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  Ансаров И.Е.

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 sklad\_uchet

ON (

NAME = 'sklad\_uchet',

FILENAME = 'C:\MSSQL\sklad\_uchet.mdf',

SIZE = 10MB,

MAXSIZE = 100MB,

FILEGROWTH = 5MB

)

LOG ON (

NAME = 'sklad\_uchet\_log',

FILENAME = 'C:\MSSQL\sklad\_uchet.ldf',

SIZE = 10MB,

MAXSIZE = 100MB,

FILEGROWTH = 5MB

);

GO

USE sklad\_uchet;

GO

CREATE TYPE FullNameType FROM NVARCHAR(150) NOT NULL;

GO

CREATE TYPE PhoneType FROM NVARCHAR(20) NULL;

GO

CREATE TYPE MoneyType FROM DECIMAL(12,2) NOT NULL;

GO

CREATE TYPE PercentType FROM DECIMAL(5,2) NOT NULL;

GO

CREATE TABLE Bank (

BankCode CHAR(5) PRIMARY KEY,

BankName NVARCHAR(100) NOT NULL

);

GO

CREATE TABLE Organization (

BIN CHAR(12) PRIMARY KEY,

OrgName NVARCHAR(150) NOT NULL,

BankCode CHAR(5) NOT NULL,

AccountNumber NVARCHAR(30) NOT NULL,

Address NVARCHAR(200),

Director FullNameType,

Phone PhoneType,

CONSTRAINT FK\_Organization\_Bank FOREIGN KEY (BankCode)

REFERENCES Bank(BankCode)

ON UPDATE CASCADE

);

GO

CREATE TABLE Category (

CategoryCode INT IDENTITY(1,1) PRIMARY KEY,

CategoryName NVARCHAR(100) NOT NULL

);

GO

CREATE TABLE Product (

Article NVARCHAR(20) PRIMARY KEY,

ProductName NVARCHAR(100) NOT NULL,

Unit NVARCHAR(20) NOT NULL,

CategoryCode INT NOT NULL,

CONSTRAINT FK\_Product\_Category FOREIGN KEY (CategoryCode)

REFERENCES Category(CategoryCode)

ON UPDATE CASCADE

);

GO

CREATE TABLE Warehouse (

WarehouseID INT IDENTITY(1,1) PRIMARY KEY,

WarehouseName NVARCHAR(100) NOT NULL,

Storekeeper FullNameType

);

GO

CREATE TABLE Stock (

WarehouseID INT NOT NULL,

Month INT CHECK (Month BETWEEN 1 AND 12),

Year INT CHECK (Year >= 2000),

Article NVARCHAR(20) NOT NULL,

AvgPrice MoneyType CHECK (AvgPrice >= 0),

QtyStart INT DEFAULT 0 CHECK (QtyStart >= 0),

QtyIn INT DEFAULT 0 CHECK (QtyIn >= 0),

QtyOut INT DEFAULT 0 CHECK (QtyOut >= 0),

CONSTRAINT PK\_Stock PRIMARY KEY (WarehouseID, Month, Year, Article),

CONSTRAINT FK\_Stock\_Product FOREIGN KEY (Article)

REFERENCES Product(Article)

ON UPDATE CASCADE,

CONSTRAINT FK\_Stock\_Warehouse FOREIGN KEY (WarehouseID)

REFERENCES Warehouse(WarehouseID)

ON UPDATE CASCADE

);

GO

CREATE TABLE Tax (

TaxCode INT IDENTITY(1,1) PRIMARY KEY,

TaxName NVARCHAR(100) NOT NULL,

PercentRate PercentType CHECK (PercentRate >= 0)

);

GO

CREATE TABLE Invoice (

InvoiceNo INT IDENTITY(1,1) PRIMARY KEY,

InvoiceDate DATE NOT NULL,

InvoiceType NVARCHAR(10) CHECK (InvoiceType IN (N'приходная', N'расходная')),

BIN CHAR(12) NOT NULL,

WarehouseID INT NOT NULL,

CONSTRAINT FK\_Invoice\_Organization FOREIGN KEY (BIN)

REFERENCES Organization(BIN)

ON UPDATE CASCADE,

CONSTRAINT FK\_Invoice\_Warehouse FOREIGN KEY (WarehouseID)

REFERENCES Warehouse(WarehouseID)

ON UPDATE CASCADE

);

GO

CREATE TABLE ProductMovement (

InvoiceNo INT NOT NULL,

Article NVARCHAR(20) NOT NULL,

Quantity INT CHECK (Quantity > 0),

Price MoneyType CHECK (Price >= 0),

CONSTRAINT PK\_ProductMovement PRIMARY KEY (InvoiceNo, Article),

CONSTRAINT FK\_ProductMovement\_Invoice FOREIGN KEY (InvoiceNo)

REFERENCES Invoice(InvoiceNo)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_ProductMovement\_Product FOREIGN KEY (Article)

REFERENCES Product(Article)

ON UPDATE CASCADE

);

GO

CREATE TABLE Taxation (

InvoiceNo INT NOT NULL,

TaxCode INT NOT NULL,

TaxSum MoneyType CHECK (TaxSum >= 0),

CONSTRAINT PK\_Taxation PRIMARY KEY (InvoiceNo, TaxCode),

CONSTRAINT FK\_Taxation\_Invoice FOREIGN KEY (InvoiceNo)

REFERENCES Invoice(InvoiceNo)

ON DELETE CASCADE,

CONSTRAINT FK\_Taxation\_Tax FOREIGN KEY (TaxCode)

REFERENCES Tax(TaxCode)

);

GO

CREATE INDEX IX\_Organization\_Bank ON Organization(BankCode);

CREATE INDEX IX\_Product\_Category ON Product(CategoryCode);

CREATE INDEX IX\_Stock\_Product ON Stock(Article);

CREATE INDEX IX\_Stock\_Warehouse ON Stock(WarehouseID);

CREATE INDEX IX\_Invoice\_Org ON Invoice(BIN);

CREATE INDEX IX\_Invoice\_Warehouse ON Invoice(WarehouseID);

CREATE INDEX IX\_ProductMovement\_Invoice ON ProductMovement(InvoiceNo);

CREATE INDEX IX\_Taxation\_Invoice ON Taxation(InvoiceNo);

GO

INSERT INTO Bank (BankCode, BankName)

VALUES ('B001', N'Каспи Банк'), ('B002', N'Халык Банк');

INSERT INTO Organization (BIN, OrgName, BankCode, AccountNumber, Address, Director, Phone)

VALUES

('123456789012', N'ТОО Альфа', 'B001', 'KZ123456789', N'Алматы, ул. Абая 10', N'Иванов А.А.', N'+77011234567'),

('987654321098', N'ТОО Бета', 'B002', 'KZ987654321', N'Астана, пр. Назарбаева 25', N'Петров Б.Б.', N'+77019876543');

INSERT INTO Category (CategoryName)

VALUES (N'Компьютеры'), (N'Периферия'), (N'Смартфоны');

INSERT INTO Product (Article, ProductName, Unit, CategoryCode)

VALUES

('A001', N'Ноутбук Lenovo', N'шт', 1),

('A002', N'Монитор Samsung', N'шт', 2),

('A003', N'Мышь Logitech', N'шт', 2),

('A004', N'iPhone 14', N'шт', 3);

INSERT INTO Warehouse (WarehouseName, Storekeeper)

VALUES

(N'Главный склад', N'Сидоров С.С.'),

(N'Склад №2', N'Кузнецов К.К.');

INSERT INTO Stock (WarehouseID, Month, Year, Article, AvgPrice, QtyStart, QtyIn, QtyOut)

VALUES

(1, 10, 2025, 'A001', 250000, 5, 3, 2),

(1, 10, 2025, 'A002', 120000, 10, 5, 4),

(2, 10, 2025, 'A003', 8000, 50, 20, 10);

INSERT INTO Tax (TaxName, PercentRate)

VALUES

(N'НДС', 12.00),

(N'Пошлина', 5.00);

INSERT INTO Invoice (InvoiceDate, InvoiceType, BIN, WarehouseID)

VALUES

('2025-10-01', N'приходная', '123456789012', 1),

('2025-10-02', N'расходная', '987654321098', 2);

INSERT INTO ProductMovement (InvoiceNo, Article, Quantity, Price)

VALUES

(1, 'A001', 2, 250000),

(1, 'A002', 3, 120000),

(2, 'A003', 5, 8000),

(2, 'A004', 1, 500000);

INSERT INTO Taxation (InvoiceNo, TaxCode, TaxSum)

VALUES

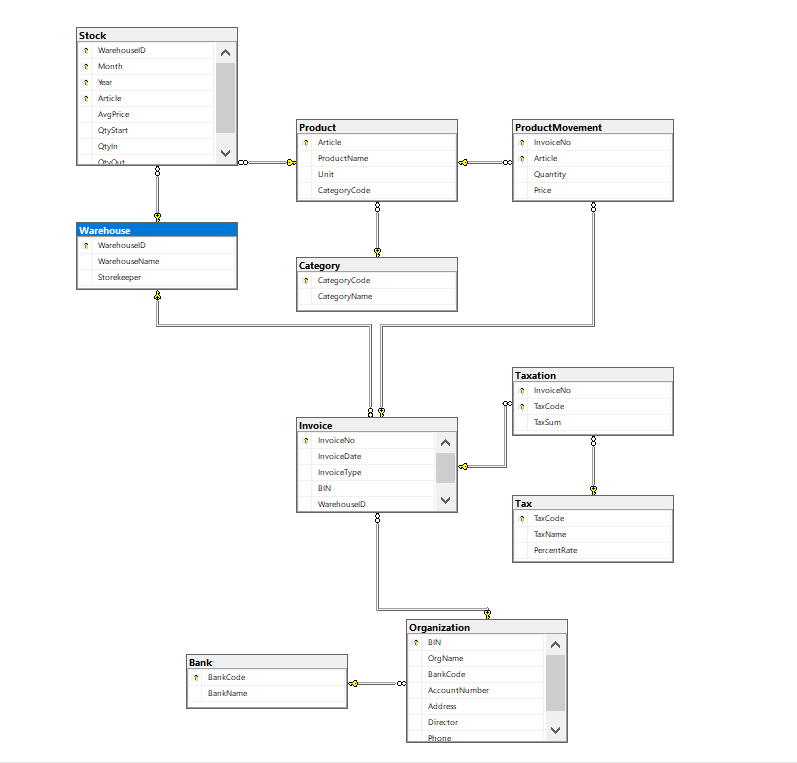
(1, 1, 15000.00),

(1, 2, 6000.00),

(2, 1, 7200.00);

GO

**Схема БД:**



**Аннотация:**

В схеме показан **учет движения товаров на складах организации**.  
База данных отражает взаимосвязь между товарами, их категориями, складами, накладными, движением и налогообложением.  
Один товар принадлежит определенной категории и может храниться на разных складах, участвовать в нескольких накладных и движениях.  
Каждая накладная оформляется для одной организации и одного склада, но может включать несколько позиций товаров и начисленные налоги.

**Описание зависимостей между таблицами БД:**

**Organization – Bank**

**Тип связи:** один ко многим (1 → ∞)  
**Ключи:**

Bank.BankCode — PRIMARY KEY

Organization.BankCode — FOREIGN KEY, ссылается на Bank(BankCode)  
**Ограничения:**

ON UPDATE CASCADE — при изменении кода банка (BankCode) обновляется соответствующее значение в Organization.  
**Описание:**  
Одна запись о банке может быть связана с несколькими организациями, но каждая организация имеет только один банк.

**Product – Category**

**Тип связи:** один ко многим (1 → ∞)  
**Ключи:**

Category.CategoryCode — RIMARY KEY

Product.CategoryCode — — FOREIGN KEY, ссылается на Category(CategoryCode)  
**Ограничения:**

ON UPDATE CASCADE — при изменении кода категории обновляется значение в Product.  
**Описание:**  
Каждая категория включает множество товаров, но каждый товар принадлежит только одной категории.

**Stock – Warehouse**

**Тип связи:** один ко многим (1 → ∞)  
**Ключи:**

Warehouse.WarehouseID — **PRIMARY KEY**

Stock.WarehouseID — FOREIGN KEY, ссылается на Warehouse(WarehouseID)  
**Ограничения:**

ON UPDATE CASCADE — при изменении ID склада обновляется значение в Stock.  
**Описание:**  
Один склад может содержать множество позиций остатков, но каждая запись в Stock относится к одному складу.

**Stock – Product**

**Тип связи:** один ко многим (1 → ∞)  
**Ключи:**

Product.Article — **PRIMARY KEY**

Stock.Article — FOREIGN KEY, ссылается на Product(Article)  
**Ограничения:**

ON UPDATE CASCADE — при изменении артикула товара обновляется значение в Stock.  
**Описание:**  
Каждая позиция склада относится к конкретному товару. Один товар может храниться на разных складах.

**Invoice – Organization**

**Тип связи:** один ко многим (1 → ∞)  
**Ключи:**

Organization.BIN — **PRIMARY KEY**

Invoice.BIN —FOREIGN KEY, ссылается на Organization(BIN)  
**Ограничения:**

ON UPDATE CASCADE — при изменении BIN организации обновляется соответствующее значение в Invoice.  
**Описание:**  
Одна организация может участвовать во множестве накладных, но каждая накладная оформлена на одну организацию.

**Invoice – Warehouse**

**Тип связи:** один ко многим (1 → ∞)  
**Ключи:**

Warehouse.WarehouseID — **PRIMARY KEY**

Invoice.WarehouseID — FOREIGN KEY, ссылается на Warehouse(WarehouseID)  
**Ограничения:**

ON UPDATE CASCADE — при изменении идентификатора склада обновляются связанные накладные.  
**Описание:**  
Один склад может иметь несколько накладных (приходных и расходных).

**ProductMovement – Invoice**

**Тип связи:** один ко многим (1 → ∞)  
**Ключи:**

Invoice.InvoiceNo — **PRIMARY KEY**

ProductMovement.InvoiceNo — FOREIGN KEY, ссылается на Invoice(InvoiceNo)  
**Ограничения:**

ON DELETE CASCADE — при удалении накладной удаляются все связанные движения товаров;

ON UPDATE CASCADE — при изменении номера накладной обновляется связь.  
**Описание:**  
Каждое движение товара связано с одной накладной, но одна накладная может содержать множество позиций товаров.

**ProductMovement – Product**

**Тип связи:** один ко многим (1 → ∞)  
**Ключи:**

Product.Article — **PRIMARY KEY**

ProductMovement.Article — FOREIGN KEY, ссылается на Product(Article)  
**Ограничения:**

ON UPDATE CASCADE — при изменении артикула товара обновляется связь.  
**Описание:**  
Один товар может участвовать во множестве движений, но каждая запись движения относится к конкретному товару.

**Taxation – Invoice**

**Тип связи:** один ко многим (1 → ∞)  
**Ключи:**

Invoice.InvoiceNo — **PRIMARY KEY**

Taxation.InvoiceNo — FOREIGN KEY, ссылается на Invoice(InvoiceNo)  
**Ограничения:**

ON DELETE CASCADE — при удалении накладной каскадно удаляются все записи о начисленных налогах.  
**Описание:**  
Каждая накладная может иметь несколько налогов, связанных с ней.

**Taxation – Tax**

**Тип связи:** один ко многим (1 → ∞)  
**Ключи:**

Tax.TaxCode — **PRIMARY KEY**

Taxation.TaxCode — FOREIGN KEY, ссылается на Tax(TaxCode)  
**Описание:**  
Один налог может применяться во многих накладных.

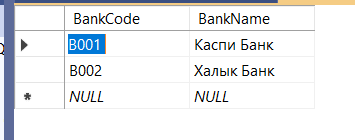
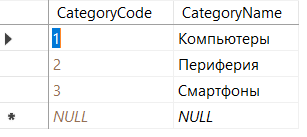
**⚙️ Ограничения целостности данных**

| **Тип ограничения** | **Таблица / Поле** | **Описание** |
| --- | --- | --- |
| **PRIMARY KEY** | Все ID-поля и коды (BankCode, BIN, Article, WarehouseID, TaxCode, InvoiceNo) | Уникальные ключи таблиц |
| FOREIGN KEY | Все связи, перечисленные выше | Поддержание связей между таблицами |
| **NOT NULL** | OrgName, BankCode, AccountNumber, ProductName, Unit, WarehouseName, InvoiceDate, InvoiceType и др. | Поля обязательны для заполнения |
| **CHECK (Quantity > 0)** | ProductMovement.Quantity | Проверка положительного количества товара |
| **CHECK (AvgPrice >= 0)** | Stock.AvgPrice | Средняя цена не может быть отрицательной |
| **CHECK (PercentRate >= 0)** | Tax.PercentRate | Процент налога не может быть отрицательным |
| **CHECK (Month BETWEEN 1 AND 12)** | Stock.Month | Проверка корректности месяца |
| **CHECK (Year >= 2000)** | Stock.Year | Контроль минимального года учёта |

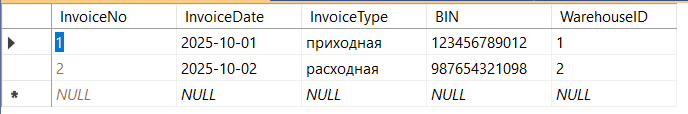
**📈 Индексы**

| **Таблица** | **Поле** | **Тип индекса** | **Назначение** |
| --- | --- | --- | --- |
| **Bank** | BankCode | PRIMARY KEY (кластерный) | Уникальный код банка |
| **Organization** | BIN | PRIMARY KEY (кластерный) | Уникальный BIN организации |
| **Organization** | BankCode | FOREIGN KEY (некластерный) | Быстрый поиск организаций по банку |
| **Category** | CategoryCode | PRIMARY KEY (кластерный) | Идентификатор категории |
| **Product** | Article | PRIMARY KEY (кластерный) | Уникальный артикул товара |
| **Product** | CategoryCode | FOREIGN KEY (некластерный) | Быстрый фильтр по категориям |
| **Warehouse** | WarehouseID | PRIMARY KEY (кластерный) | Уникальный склад |
| **Stock** | (WarehouseID, Month, Year, Article) | PRIMARY KEY (кластерный составной) | Уникальность остатков на складе |
| **Invoice** | InvoiceNo | PRIMARY KEY (кластерный) | Уникальный номер накладной |
| **ProductMovement** | (InvoiceNo, Article) | PRIMARY KEY (кластерный) | Уникальность товара в накладной |
| **ProductMovement** | InvoiceNo | FOREIGN KEY (некластерный) | Быстрый доступ по накладной |
| **Taxation** | (InvoiceNo, TaxCode) | PRIMARY KEY (кластерный) | Уникальность налогов по накладной |
| **Taxation** | InvoiceNo | FOREIGN KEY (некластерный) | Связь с накладной |

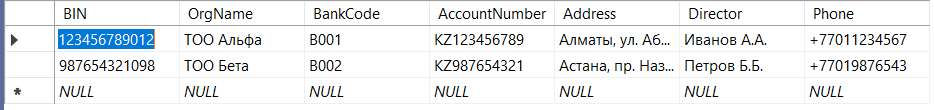
**Скрины экранной формы Edit Top 200 rows для каждой из таблиц БД с тестовыми наборами данных.  
  
Bank:**

  
  
**Category**:  


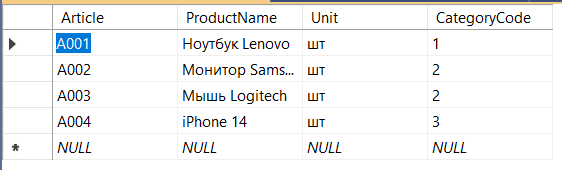
**Inovice:**



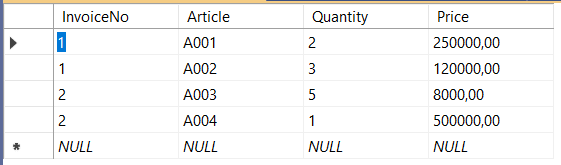
**Organization:**



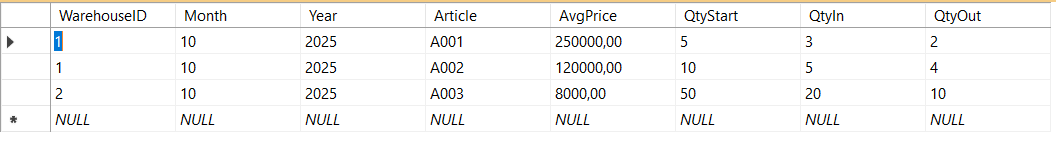
**Product:**



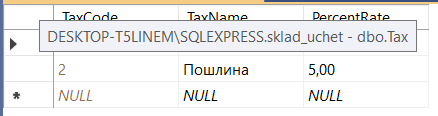
**ProductMovement:**



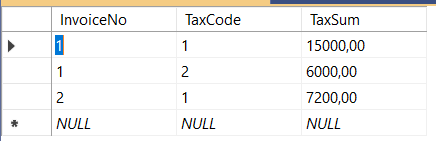
**Stock:**



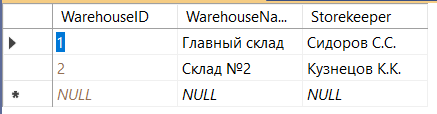
**Tax:**



**Taxation:**



**Warehouse:**



**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;