать связь между таблицами *Transactions* и *Customers*.



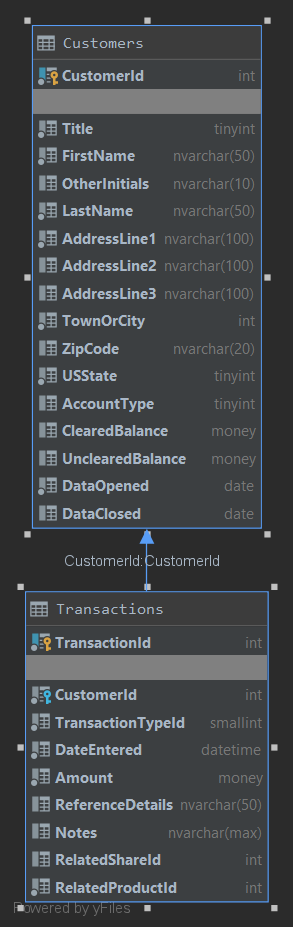
*Рисунок 38 – Transactions – ПКМ – New – Foreign Key*

Были заполнены данные о связи.



*Рисунок 39 – Окно создания новой связи*

После создания связи была создана диаграмма БД, на которой можно увидеть созданную связь.



*Рисунок 40 – Диаграмма базы данных*

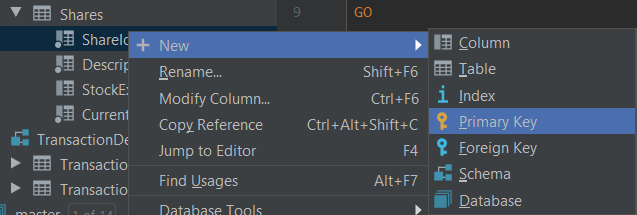
### Создание связи с помощью SQL запроса.

Была создана связь между таблицами *Transactions* и *Shares*. Файл *create\_transactions\_shares\_fk.sql* содержит SQL запрос для создания данной связи.

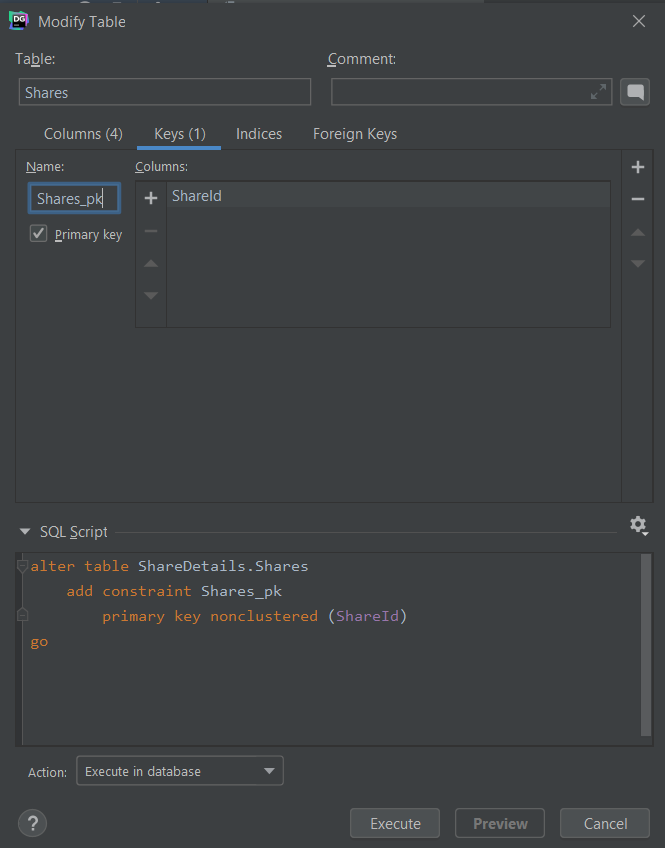
USE ApressFinancial  
GO  
  
ALTER TABLE TransactionDetails.Transactions  
WITH NOCHECK  
ADD CONSTRAINT Transactions\_Shares\_RelatedShareId\_fk  
 FOREIGN KEY (RelatedShareId)  
 REFERENCES ShareDetails.Shares(ShareId)  
GO

*Листинг 10 – Код создания связи*

Данный код при выполнении выдаёт ошибку. Это связано с тем, что в таблице *Shares* поле *ShareId* не является *Primary Key*.

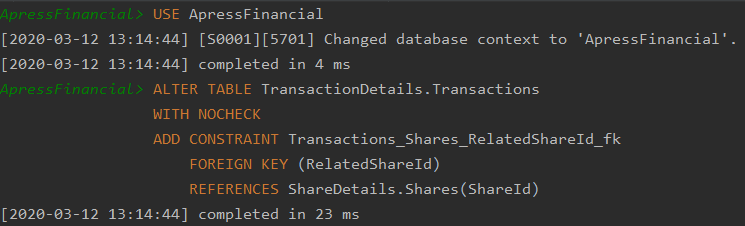


*Рисунок 41 – ShareId – ПКМ – New – Primary Key*

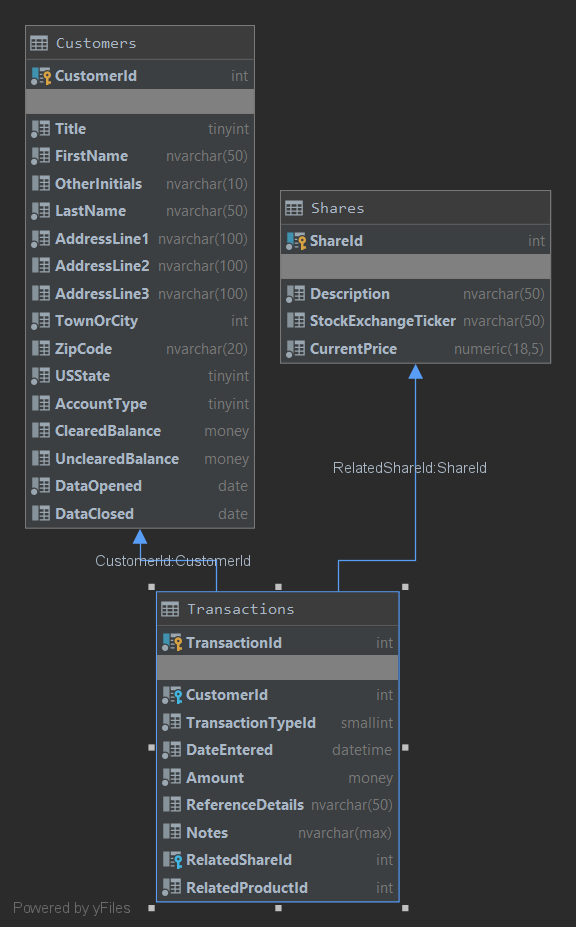


*Рисунок 42 – Окно создания Primary Key*

Теперь код успешно выполнился.



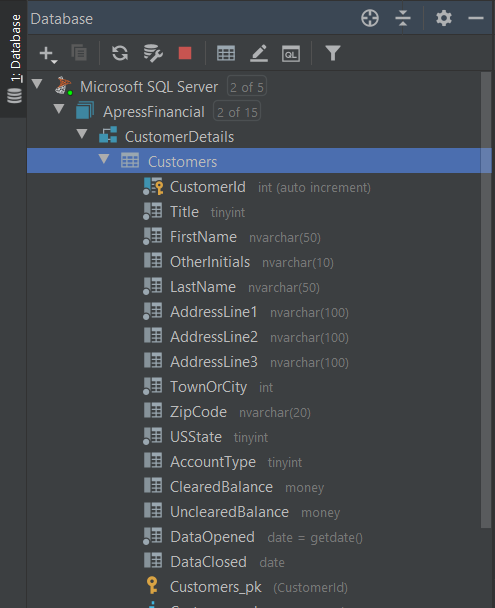
*Рисунок 43 – Результат выполнения кода в консоли*



*Рисунок 44 – новая схема учебной базы данных*

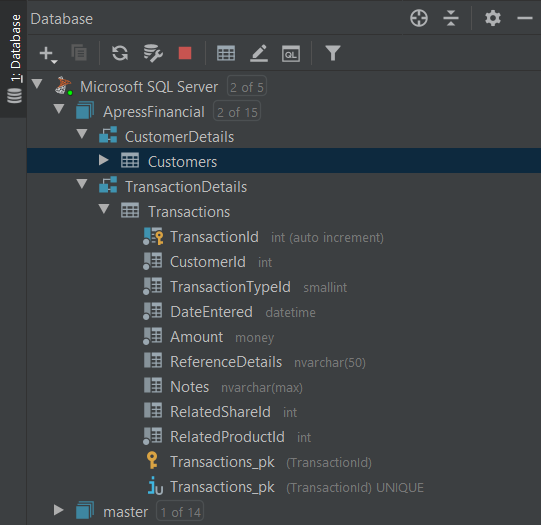
## Окончательные результаты.

### С помощью контекстного меню таблица была успешно создана.

**

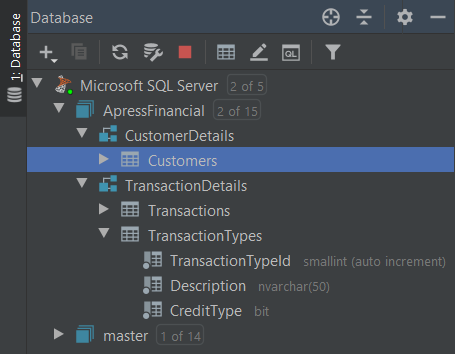
*Рисунок 45 – Таблица успешно создана*

### С помощью SQL запроса таблица была успешно создана.

**

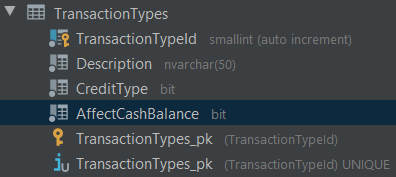
*Рисунок 46 – Таблица успешно создана*

### С помощью SQL запроса была успешно создана новая таблица.

**

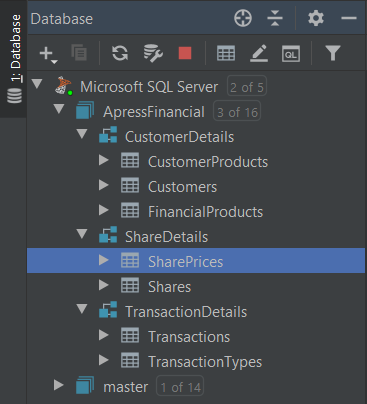
*Рисунок 47 – Таблица успешно создана*

### С помощью SQL запроса таблица *TransactionTypes* была изменена.



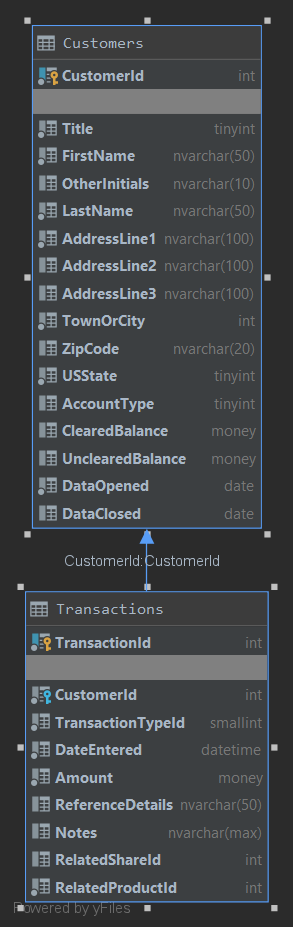
*Рисунок 48 – Таблица TransactionTypes после всех изменений*

### Были созданы оставшиеся таблицы.

**

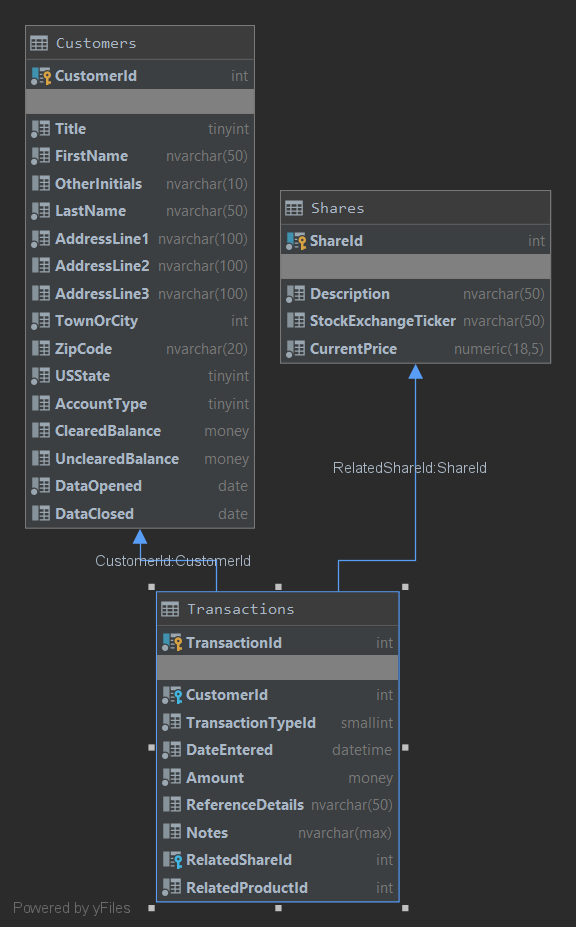
*Рисунок 49 – Учебная база данных после создания новых таблиц*

### С помощью контекстного меню была создана новая связь.



*Рисунок 50 – Диаграмма базы данных*

### С помощью SQL запроса была создана новая связь.



*Рисунок 51 – новая схема учебной базы данных*

## Выводы и анализ результатов работы.

В ходе данной лабораторной работы локально были созданы таблицы базы данных с помощью контекстного меню DataGrip IDE и с помощью SQL запроса. Были созданы связи между таблицами.

# «Создание Индексов и Диаграмм»

## Цель работы.

Создание индексов и построение диаграмм в учебной базе данных с помощью контекстного меню и SQL запросов.

## Задачи, решаемые при выполнении работы.

### Создание индекса с использованием контекстного меню конструктора таблиц.

### Создание индекса в Query Editor.

### Изменение индекса.

### Удаление индекса.

### Построение диаграмм базы данных.

## Объект исследования.

Microsoft SQL Server, учебная база данных, индексы и диаграммы базы данных.

## Исходные данные.

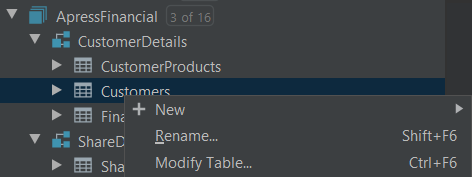
### Методическое пособие.

### Учебная база данных.

## Выполнение работы.

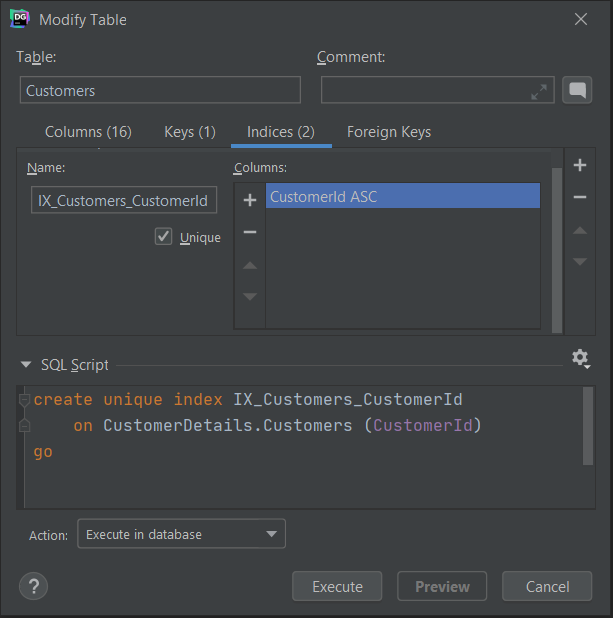
### Создание индекса с помощью конструктора таблиц.

Создадим новый индекс для таблицы Customers.



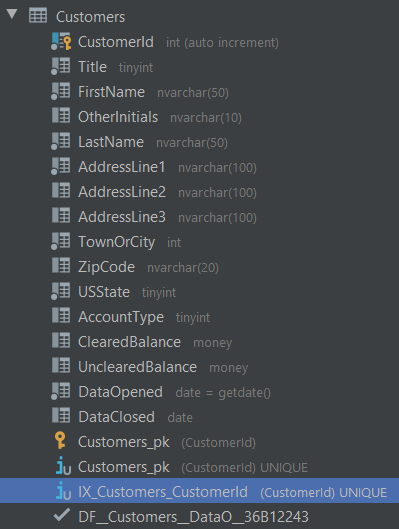
*Рисунок 52 – Редактирование таблицы: Customers – ПКМ – Modify Table…*

В окне Modify Table переходим во вкладку Indices.



*Рисунок 53 – Окно создания нового индекса*

Был создан новый индекс *IX\_Customers\_CustomerId*.



*Рисунок 54 – Новый индекс был добавлен в таблицу Customers*

### Создание индекса с использованием Query Editor.

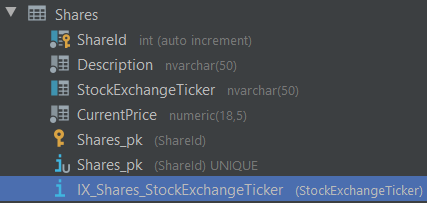
Было отработано создание индексов с использованием Query Editor.

Был создан новый индекс *IX\_Shares\_StockExchangeTicker* с использованием команд T-SQL.

USE ApressFinancial  
GO  
  
CREATE INDEX IX\_Shares\_StockExchangeTicker  
ON ShareDetails.Shares (StockExchangeTicker)  
GO

*Листинг 11 – Код создания индекса IX\_Shares\_StockExchangeTicker*

В результате выполнения кода был создан новый индекс.



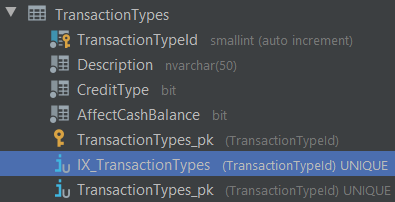
*Рисунок 55 – Создан новый индекс в таблице Shares*

Был создан новый индекс *IX\_TransactionTypes* с использованием команд T-SQL.

USE ApressFinancial  
GO  
  
CREATE UNIQUE CLUSTERED INDEX IX\_TransactionTypes  
ON TransactionDetails.TransactionTypes (TransactionTypeId ASC)  
WITH ( STATISTICS\_NORECOMPUTE = ON, SORT\_IN\_TEMPDB = ON,  
 DROP\_EXISTING = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = ON,  
 ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON ) ON [SECONDARY]  
GO

*Листинг 12 – Код создания индекса IX\_TransactionTypes для таблицы TransactionTypes*

В результате выполнения кода был создан новый индекс.



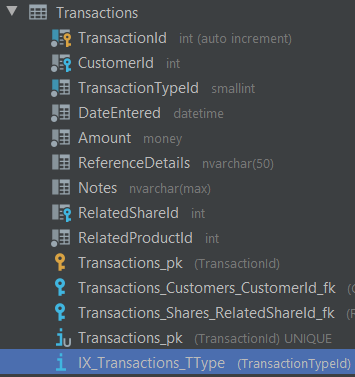
*Рисунок 56 – Создан новый индекс в таблице TransactionTypes*

Был создан новый индекс *IX\_Transactions\_TType* с использованием команд T-SQL.

USE ApressFinancial  
GO  
  
CREATE NONCLUSTERED INDEX IX\_Transactions\_TType  
ON TransactionDetails.Transactions (TransactionTypeId ASC)  
WITH ( STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = ON,  
 DROP\_EXISTING = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = ON,  
 ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [SECONDARY]  
GO

*Листинг 13 – Код создания индекса IX\_Transaction\_TType для таблицы Transactions*

В результате выполнения кода был создан новый индекс.



*Рисунок 57 – Создан новый индекс в таблице Transactions*

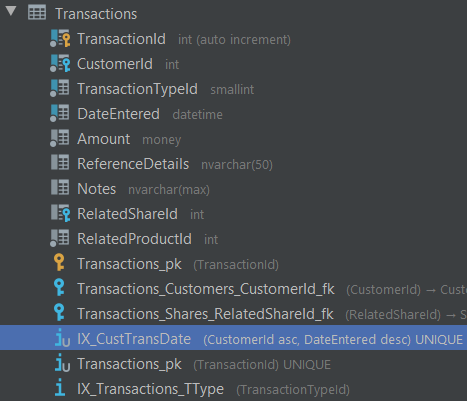
### Изменение индекса.

Изменить индекс с использованием инструкции ALTER невозможно. Чтобы изменить индекс, необходимо удалить старый и создать новый. Можно пересоздать уже существующий индекс. Для этого при создании индекса необходимо указать параметр DROP\_EXISTING со значением ON.

Далее был создан ещё один индекс *IX\_CustTransDate*.

USE ApressFinancial  
GO  
  
CREATE UNIQUE CLUSTERED INDEX IX\_CustTransDate  
ON TransactionDetails.Transactions (CustomerId ASC, DateEntered DESC)  
WITH ( STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = ON,  
 DROP\_EXISTING = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = OFF,  
 ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = OFF ) ON [SECONDARY]  
GO

*Листинг 14 – Код создания индекса IX\_CustTransDate*



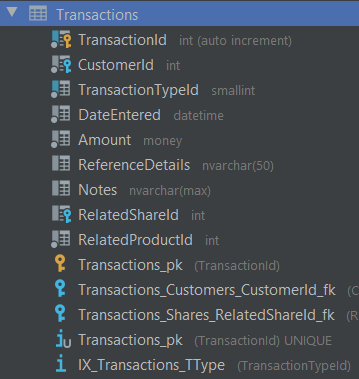
*Рисунок 58 – Создан новый индекс в таблице Transactions*

### Удаление индекса.

Был удалён индекс *IX\_CustTransDate*.

USE ApressFinancial  
GO  
  
DROP INDEX IX\_CustTransDate  
ON TransactionDetails.Transactions  
GO

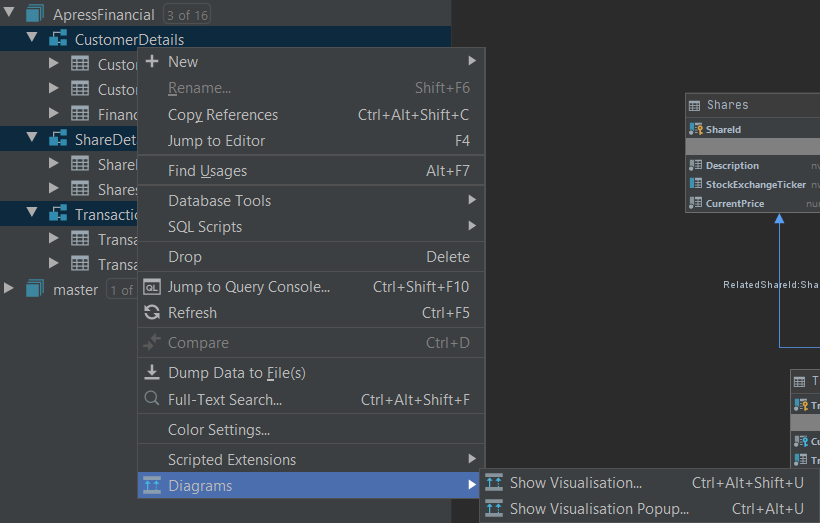
*Листинг 15 – Код удаления индекса IX\_CustTransDate*



*Рисунок 59 – Индекс удалён*

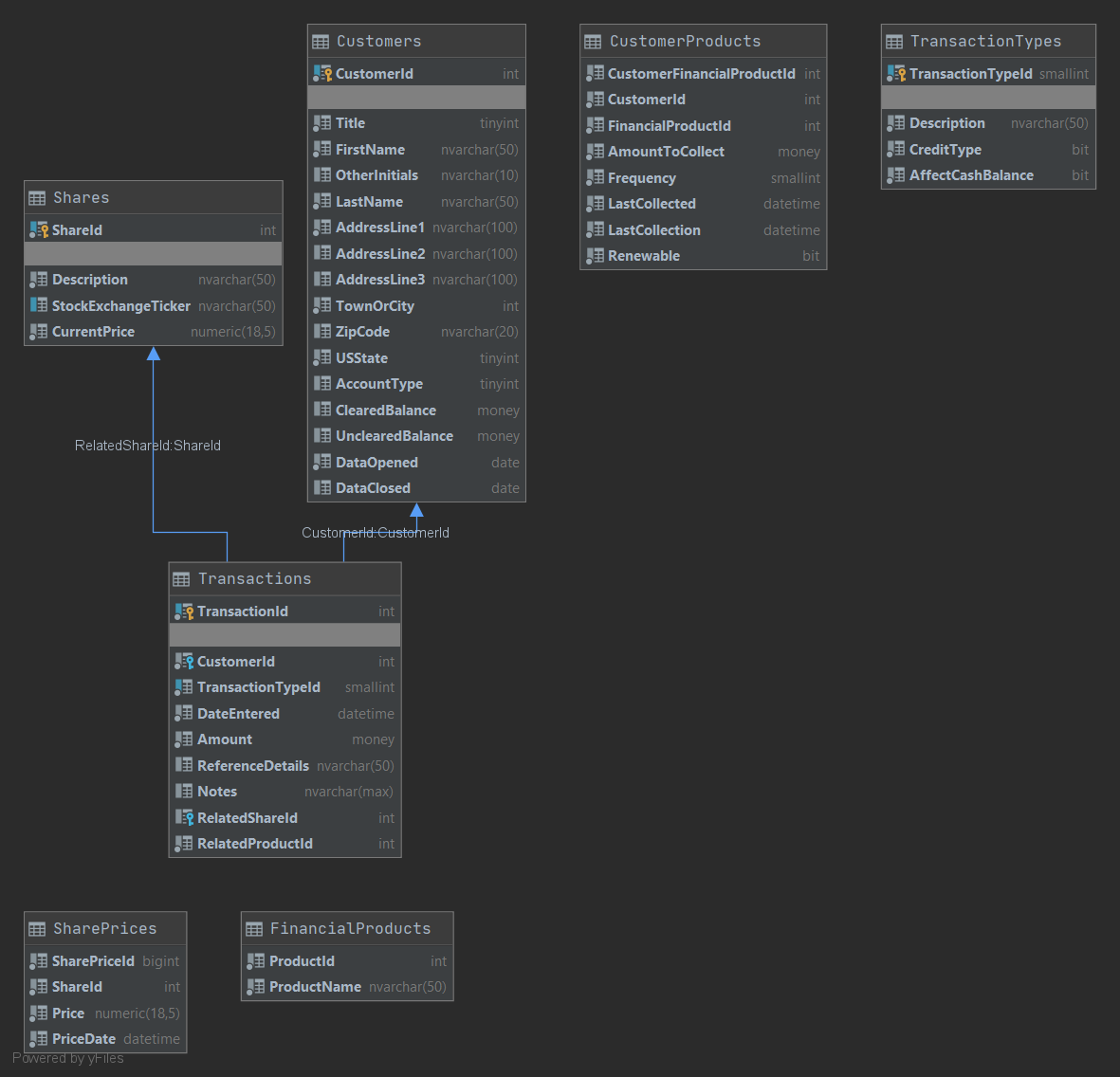
### Построение диаграмм баз данных.

DataGrip IDE уже имеет необходимые инструменты для построения диаграмм. Для этого используется пакет yFiles.



*Рисунок 60 – Создание диаграммы через диалоговое окно*

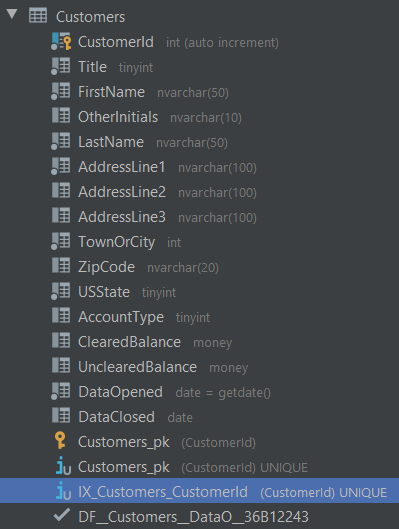
В результате была получена диаграмма для текущей учебной базы данных.



*Рисунок 61 – Диаграмма базы данных*

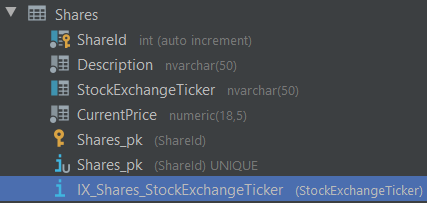
## Окончательные результаты.

### С помощью контекстного меню был создан индекс *IX\_Customers\_CustomerId*.

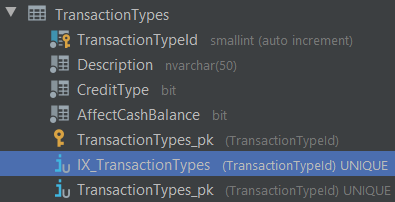


*Рисунок 62 – Новый индекс был добавлен в таблицу Customers*

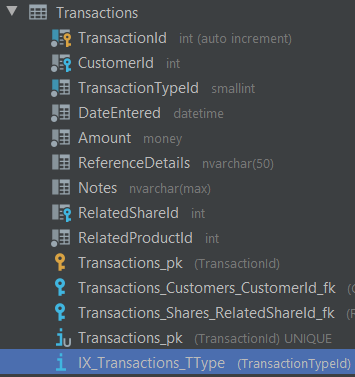
### Были созданы индексы с использованием Query Editor.



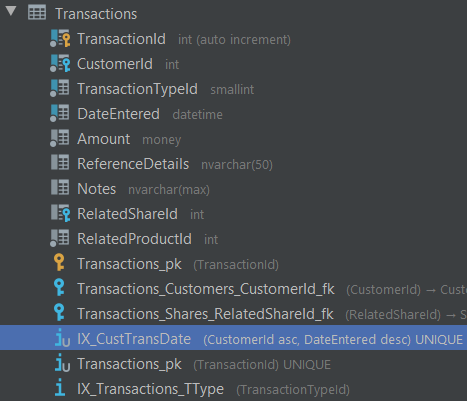
*Рисунок 63 – Создан новый индекс в таблице Shares*



*Рисунок 64 – Создан новый индекс в таблице TransactionTypes*

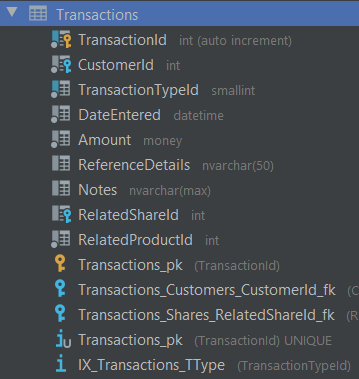


*Рисунок 65 – Создан новый индекс в таблице Transactions*



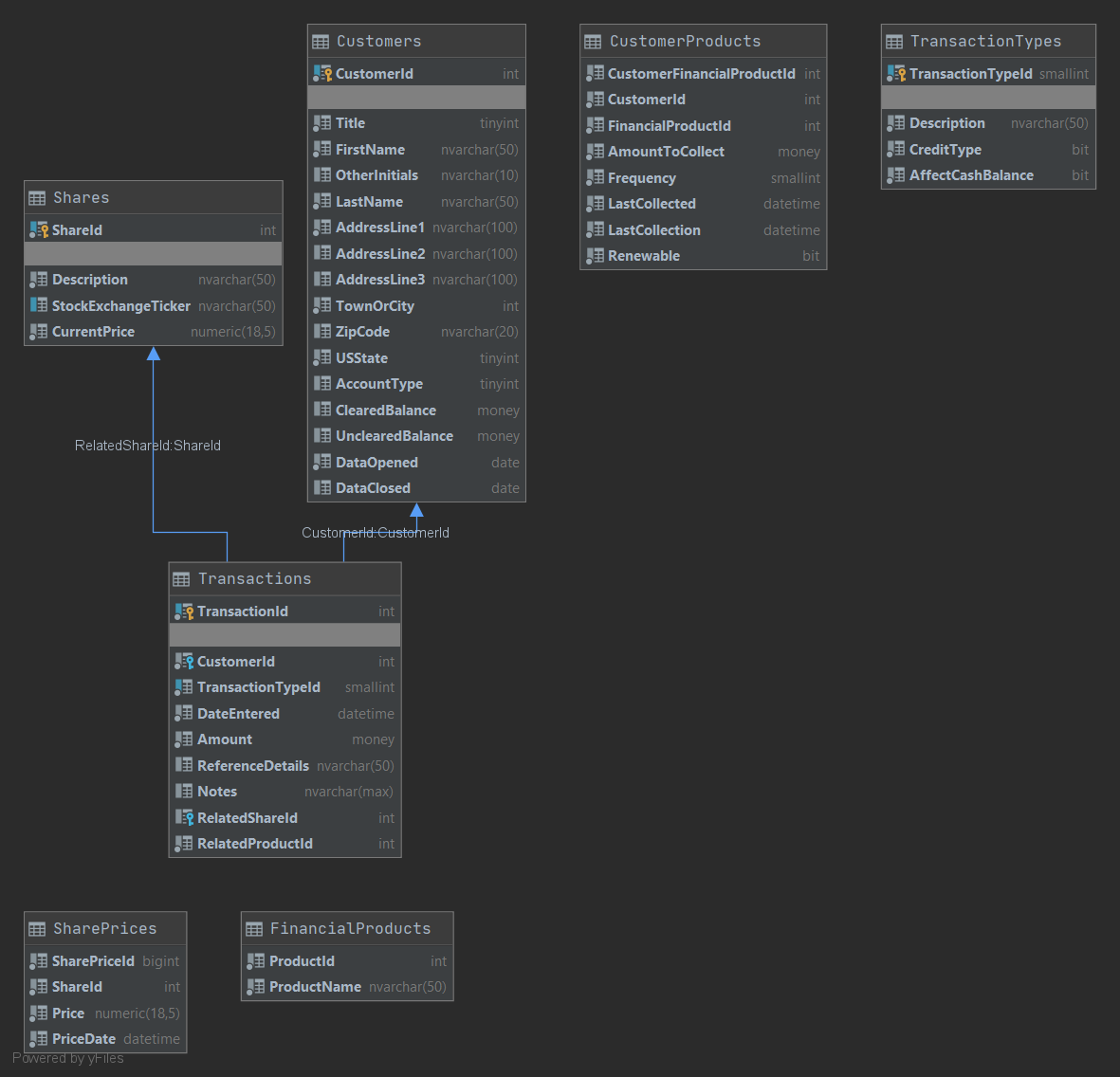
*Рисунок 66 – Создан новый индекс в таблице Transactions*

### Был удалён индекс с использованием Query Editor.



*Рисунок 67 – Индекс удалён*

### Была построена диаграмма для базы данных.



*Рисунок 68 – Диаграмма базы данных*

## Выводы и анализ результатов работы.

В ходе данной лабораторной работы локально были созданы индексы для таблиц базы данных с помощью контекстного меню DataGrip IDE и с помощью SQL запроса. Был рассмотрен процесс построения диаграммы для базы данных.