НАО «КАРАГАНДИНСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИМЕНИ АБЫЛКАСА САГИНОВА»

Кафедра информационных технологий и безопасности

**Лабораторная работа № 2**

**Дисциплина:** "Введение в базы данных"

**Тема:** Создание баз данных и таблиц в среде MS SQL Server. Информационное наполнение.

**Принял:**

Преподаватель: Жакина М.М

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)      (дата)*

**Выполнил:**

ст.гр.СИБ-23-7  Аксеной А.И

Караганда 2025

**Лабораторная работа №2**

**Тема:** Создание баз данных и таблиц в среде MS SQL Server. Информационное наполнение.  
  
**Цель:**

- усвоить способы создания таблиц, умолчаний, правил, ограничений БД средствами СУБД MS SQL Server;  
- усвоить способы создания индексов, отношений и схемы отношений (диаграммы) базы данных средствами СУБД MS SQL Server;  
- усвоить способы создания вода, удаления, редактирования данных в таблицах БД MS SQL Server.

**Ход работы:**

CREATE DATABASE laba2

ON (

NAME = 'laba2',

FILENAME = 'C:\MSSQL\SRSP\laba2.mdf',

SIZE = 10MB,

MAXSIZE = 100MB,

FILEGROWTH = 5MB

)

LOG ON (

NAME = 'laba2\_log',

FILENAME = 'C:\MSSQL\SRSP\laba2.ldf',

SIZE = 20MB,

MAXSIZE = 100MB,

FILEGROWTH = 5MB

);

GO

USE laba2;

GO

CREATE TYPE FullNameType FROM NVARCHAR(150) NOT NULL;

GO

CREATE TYPE CityType FROM NVARCHAR(100) NOT NULL;

GO

CREATE TYPE PositiveInt FROM INT NOT NULL;

GO

CREATE TYPE PriceType FROM DECIMAL(10,2) NOT NULL;

GO

CREATE TABLE Sellers (

SellerID INT IDENTITY(1,1) PRIMARY KEY,

FullName FullNameType,

City CityType

);

GO

CREATE TABLE Products (

ProductID INT IDENTITY(1,1) PRIMARY KEY,

ProductName NVARCHAR(150) NOT NULL

);

GO

CREATE TABLE ProductMovements (

MovementID INT IDENTITY(1,1) PRIMARY KEY,

MovementDate DATE NOT NULL,

SellerID INT NOT NULL,

ProductID INT NOT NULL,

Quantity PositiveInt CHECK (Quantity > 0),

Price PriceType CHECK (Price >= 0),

BuyerName FullNameType,

CONSTRAINT FK\_ProductMovements\_Seller FOREIGN KEY (SellerID)

REFERENCES Sellers(SellerID)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_ProductMovements\_Product FOREIGN KEY (ProductID)

REFERENCES Products(ProductID)

ON DELETE CASCADE

ON UPDATE CASCADE

);

GO

**Добавляем информацию в нашу БД**

INSERT INTO Sellers (FullName, City) VALUES

(N'Романов В.К.', N'МСК'),

(N'Петров П.П.', N'Санкт-Петербург');

INSERT INTO Products (ProductName) VALUES

(N'Мышь'),

(N'Телефон');

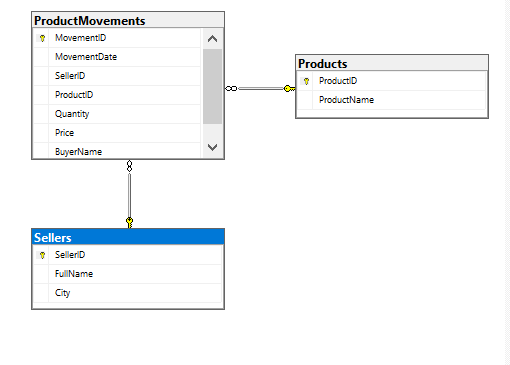
INSERT INTO ProductMovements (MovementDate, SellerID, ProductID, Quantity, Price, BuyerName) VALUES

('2025-10-08', 1, 1, 2, 5500.00, N'Сидоров А.А.'),

('2025-09-05', 2, 2, 5, 135000.00, N'Кузнецова Е.Е.');

GO

**Схема БД:**



**Аннотация:**

В схеме показан учет продаж товаров. Имееются продавцы, продукты и движение продуктов. Есть связи, где один продавец может учавствовать в нескольких продажах. И один товар может быть продан несколько раз.

**Описание зависимостей между таблицами БД:**

**ProductMovements – Sellers**

Тип связи, что SellerID связан с FOREIGN KEY

Так же ограничения:   
ON DELETE CASCADE– при удалении продавца, каскадом удаляются все его продажи.

ON UPDATE CASCADE – при изменении в таблице, будет и обновление в ProductMovements.

**ProductMovements – Products**

Снова связь через FOREIGN KEY

**И ограничения:**

ON DELETE CASCADE

ON UPDATE CASCADE

**Ограничения целостности**, то это все, где есть NOT NULL по типу ФИО Sellers, City, ProductName.  
CHECK **(Quantity > 0)** в ProductMovemetns и т.п.

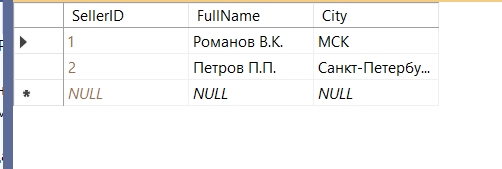
**Индексы:**

Sellers – PRIMARY KEY (Кластерный)

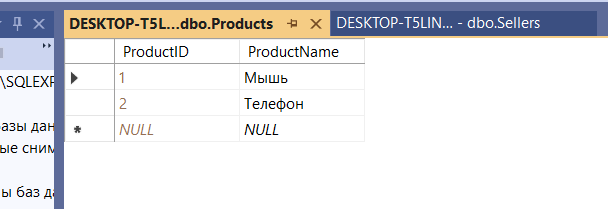
ProductsMovements – FOREIGN KEY (Некластерный)

**Скрины экранной формы Edit Top 200 rows для каждой из таблиц БД с тестовыми наборами данных.**

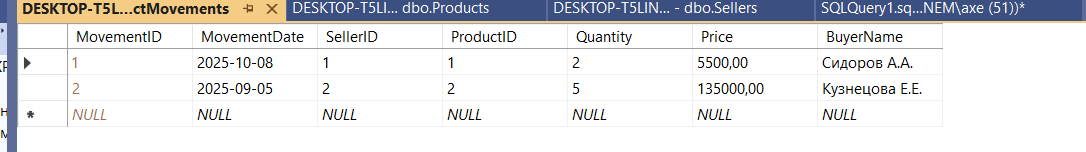
Sellers:



Products:

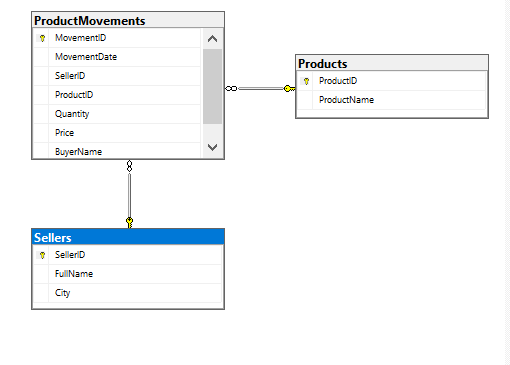


ProductsMovemets:



Индивидуальное задание:

1. Построить диаграмму вашей базы данных.



**Контрольные вопросы**:

1. **Перечень поддерживаемых типов данных?**

**bigint** - 8-ми байтное целое число от -2^63 до 2^63-1.   
**binary(n)** - Двоичные данные фиксированной длины n байт(1-8000), размен – n+4 байта.  
**bit** – Логический тип 0, 1 или NULL, размером 1 бит (до 8 бит в колонке).  
**char(n)** – Символы фикс.длины n(1-8000), не Unicode. Размером n-байт.  
**cursor** – Ссылка на курсор(для переменных и процедур), размер 1 байт.  
**date** – Даты с 0001-01-01 по 9999-12-31, размером 3 байта.  
**datetime** – Дата и время (1753-9999), точность 3 сотых секунды, размер 8 байт.  
**datetime2** – Расширенный datetime с большей точностью, размер 6-8 байт.  
**datetimeoffset** – datetime с часовым поясом(UTC-смещение), размер 8-10 байт.  
**decimal(p,s)/numeric(p,s)** – числа с фиксированной точностью и масштабом. Размер 5-17 байт(зависит от точности).  
**float(n)** – Число с плавающей точностью с диапазоном ±1.79e+308. Размер 4 или 8 байт.  
**hierarchyid** – Представление иерархии, размер зависит от структуры.  
**image** - Устаревший тип для больших бинарных данных (заменён на varchar(max)). Указатель — 16 байт.  
**int** – Целое число от -2^31 до 2^31-1, размером 4 байта.  
**money** – Денежные данные с точностью до 0.0001, размер 8 байт.  
**nchar(n)** – Unicode-строка, фиксированной длины n-символов (1-4000), размер 2\*n байт.  
**ntext** - Устаревший Unicode-текст (заменён на nvarchar(max)).  
**nvarchar(n)** - Unicode-строка переменной длины до n символов (1–4000). Размер — 2 байта на символ.  
**real** - Число с плавающей точностью меньшей точности (float(24)). Размер — 4 байта.  
**sql\_variant** - Контейнер для разных типов данных. Размер зависит от содержимого.  
**smalldatetime** - Дата и время с точностью до минуты (1900–2079). Размер — 4 байта.  
**smallint** - Целое 2 байта, от -32768 до 32767.  
**smallmoney** - Денежный тип с меньшим диапазоном и точностью, 4 байта.  
**table** - Табличный тип данных (локальные переменные, результаты функций).  
**text** - Устаревший тип для больших текстов (заменён на varchar(max)).  
**time** - Время с точностью до наносекунд. Размер 3–5 байт.  
**tinyint** - Целое 1 байт, от 0 до 255.  
**uniqueidentifier** - GUID — уникальный идентификатор, 16 байт.  
**varbinary(n)** - Двоичные данные переменной длины до n байт (1–8000). Размер — длина данных + 4 байта.  
**varchar(n)** - Строка переменной длины (не Unicode) до n символов (1–8000).  
**xml** - Тип для хранения XML данных.

1. **Пользовательский тип данных. Назначение и способы создания?**

Назначение для повторного использования, гарантия единообразия, централизованное управление изменениями и повышение читаемости.  
  
Способы создания основываясь на моей лабе:  
  
CREATE TYPE FullNameType FROM NVARCHAR(150) NOT NULL;

CREATE TYPE CityType FROM NVARCHAR(100) NOT NULL;

CREATE TYPE PositiveInt FROM INT NOT NULL;

CREATE TYPE PriceType FROM DECIMAL(10,2) NOT NULL;  
  
Так же и через UI: Заходим в БД, Programmability, Types – User-Defined Data Types, ПКМ – New User-Defined Data Type и указываем по шаблону.

1. **Способы определения таблиц?**

SELECT \*

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_TYPE = 'BASE TABLE';

SELECT s.name AS SchemaName, t.name AS TableName

FROM sys.tables t

JOIN sys.schemas s ON t.schema\_id = s.schema\_id;

**Через UI интерфейс**

1. **Назначение умолчаний, правил и ограничений?**

DEFAULT - Нужно для автоматической установки значения по умолчанию, если при вставке поле было пропущено.

**Ограничения** используются для контроля допустимых значений, связей между таблицами и уникальности.

Уникальный идентификатор каждой строки в таблице: PRIMARY KEY

Установка связи между таблицами: FOREIGN KEY

Установка ограничений на допустимые значения: CHECK

Гарантия уникальности значений в столбцах: UNIQUE

1. **Перечень поддерживаемых умолчаний, ограничений, правил?**

Для значений по умолчанию — DEFAULT.

Для контроля данных — CHECK, NOT NULL, UNIQUE.

Для связей между таблицами — FOREIGN KEY.

Для уникальной идентификации — PRIMARY KEY.

Старые правила RULE больше не используются — лучше применять CHECK.

1. **Способы назначений умолчаний, ограничений, правил**

В момент создания таблицы:

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

OrderDate DATE DEFAULT GETDATE(),

Quantity INT CHECK (Quantity > 0),

CustomerID INT NOT NULL,

CONSTRAINT FK\_Orders\_Customers FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

**После создания:**

ALTER TABLE Orders

ADD CONSTRAINT DF\_OrderDate DEFAULT GETDATE() FOR OrderDate;

**Правило:**

Quantity PositiveInt CHECK (Quantity > 0)

1. **Что представляют собой Null-значение?**

Это спец.значение в SQL, обозначающее неизвестное, отсутствующее или неопределенное значение. Используется при отсутствии данных, незаполненных полях, опциональных значениях.

1. **Свойство колонки IDENTITY?**

Это автоматическая нумерация строк , используется чаще всего для первичных ключей, не требует явного указания при INSERT

CREATE TABLE Products (

ProductID INT IDENTITY(1,1) PRIMARY KEY,

ProductName NVARCHAR(100) NOT NULL

);

1. **Назначение механизма отношений. Виды отношений?**

Это связи между таблицами, в реляционной БД, которые позволяют объединять связанные данные из разных таблиц, обеспечение целостности данных, упрощение обновления и удаления.

**Один к одному(1:1):** Каждой записи таблицы А соответствует одна запись таблицы В

CREATE TABLE Users (

UserID INT PRIMARY KEY,

Name NVARCHAR(100)

);

CREATE TABLE Passports (

PassportID INT PRIMARY KEY,

UserID INT UNIQUE FOREIGN KEY REFERENCES Users(UserID)

);

**Один ко многим(1:N):**

Одна запись таблицы А соответствует многим записям таблицы В.

Реализация через: FOREIGN KEY

CREATE TABLE ProductMovements (

SellerID INT,

FOREIGN KEY (SellerID) REFERENCES Sellers(SellerID)

);

**Многие ко многим(M:N):**

Одна запись А соответствует многим записям В и наоборот.  
Реализуется через промежуточную таблицу-связку

CREATE TABLE OrderProducts (

OrderID INT,

ProductID INT,

Quantity INT,

PRIMARY KEY (OrderID, ProductID),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

1. **Способы создания отношений**

Связка таблиц: FOREIGN KEY

Удаление связанных записей: ON DELETE CASCADE

Автоматическое обновление внешних ключей: ON UPDATE CASCADE

Указание на какую таблицу и колонку ссылка: REFERENCES

1. **Ссылочная целостность данных. Способы ее поддержания?**

Это гарантия, что внешние ключи в таблице всегда ссылаются на существующие строки в другой таблице. Через FOREIGN KEY, ON UPDATE CASCADE, ON DELETE CASCADE.

1. **Способы отображения зависимостей между таблицами БД**

**ER(Entity-Relationship) Diagram** – Создание через ПКМ на БД – новая БД диаграмма, добавляем таблицы и связь будет создана автоматом.

**Скрипт:**

SELECT

fk.name AS ForeignKey,

tp.name AS ParentTable,

cp.name AS ParentColumn,

tr.name AS ReferencedTable,

cr.name AS ReferencedColumn

FROM sys.foreign\_keys AS fk

INNER JOIN sys.foreign\_key\_columns AS fkc ON fk.object\_id = fkc.constraint\_object\_id

INNER JOIN sys.tables AS tp ON fkc.parent\_object\_id = tp.object\_id

INNER JOIN sys.columns AS cp ON fkc.parent\_object\_id = cp.object\_id AND fkc.parent\_column\_id = cp.column\_id

INNER JOIN sys.tables AS tr ON fkc.referenced\_object\_id = tr.object\_id

INNER JOIN sys.columns AS cr ON fkc.referenced\_object\_id = cr.object\_id AND fkc.referenced\_column\_id = cr.column\_id;

1. **Способы построения и изменения схемы БД**

**Построение с помощью скрипта:**

CREATE TABLE Sellers (

SellerID INT PRIMARY KEY,

FullName NVARCHAR(150),

City NVARCHAR(100)

);

**Создаем ключ:**

ALTER TABLE ProductMovements

ADD CONSTRAINT FK\_PM\_Seller FOREIGN KEY (SellerID)

REFERENCES Sellers(SellerID);

**UI интерфейс:**

ПКМ по таблице и Design

**Способы изменений:**Добавление: ALTER TABLE Sellers ADD Phone NVARCHAR(20);

Изменение типа: ALTER TABLE ProductMovements ALTER COLUMN Price DECIMAL(12,2);  
Удаление колонки: ALTER TABLE Sellers DROP COLUMN Phone;

**Удаление:**

Таблицы: DROP TABLE Products;  
Ключа: ALTER TABLE ProductMovements DROP CONSTRAINT FK\_PM\_Seller;

**UI:**

ПКМ – Design и вносим изменения.

1. **Назначение и классификация индексов**

**Назначение:**

Ускорение поиска: WHERE, JOIN, ORDER BY, GROUP BY  
Ускорение сортировки: ORDER BY  
Обеспечение уникальности: UNIQUE INDEX, PRIMARY KEY  
Поддержка ограничение: Автоматическое создание индексов для UNIQUE INDEX, PRIMARY KEY

**Классификация:**

По кол-ву столбцов: Single or composite  
Уникальность: UNIQUE or NONUNIQUE  
Структура хранения: Кластеризованный или нет   
Назначение: Первичный ключ, уникальный ключ, функциональный, фильтрованный.

1. **Способы создания индексов**

CREATE NONCLUSTERED INDEX IX\_ProductMovements\_SellerIDON ProductMovements(SellerID);

**Так же и по нескольким полям:**

CREATE NONCLUSTERED INDEX IX\_ProductMovements\_Seller\_Product

ON ProductMovements(SellerID, ProductID);

**Через UI:**   
ПКМ по таблице – design, вкладка индексы и ключи, добавить и указать столбец и тип индекса и сохранить.

1. **Способы ввода данных в таблицы. Ограничения целостности при вводе данных**

INSERT: INSERT INTO Sellers (FullName, City) VALUES (N'Иванов И.И.', N'Москва');  
INSERT INTO:   
INSERT INTO Products (ProductName)

SELECT ProductName FROM TempProducts;

1. **Понятие транзакции. Свойства транзакций**

Это единица работы с БД, представляющая собой последовательность операций, которые: либо все выполняются полностью, либо не выполняются вовсе.

**Свойства: ACID**A – Atomicity – все действия выполняются полностью или не выполняются вообще.  
C – Consistency(Согласованность) – После транзакции данные остаются в корректном состоянии.  
I – Isolation – Одновременные транзакции не мешают друг другу.  
D – Durability(Долговечность) – После фиксации(COMMIT) данные сохраняются даже при сбое.

1. **Удаление данных, усечение таблиц, удаление таблиц**

Удаление данных: DELETE FROM Sellers WHERE SellerID = 1;  
Усечение таблицы: TRUNCATE TABLE ProductMovements;  
Удаление таблицы: DROP TABLE Sellers;