Основы SQL. DDL

Базы данных

Юдинцев В. В.

Кафедра математических методов в экономике

21 марта 2023 г.

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Содержание

SQL

Язык SQL

- **SQL** – язык структурированных запросов (Structured Query Language) – специальный язык программирования, разработанный для управления данными в реляционных системах управления базами данных.
- Создан в 1970-х годах под названием SEQUEL для СУБД System R.
- Первый официальный стандарт языка был принят в 1986 году
- Развитие: SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2006, SQL:2008, SQL:2011, SQL:2016.

Язык SQL

```
--Создание таблицы Goods
CREATE TABLE Goods (
    ProductId   INT IDENTITY(1,1) NOT NULL CONS
    Category    INT NOT NULL,
    ProductName VARCHAR(100) NOT NULL,
    Price       MONEY NULL,
);
GO

--Создание таблицы Categories
CREATE TABLE Categories (
    CategoryId   INT IDENTITY(1,1) NOT NULL CON
    CategoryName VARCHAR(100) NOT NULL
);

--Добавление строк в таблицу Categories
INSERT INTO Categories(CategoryName)
    VALUES ('Комплектующие ПК'),
           ('Мобильные устройства');
GO

--Добавление строк в таблицу Goods
INSERT INTO Goods(Category, ProductName, Price)
    VALUES (1, 'Системный блок', 300),
           (1, 'Монитор', 200),
           (2, 'Смартфон', 250);
GO

--Выборка данных
SELECT * FROM Goods;
SELECT * FROM Categories;
```

Язык **SQL** предназначен для манипулирования данными в реляционных базах данных, определения структуры баз данных и для управления правами доступа к данным в многопользовательской среде.

Синтаксис

Как и во всяком языке программирования, в SQL есть алфавит (допустимые символы), ключевые слова, поддерживаемые типы данных, встроенные константы и функции, операторы:

- SELECT, INSERT, UPDATE, DELETE, DROP, ...
- точные числовые типы, приближенные числовые типы, типы даты и времени, строковые типы
- сложение, вычитание, умножение, деление, остаток от деления
- операторы сравнения, логические операторы
- CAST, TRIM, LOWER, UPPER

Основные типы операторов SQL

- Язык определения данных (Data Definition Language – DDL): операторы создания и управления структурами базы данных
- Язык манипулирования данными (DML): операторы извлечения, вставки, обновления и удаления данных в БД
- Язык определения доступа к данным Data Control Language (DCL): операторы определения доступа к данным.
- Язык управления транзакциями (TCL): используется для контроля обработки транзакций в БД.

Создание базы данных

Создание базы данных

- В системах клиент-сервер, таких как MySQL, для создания базы данных используется команда CREATE DATABASE.
- В однопользовательской системе SQLite база создается автоматически, если файла БД, указанного в параметрах подключения, не существует.
- Для выполнения лабораторных работ по этому курсу для каждой учетной записи уже создана БД.

Создание базы данных

- Для подключения к серверу СУБД необходимо установить приложение-клиент, например
  - **DBeaver** (рекомендуется)
    `https://dbeaver.io`
  - **HeidiSQL**
    `https://www.heidisql.com`
  - **MySQL Workbench**
    `https://www.mysql.com/products/workbench/`
  - **MySQL Shell** (приложение командной строки)
    `http://dev.mysql.com/downloads/shell/`

Подключение из командной строки

```
1 localhost:> mysql -u student -p -h 192.168.0.11
```

Опции команды mysql

- **-u** имя пользователя
- **-p** запросить пароль
- **-h** адрес сервера

```
[pi@raspberrypi ~ $ mysql -u python -p -h 192.168.0.11
[Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 40
Server version: 10.0.38-MariaDB-0+deb8u1 (Raspbian)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Подключение из командной строки

После подключения к СУБД можно создать БД (если есть права на это):

```
mysql> create database db1;
```

или подключиться к существующей базе данных:

```
mysql> use db1;
```

или посмотреть список существующих баз данных:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| db1                |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
```

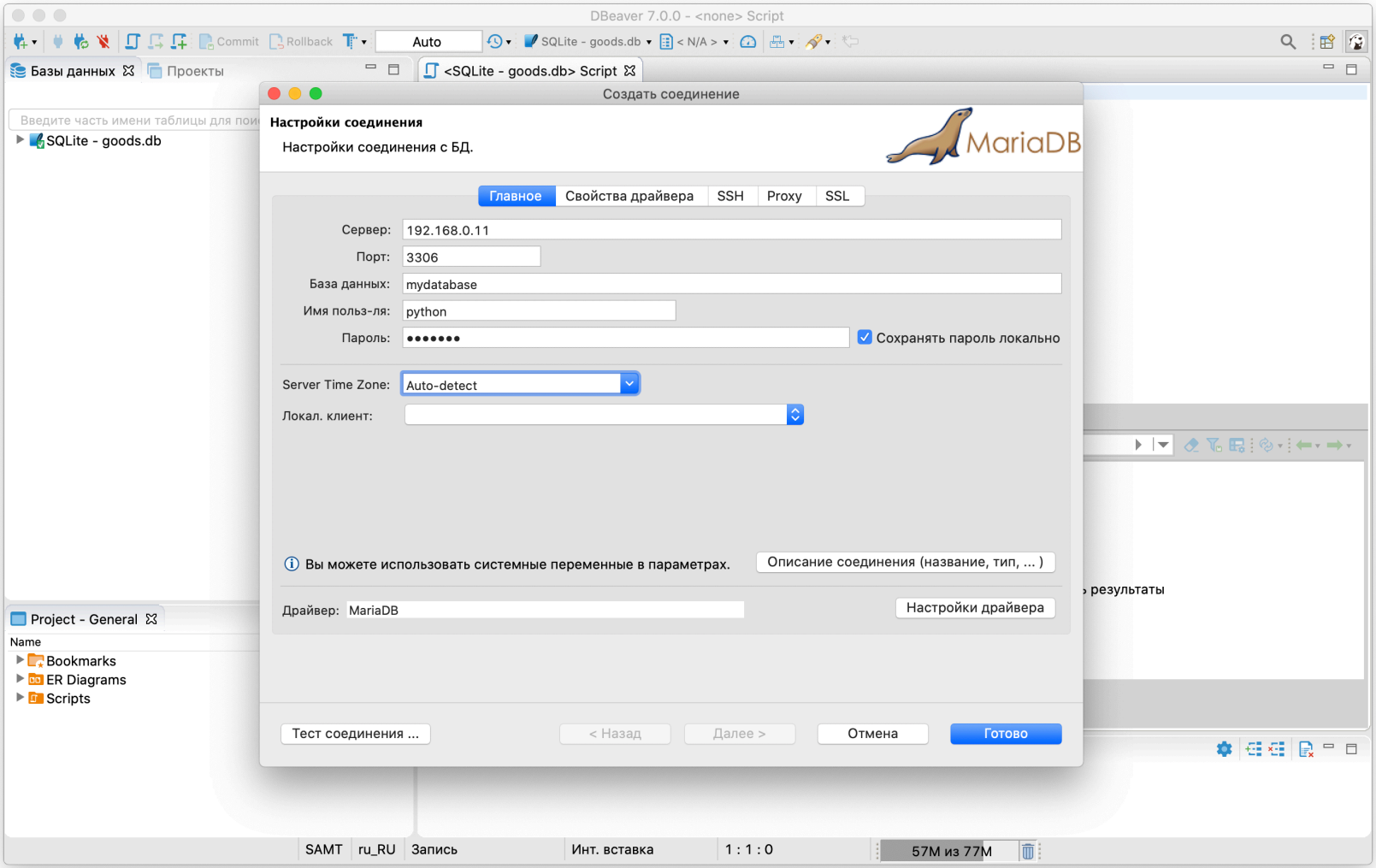
Подключение из среды Python

```
1 import mysql.connector
2
3 mydb = mysql.connector.connect(
4     host="192.168.0.11",
5     user="python",
6     passwd="n9jguqLc5e",
7     database="mydatabase"
8     )
```

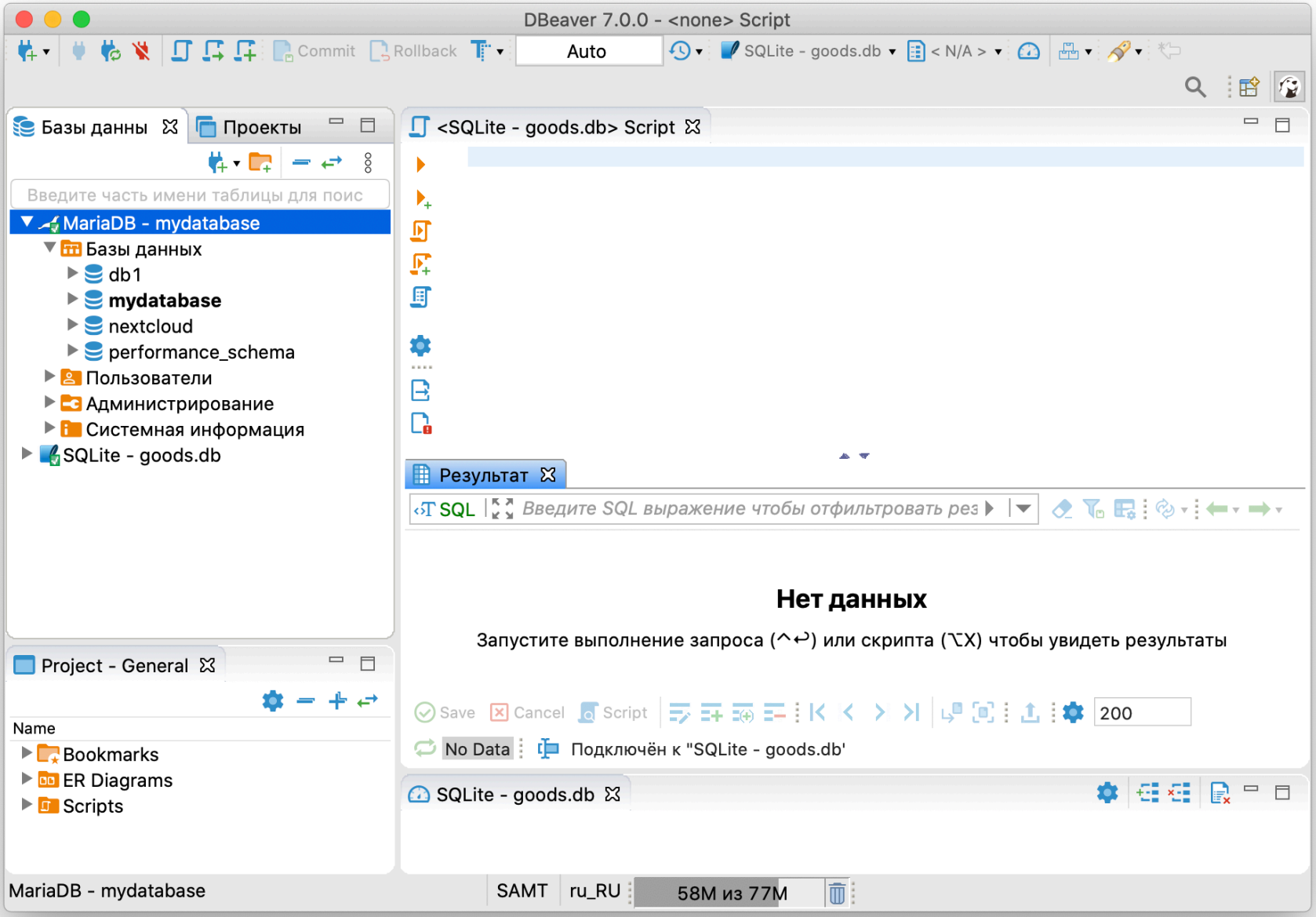
В результате выполнения этого кода создается объект (mydb) – подключение к БД.

Графический клиент DBEAVER

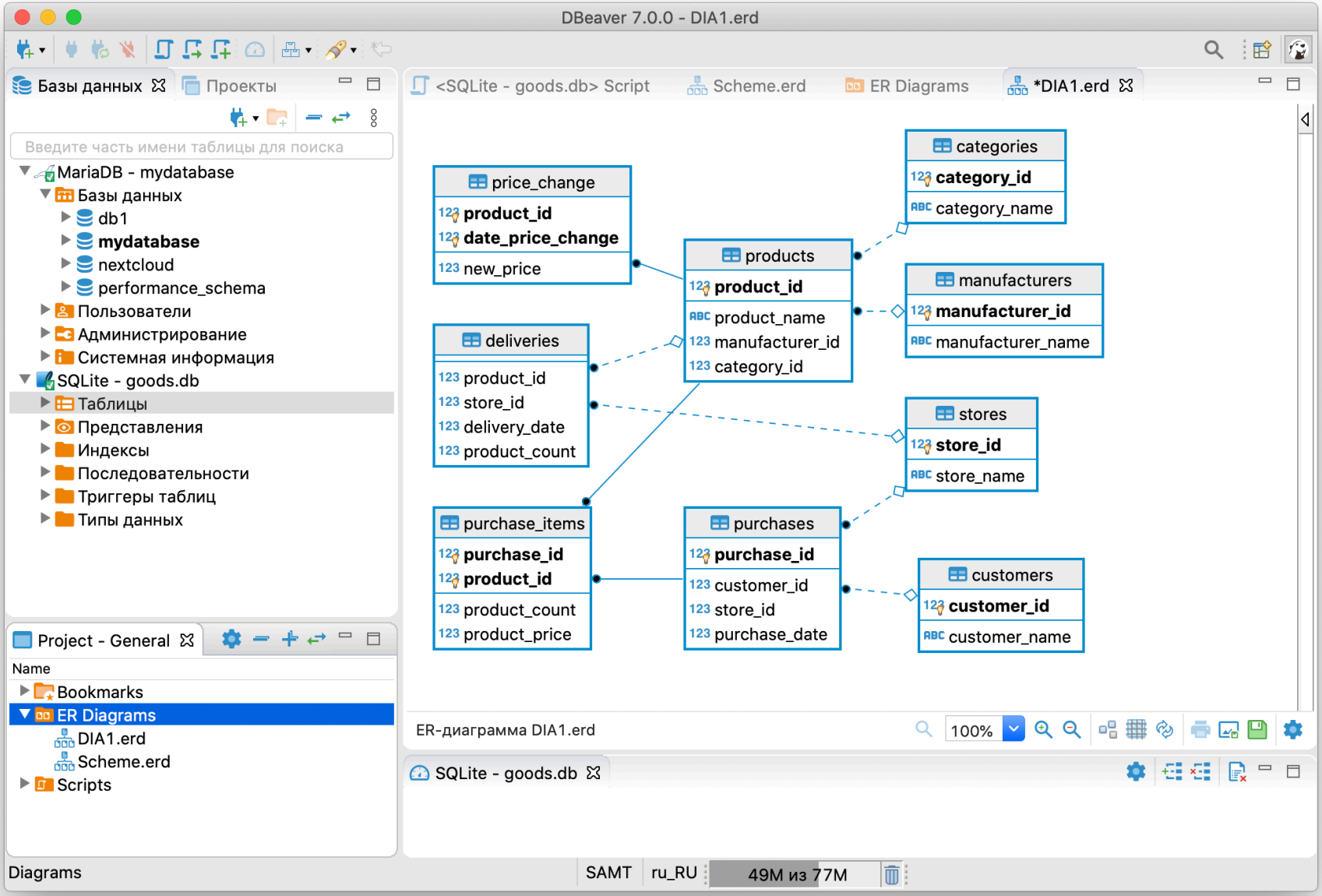
DBeaver — платформенно-независимый клиент баз данных, написанный на Java: `https://dbeaver.io`.



Графический клиент DBEAVER



Графический клиент DBEAVER



Создание таблиц

Создание таблицы

Создание простой таблицы **имя** с тремя столбцами

```
1 CREATE TABLE имя (
2     столбец1 тип1,
3     столбец2 тип2,
4     столбец3 тип3
5 );
```

Таким образом, при создании таблицы необходимо обязательно указать

- **имя** таблицы
- **наименования столбцов**
- **тип** данных в каждом столбце

Пример определения таблицы Students

```
1 CREATE TABLE Students (
2     Student_ID INTEGER,
3     FNAME      VARCHAR(50),
4     LNAME      VARCHAR(50),
5     MNAME      VARCHAR(50)
6 );
```

- **Student_ID** - номер студента (целое число)
- **FNAME** - имя (до 50 символов)
- **LNAME** - фамилия (до 50 символов)
- **MNAME** - отчество (до 50 символов)

Основные строковые типы

- **CHAR(n)**
  Строка фиксированной длины n.
- **VARCHAR(n)**
  Строка переменной длины, но не более n символов.
- **TEXT**
  Текст не более 65 535 символов.

Основные числовые типы

Целые числа (стандарт SQL не определяет размер этих чисел):

- **SMALLINT**
- **INTEGER**
- **BIGINT**

Вещественные числа

- **DECIMAL(p,s)**
  p – общее количество цифр,
  s – количество цифр после запятой.
- **REAL**
- **FLOAT**

Дата и время

- **DATE** (день, месяц и год)
- **TIME** (часы, минуты и секунды)
- **TIMESTAMP** (день, месяц, год, часы, минуты и секунды)

Некоторые числовые типы MySQL

- **TINYINT**
  1 байт, от -128 до 127
- **SMALLINT**
  2 байта, от -32768 до 32767
- **INT**
  4 байта, от -2147483648 до 2147483647
- **BIGINT**
  8 байт BIGINT,
  от -9223372036854775808 до 9223372036854775807
- **SERIAL** псевдоним для
  BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE

Некоторые числовые типы MySQL

- **FLOAT**
  4 байта, от $-3.4... \cdot 10^{38}$ до $3.4... \cdot 10^{38}$
- **DOUBLE**
  8 байта, от $-1.79... \cdot 10^{308}$ до $1.79... \cdot 10^{308}$
- **REAL** синоним **DOUBLE**
- **DECIMAL(M,D)**
  max(M) = 65 (по умолчанию: 10)
  max(D) = 30 (по умолчанию: 0)
  DECIMAL(5,2) : -999.99 до 999.99

Типы даты и времени MySQL

- **DATE** (дата)
  YYYY.MM.DD: от 1000-01-01 до 9999-12-31
- **DATETIME** (дата и время)
  YYYY.MM.DD HH.MM.SS:
  от 1000-01-01 00:00:00 to 9999-12-31 23:59:59
- **TIMESTAMP**
  от 1970-01-01 00:00:01 UTC до 2038-01-09 03:14:07 UTC
  хранится как количество секунд с 1970-01-01 00:00:00 UTC

Пример таблицы

В Самарском университете идентификатором студента может быть номер зачетной книжки, который представляет собой строку вида 2017-01011. В это случае определение таблицы для хранения информации о студентах может иметь вид:

```
CREATE TABLE Students (
    Student_ID CHAR(10),
    FNAME      VARCHAR(50),
    LNAME      VARCHAR(50),
    MNAME      VARCHAR(50)
);
```

ENUM – перечисляемые значения

```
1 CREATE TABLE buyers (
2    email VARCHAR(25) NOT NULL,
3    fname VARCHAR(25) NOT NULL,
4    lname VARCHAR(25) NOT NULL,
5    sex ENUM ('F','M')
6 );
```

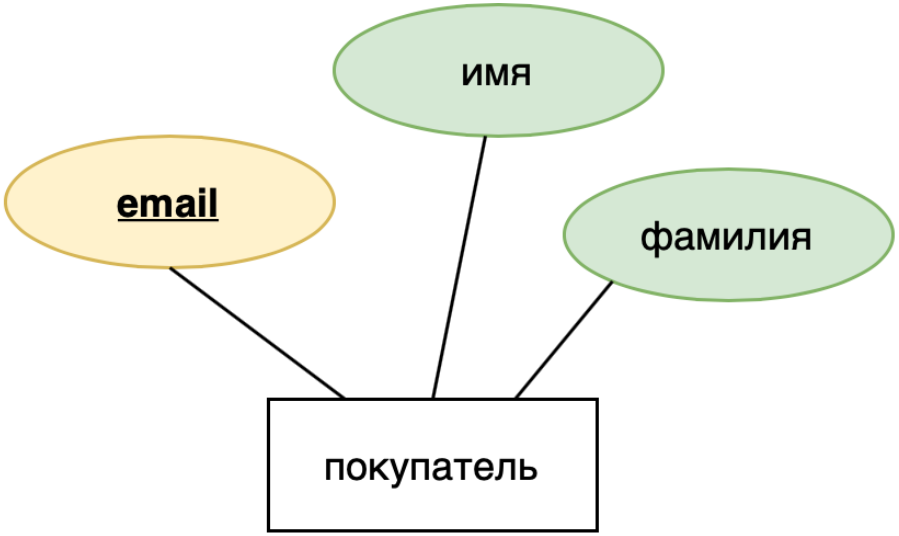
Поле **sex** (тип – строка) принимает два возможных значения **M**ale или **F**emale, а также может иметь значение NULL.

Обеспечение целостности данных

Первичный ключ

- **Первичный ключ** – набор атрибутов, уникально идентифицирующий запись в таблице.
- Если первичный ключ состоит из единственного атрибута, его называют **простым ключом**.
- Если первичный ключ состоит из двух и более атрибутов, его называют **составным ключом**.

PRIMARY KEY – Первичный ключ



BUYERS

EMAIL	FNAME	LNAME
sarah_connor@gmail.com	Connor	Sarah
ann_connor@gmail.com	Connor	Ann

PRIMARY KEY – Первичный ключ

В таблице buyers поле email является первичным ключом:

```
1 CREATE TABLE buyers (
2    email VARCHAR(25) NOT NULL PRIMARY KEY,
3    fname VARCHAR(25),
4    lname VARCHAR(25) NOT NULL
5 );
```

- **PRIMARY KEY** – поле email является ключевым и в таблице не может быть двух строк с одинаковым значением этого поля
- **NOT NULL** – поле email не может быть пустым

PRIMARY KEY – Первичный ключ

Первичный ключ может быть объявлен при помощи ключевого слова **CONSTRAINT**:

```
1 CREATE TABLE buyers (
2       email VARCHAR(25) NOT NULL,
3       fname VARCHAR(25),
4       lname VARCHAR(25) NOT NULL,
5       CONSTRAINT PK_buyer PRIMARY KEY (email, lname)
6 );
```

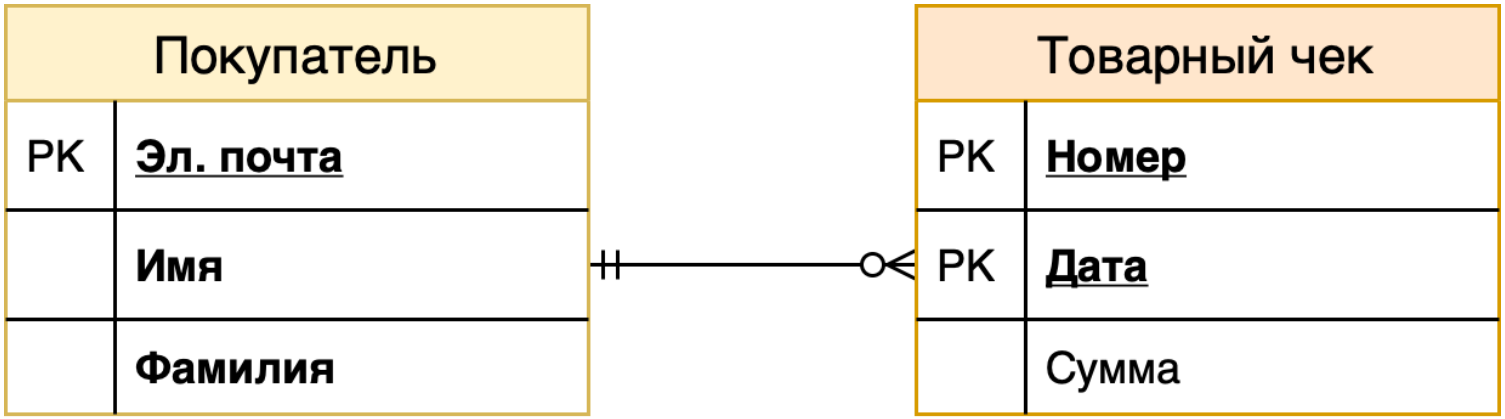
Таким способом может быть объявлен составной первичный ключ. В этом примере ”покупатели”идентифицируются (различаются) по адресу электронной почты И фамилии.

FOREGIN KEY – Внешний ключ

- FOREIGN KEY (в переводе с английского языка - «внешний ключ») - это ограничение целостности базы данных, которое используется для связи двух таблиц по значению одного или нескольких полей.
- В таблице, которая является «дочерней» в отношении другой «родительской» таблицы, столбец FOREIGN KEY ссылается на PRIMARY KEY или UNIQUE KEY в ”родительской”таблице.
- Это позволяет обеспечить целостность данных и поддерживать связь между таблицами.
- Если запись в «родительской» таблице удаляется или изменяется, то FOREIGN KEY гарантирует, что связанные записи в «дочерней» таблице также будут удалены или изменены.

FOREGIN KEY – Внешний ключ

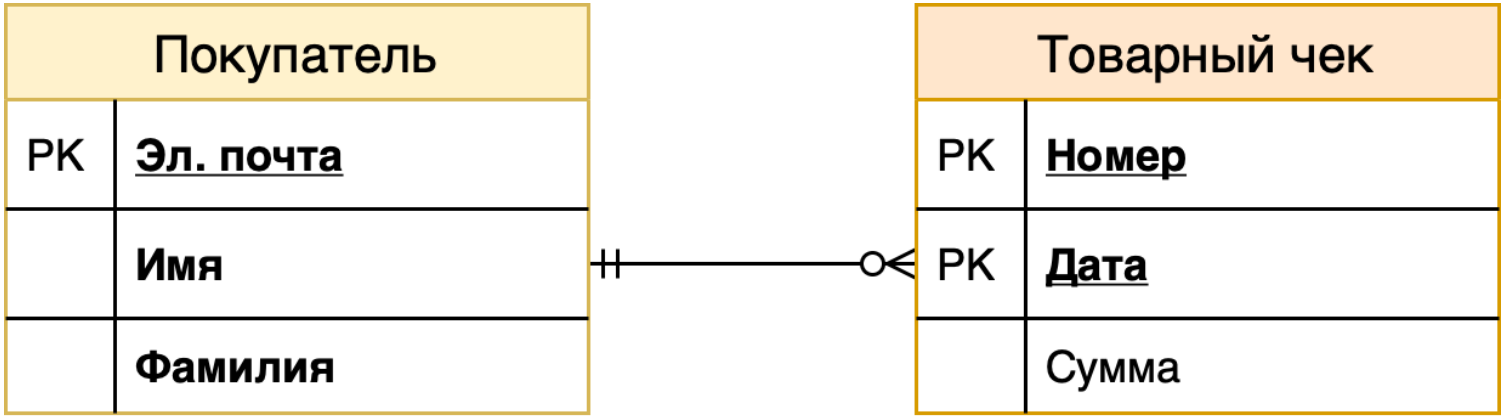
Сущность товарный чек ссылается на сущность Покупатель.



Отношение между сущностями Покупатель – Товарный чек **один ко многим**: у товарного чека может быть только один плательщик (покупатель), у покупателя может быть несколько товарных чеков.

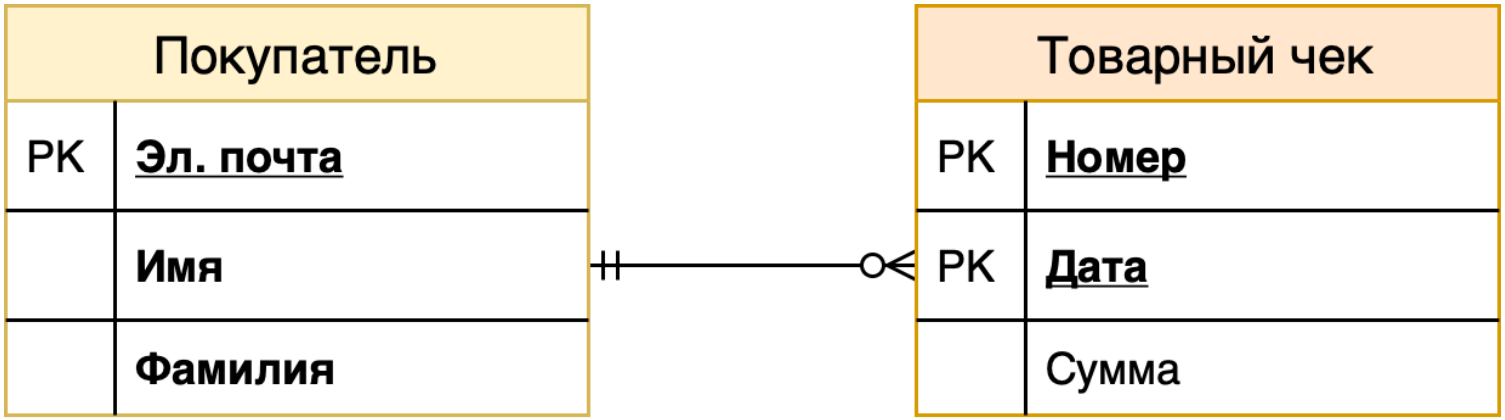
Таблица ПОКУПАТЕЛЬ

Главным ключом таблицы Покупатель является адрес электронной почты.




```sql
CREATE TABLE buyers (
   email VARCHAR(25) NOT NULL,
   fname VARCHAR(25),
   lname VARCHAR(25) NOT NULL,
   PRIMARY KEY(email)
);
```

Таблица ТОВАРНЫЙ ЧЕК




```sql
CREATE TABLE sales_slip (
   id INTEGER NOT NULL,
   sale_date DATE,
   buyer_email VARCHAR(25) NOT NULL,
PRIMARY KEY PK_sales_slip (id, sale_date),
FOREIGN KEY (buyer_email) REFERENCES buyers (email)
);
```

FOREGIN KEY – Внешний ключ

```
1  CREATE TABLE sales_slip (
2    id INTEGER NOT NULL,
3    sale_date DATE,
4    buyer_email VARCHAR(25) NOT NULL,
5  PRIMARY KEY PK_sales_slip (id, sale_date),
6  FOREIGN KEY (buyer_email) REFERENCES buyers (email)
7  );
```

- Поле buyer_email ссылается на поле email таблицы buyers.
- Это поле не может быть пустым (NULL): у каждого чека должен быть хотя бы один плательщик.
- В таблицу **sales_slip** невозможно внести информацию о счёте без ссылки на покупателя из таблицы **buyers**.

Естественный и суррогатный ключ

- **Суррогатный ключ** – это дополнительное поле, добавленное к уже имеющимся полям таблицы, которое служит первичным ключом. Значение этого поля не образуется на основе каких-либо других данных из БД, а генерируется искусственно
- Суррогатный ключ — это обычно числовое поле, в которое заносятся значения из возрастающей числовой последовательности

Естественный ключ

PERSONS

ИМЯ	ФАМИЛИЯ	ГОД РОЖДЕНИЯ	АДРЕС
Игорь	Селезнев	1995	...
Полина	Метелкина	2001	...
Филидор	Зелёный	1997	...

PAYMENTS

ДАТА	НОМЕР	ФАМИЛИЯ	ИМЯ	ГОД РОЖДЕНИЯ	СУММА
10.03.2020	15251	Селезнев	Игорь	1995	1525,12
11.03.2020	15254	Метелкина	Полина	2001	5681,51

Естественный ключ

```
1 CREATE TABLE persons (
2  fname VARCHAR(50) NOT NULL,
3  lname VARCHAR(50) NOT NULL,
4  birthdate DATE NOT NULL,
5  address VARCHAR(100) NOT NULL,
6 CONSTRAINT PK_persons PRIMARY KEY PK_PERSONS (fname,
      lname, birthdate)
7 );
```

У это таблицы составной первичный ключ, что делает громоздким организацию связей с другими таблицами.

Естественный ключ

```
1  CREATE TABLE payments (
2   payment_id    INTEGER NOT NULL,
3   payment_date DATE NOT NULL,
4   payer_fname VARCHAR(50) NOT NULL,
5   payer_lname VARCHAR(50) NOT NULL,
6   payer_birthdate DATE NOT NULL,
7  CONSTRAINT PK_payments
8   PRIMARY KEY PK_PAYMENTS (payment_id, payment_date),
9  CONSTRAINT FK_payments
10  FOREIGN KEY (payer_fname,payer_lname,payer_birthdate)
11  REFERENCES persons(fname,lname,birthdate)
12 );
```

Суррогатный ключ

PERSONS

ID	ИМЯ	ФАМИЛИЯ	ГОД РОЖДЕНИЯ	АДРЕС
10	Игорь	Селезнев	1995	...
11	Полина	Метелкина	2001	...
12	Филидор	Зелёный	1997	...

PAYMENTS

ДАТА	НОМЕР	ПЛАТЕЛЬЩИК	СУММА
10.03.2020	15251	10	1525,12
11.03.2020	15254	11	5681,51

AUTO_INCREMENT

Свойство атрибута (столбца) **AUTO_INCREMENT** позволяет автоматически генерировать уникальное значение атрибута при вставке нового значения.

```
1 CREATE TABLE sales_slip (
2   id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL,
3   sale_date DATE,
4   buyer_email VARCHAR(25) NOT NULL,
5 FOREIGN KEY (buyer_email) REFERENCES buyers (email)
6   ON UPDATE CASCADE
7   ON DELETE CASCADE
8 );
```

Суррогатный ключ

Добавим автоинкрементное поле **ID**, а для обеспечения уникальности имени, фамилии и даты рождения используем **CONSTRAINT ... UNIQUE**

```
1 CREATE TABLE persons (
2   id INTEGER NOT NULL AUTO_INCREMENT,
3   fname VARCHAR(50) NOT NULL,
4   lname VARCHAR(50) NOT NULL,
5   birthdate DATE NOT NULL,
6   address VARCHAR(100) NOT NULL,
7 CONSTRAINT PK_persons PRIMARY KEY PK_PERSONS (id),
8 CONSTRAINT UC_name_birth UNIQUE (fname, lname, birthdate)
9 );
```

Суррогатный ключ

Таблица **payments** будет содержать числовое поле – ссылку на **ID** из таблицы **persons**

```
1 CREATE TABLE payments (
2  payment_id INTEGER NOT NULL,
3  payment_date DATE NOT NULL,
4  payer_id INTEGER NOT NULL,
5 CONSTRAINT PK_payments
6 PRIMARY KEY PK_PAYMENTS (payment_id, payment_date),
7 CONSTRAINT FK_payments
8 FOREIGN KEY (payer_id) REFERENCES persons (id)
9 );
```

Суррогатный ключ

В СУБД есть специальный тип поля **SERIAL**, который является псеводимом записи **BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE**

```
1 CREATE TABLE persons (
2   id SERIAL,
3   fname VARCHAR(50) NOT NULL,
4   lname VARCHAR(50) NOT NULL,
5   birthdate DATE NOT NULL,
6   address VARCHAR(100) NOT NULL,
7 CONSTRAINT PK_persons PRIMARY KEY PK_PERSONS (id),
8 CONSTRAINT UC_name_birth UNIQUE (fname, lname, birthdate)
9 );
```

Ограничения на значения столбцов

Для ограничения величины атрибутов используется ключевое слово **CHECK**.

```
1  CREATE TABLE sales_slip (
2    id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL,
3    sale_date DATE,
4    buyer_email VARCHAR(25) NOT NULL,
5    sum_total NUMERIC(12,2) CHECK (sum_total>0)
6  FOREIGN KEY (buyer_email) REFERENCES buyers (email)
7  )
8
```

Поле **sum_total** должно быть больше нуля, но может быть и NULL, т.к. нет требования **NOT NULL**.

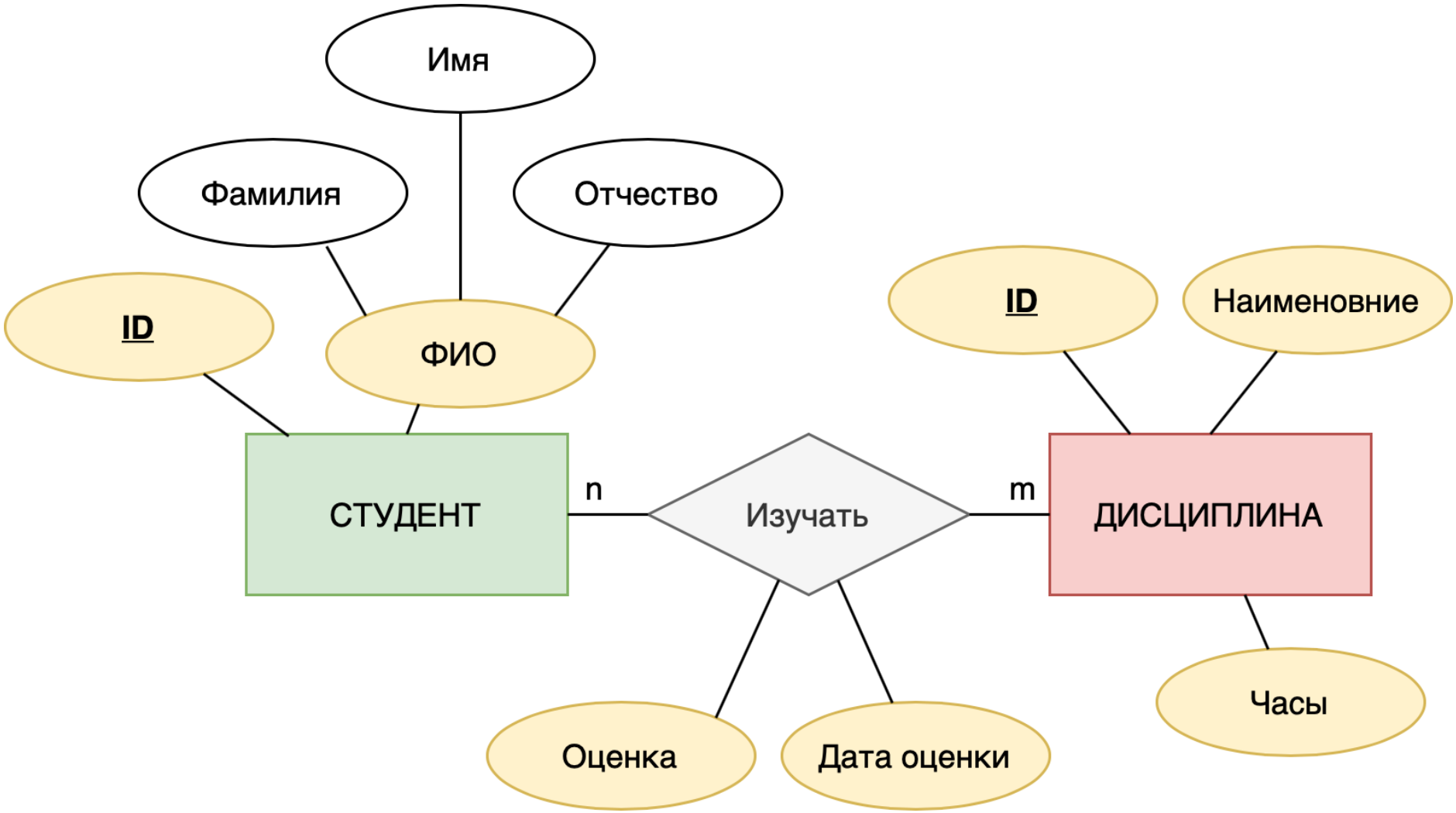
Ограничения на значения столбцов

```sql
CREATE TABLE students
(
   id SERIAL PRIMARY KEY,
   name VARCHAR(50) NOT NULL,
   birthdate DATE CHECK (birthdate > '1990-01-01'),
   enrollmentdate DATE,
   CONSTRAINT enrollment_gt CHECK (enrollmentdate >
     birthdate)
);
```

```sql
INSERT INTO students (name, birthdate, enrollmentdate)
VALUES ("Иночкин К. С.", "1991-12-25", "2015-08-01");
```

```sql
INSERT INTO students (name, birthdate, enrollmentdate)
VALUES ("Митрофанова Е. В.", "1991-10-05", "1915-08-01");
```

Многие ко многим



Промежуточная сущность

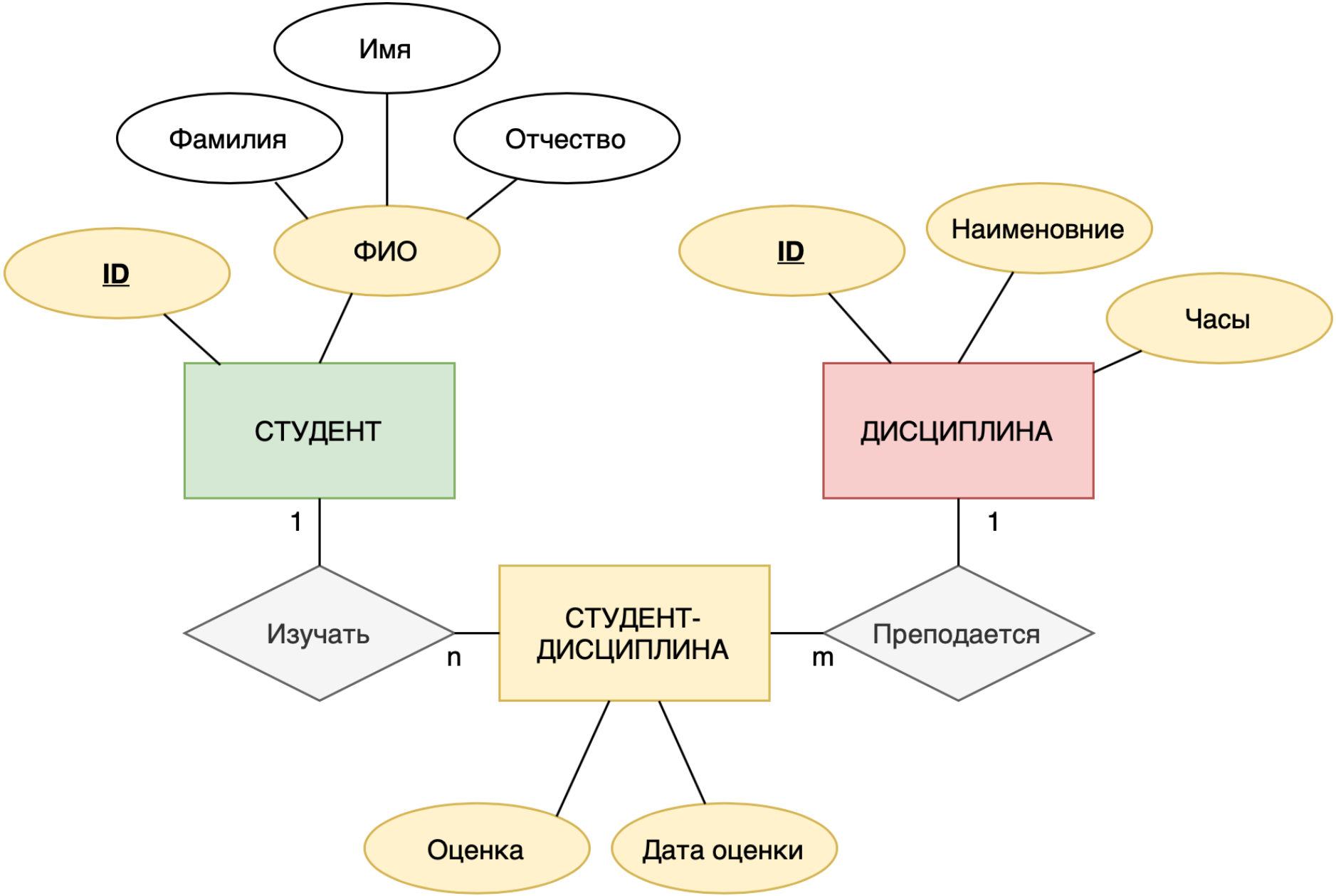


Таблица связей n:m

Отношение многие ко многим

Студент = Идентификатор, Фамилия, Имя, Отчество

```sql
CREATE TABLE students (
   id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL,
   fname VARCHAR(20),
   mname VARCHAR(20),
   lname VARCHAR(20)
);
```

Отношение многие ко многим

Дисциплина = Идентификатор, Наименование, Количество часов

```sql
CREATE TABLE subjects (
   id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL,
   name  VARCHAR(100),
   hours INTEGER UNSIGNED NOT NULL
);
```

Отношение многие ко многим

Таблица, связывающая дисциплину со студентом:

```sql
CREATE TABLE subject2student (
   student_id INTEGER NOT NULL,
   subject_id INTEGER NOT NULL,
   mark       SMALLINT UNSIGNED,
   mark_date  DATE,
FOREIGN KEY (student_id) REFERENCES students (id),
FOREIGN KEY (subject_id) REFERENCES subjects (id)
);
```

Пример (SQLite):

https://colab.research.google.com/drive/1v913bNDvrWc0EeuwfIxwj-72Gk1FO4ik

ON UPDATE, ON DELETE

При помощи ключевых слов ON DELETE и ON UPDATE можно определить действия которые будут выполнятся в зависимой таблице (sales_slip) при изменении ключевого поля таблицы buyers

```
FOREIGN KEY (имя_столбца,...)
REFERENCES имя_таблицы (имя_столбца, ...)
[ON DELETE действие]
[ON UPDATE действие]
```

ON UPDATE, ON DELETE

Действие	Описание
CASCADE	Изменение значения первичного ключа приводит к автоматическому изменению соответствующих значений внешнего ключа
SET NULL	При изменении значения или удалении первичного ключа все значения в связанных с внешним ключом колонках устанавливаются в NULL (появляются «брошенные» строки)
RESTRICT	Режим по умолчанию. Внешний ключ воспринимает только значения первичного ключа или NULL.
NO ACTION	То же самое, что и RESTRICT

ON UPDATE, ON DELETE

```
1 CREATE TABLE sales_slip (
2   id INTEGER PRIMARY KEY NOT NULL,
3   sale_date DATE,
4   buyer_email VARCHAR(25) NOT NULL,
5 FOREIGN KEY (buyer_email) REFERENCES buyers (email)
6   ON UPDATE CASCADE
7   ON DELETE CASCADE
8 )
```

- Изменение поля email в таблице buyers приведет к обновлению соответствующего значения buyer_email
- Удаление записи в таблице buyers приведет к удалению всех записей с удаленным значением buyer_email

Индексы

Индекс

Ключевое поле в таблице buyers имеет строковый тип VARCHAR

```
1 CREATE TABLE buyers (
2    email VARCHAR(25) NOT NULL,
3    fname VARCHAR(25) NOT NULL,
4    lname VARCHAR(25) NOT NULL,
5 PRIMARY KEY (email)
6 );
```

Если для поиска адреса эл. почты использовать обычный метод сравнения двух строк с последовательным просмотром всех строк таблицы, то это приведет к большим затратам времени

Индекс

Ключевое поле в таблице buyers имеет строковый тип VARCHAR

```
1 CREATE TABLE buyers (
2    email VARCHAR(25) NOT NULL,
3    fname VARCHAR(25) NOT NULL,
4    lname VARCHAR(25) NOT NULL,
5 PRIMARY KEY (email)
6 );
```

При создании таблицы buyers будет автоматически создан индекс: при вставке нового значения в таблицу строка (эл. адрес) будет дополнительно преобразована в цифровой код (хэш), поиск заданной строки будет быстро выполнятся по её цифровому коду.

Индексы

- Индексы в БД позволяют выполнять быстрый поиск данных
- В таблице автоматически создается индекс главного ключа и внешнего ключа
- В таблице можно создавать пользовательские индексы

CREATE INDEX

Для быстрого поиска покупателя с заданной фамилией в большой таблице, целесообразно объявить индекс для поля fname:

```
1 CREATE INDEX fname_idx ON buyers(fname);
```

Если требуется уникальность значений индексируемого поля, необходимо использовать ключевое слово UNIQUE:

```
1 CREATE UNIQUE INDEX fname_idx ON buyers(fname);
```

Изменение таблиц

Изменение таблиц

Для изменения атрибутов и свойств таблицы используется инструкция **ALTER TABLE**, например для добавления столбца в таблицу:

```
1 ALTER TABLE ORDERS ADD COLUMN PRICE NUMERIC(9,2)
```

Изменение типа столбца, если он уже есть, но другого типа

```
1 ALTER TABLE ORDERS ALTER COLUMN PRICE NUMERIC(9,2)
```

Удаление столбца

```
1 ALTER TABLE ORDERS DROP COLUMN PRICE
```

См. https://www.w3schools.com/sql/sql_alter.asp

Изменение таблиц

Добавление первичного ключа

```
1 ALTER TABLE ORDERS ADD PRIMARY KEY (ID);
```

Добавление требования уникальности

```
1 ALTER TABLE Persons
2       ADD CONSTRAINT UC_Person
3       UNIQUE (ID, LastName);
```

См. https://www.w3schools.com/sql/sql_alter.asp

Изменение таблиц

Изменение таблиц может привести к потере существующих данных, поэтому до всех изменений следует создать резервную копию базы.

Удаление таблицы

Удалить таблицу ORDERS

```
1 DROP TABLE ORDERS
```

Удалить таблицу ORDERS, если она существует

```
1 DROP TABLE IF EXISTS ORDERS
```

Список использованных источников

- Хомоненко А. Д., Цыганков В. М., Мальцев М. Г. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А. Д. Хомоненко. – 6-е изд., доп. - СПб.: КОРОНА-Век, 2009. – 736 с.
- Осипов Д. Л. Технологии проектирования баз данных. – М.: ДМК Пресс, 2019.
- SQL Учебник

  `https://schoolsw3.com/sql/index.php`
- SQL Tutorial

  `https://www.tutorialspoint.com/sql/index.htm`

https://classmech.ru/pages/databases/