Основы SQL

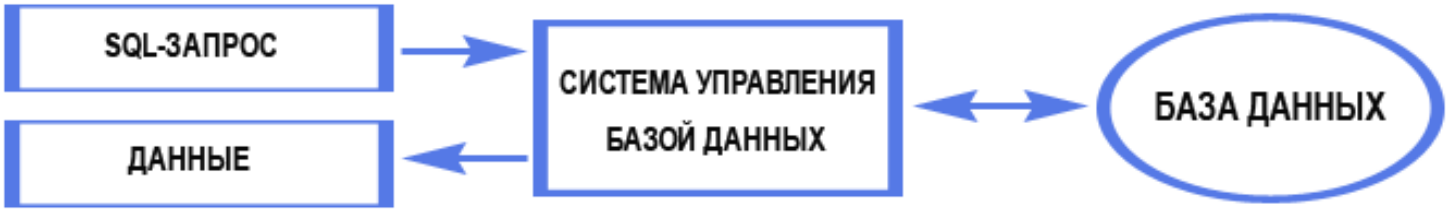
Материалы

- http://www.mysql.ru/docs/man/Introduction.html

SQL

- **SQL** – язык запросов, позволяющий создавать/изменять/удалять базы данных, а также искать в них данные

- Примерно так выглядит взаимодействие пользователей с базой данных



Стандарт и реализации SQL

- SQL – является стандартом, т.е. четко задокументирован сам язык, его синтаксис и конструкции и т.д.
- Но реализации SQL часто отходят от стандарта
- Часть функций стандарта не реализована, могут быть добавлены дополнительные функции, синтаксис и возможности
- Каждая СУБД имеет свои особенности, поэтому предпочтительно по максимуму соответствовать стандарту
- Это полезно если понадобится перенести БД на другую СУБД

Синтаксис SQL

- **Пример** – запрос на получение товаров с ценой больше 30:
- SELECT * FROM products
  WHERE price > 30;

- Вообще говоря, SQL **регистронезависимый** – все можно писать в любом регистре – и ключевые слова, и имена таблиц и полей
- select * from Products
  where Price > 30;

- Команды завершают точкой с запятой

Литералы строк

- Согласно стандарту SQL, литералы строк заключаются в одинарные кавычки
- SELECT * FROM products
  WHERE name = ‘Intel Core i5’;

- В MySQL также можно использовать и двойные кавычки, но лучше так не делать, т.к. это работает не во всех других СУБД
- SELECT * FROM products
  WHERE name = “Intel Core i5”;

Экранирование кавычки

- Если внутри строки нужен символ одинарной кавычки ‘ то его нужно удвоить (стандарт SQL)
- SELECT * FROM products
  WHERE name = ’Intel Core ‘’i5’’’;

- Либо можно поставить перед кавычкой обратный слэш (не стандарт)
- SELECT * FROM products
  WHERE name = ’Intel Core \’i5\’’;

Комментарии

- Однострочные:
  - От -- до конца строки
    После -- должен быть хотя бы 1 пробел
  - От символа # до конца строки (не стандарт)
- Многострочные:
  - /* Многострочный комментарий */

- Примеры:
  - SELECT * FROM table1; # Комментарий
    SELECT * FROM table2; -- Комментарий
    /* SELECT * FROM table3; */

Консоль

- Кроме Workbench, работать с MySQL можно через консоль
- Это удобно, если нужно написать некоторый скрипт или нужно работать с MySQL через SSH

- Залогиниться можно так:
  mysql -uroot -p1234
- После –u идет логин пользователя, после –p идет пароль

- Либо так:
  mysql -uroot -p
- Тогда после этого нужно будет ввести пароль

Создание, удаление и редактирование БД и таблиц

Создание базы данных

- CREATE DATABASE имя_базы;

- Пример:
- CREATE DATABASE shop;

- Команды, начинающиеся с CREATE используются для создания объектов БД – создания БД, таблиц, индексов и т.д.

Допустимые имена объектов БД

- Имена БД, таблиц, столбцов и других объектов БД должны удовлетворять правилам
- Во-первых, имена могут быть заключенными в кавычки и не заключенными в кавычки. Причем в качестве кавычки берется не обычная кавычка, а ` (где буква Ё)

- Если имя не является зарезервированным словом и не содержит специальных символов, то его можно не заключать в кавычки
- Иначе – можно обращаться по имени, только заключая его в кавычки

Допустимые имена объектов БД

- Обычное имя таблицы, кавычки не обязательны
- SELECT * FROM products;

- Обычное имя таблицы, но заключили в кавычки
- SELECT * FROM `products`;

- Имя таблицы – зарезервированное слово
- SELECT * FROM `table`;

Команда USE

- MySQL выполняет команды в контексте некоторой БД
- Чтобы выбрать нужную базу для последующих операций, нужно использовать оператор USE
- USE имя_базы;

- Например:
- USE shop;
  # дальше идут команды

Создание таблицы

- CREATE TABLE имя_таблицы
  (
  имя_столбца1 тип,
  имя_столбца2 тип,
  и т. д.
  );

- Пример:
- USE shop;
  CREATE TABLE products
  (
  id INT,
  name VARCHAR(255)
  );

Использовали USE, чтобы таблица создалась в нужной базе

Но прямо так таблицы обычно не создают

Задание первичного ключа

- При создании таблицы мы не указали первичный ключ
- Вообще говоря, это можно сделать потом отдельно, но чаще всего его указывают сразу
- ```
  USE shop;
  CREATE TABLE products
  (
     id INT  AUTO_INCREMENT  PRIMARY KEY,
     name VARCHAR(255)
  );
  ```
- Тут еще ключу добавлен параметр AUTO_INCREMENT
- Он заставляет БД автоматически генерировать значение поля при вставке новых строк
- Это очень удобно для первичного ключа
  ```

Задание первичного ключа

- Есть еще другой вариант:
- USE shop;

```
CREATE TABLE products
(
   id INT AUTO_INCREMENT,
   name VARCHAR(255),
   PRIMARY KEY (id)
);
```

- Этот синтаксис хорош тем, что позволяет задать в качестве первичного ключа совокупность полей, а не одно поле
- Например, PRIMARY KEY (id1, id2)

Понятие NULL

- В SQL каждый тип данных может принимать специальное значение NULL
- Его смысл – отсутствие данных

- При создании столбцов таблицы можно указать, могут они принимать NULL или нет
- Если столбец не позволяет вставлять NULL, и попытаться это сделать, то будет ошибка и операция не выполнится

NULL / NOT NULL

- ```
  USE shop;

  CREATE TABLE products
  (
     id INT AUTO_INCREMENT,
     name VARCHAR(255) NOT NULL,
     PRIMARY KEY (id)
  );
  ```

- Запретили NULL для столбца **name** при помощи NOT NULL
- Если ничего не написать, то столбец позволяет значение NULL
- Можно разрешить NULL явно: name VARCHAR(255) NULL
  ```

Другие возможности CREATE TABLE

- Ограничение уникальности для столбца - UNIQUE:
  - name VARCHAR(255) NOT NULL **UNIQUE**
  - После этого в этом столбце не допускаются повторяющиеся значения

- Значение по умолчанию для столбца – DEFAULT:
  - price INT NOT NULL **DEFAULT 0**
  - Указанное значение будет использоваться если при вставке не укажут другое значение

Другие возможности CREATE TABLE

- Проверка корректности при вставке и изменении данных - CHECK:
  - price INT NOT NULL CHECK (price > 0)
  - Если условие будет ложным, то операция вставки/изменения не выполнится, будет ошибка
  - В MySQL CHECK не поддерживается – его можно написать, но проверки выполняться не будут

Удаление базы данных и таблицы

- DROP DATABASE shop;

- Команды, начинающиеся с DROP используются для удаления объектов БД – БД, таблиц, индексов и т.д.

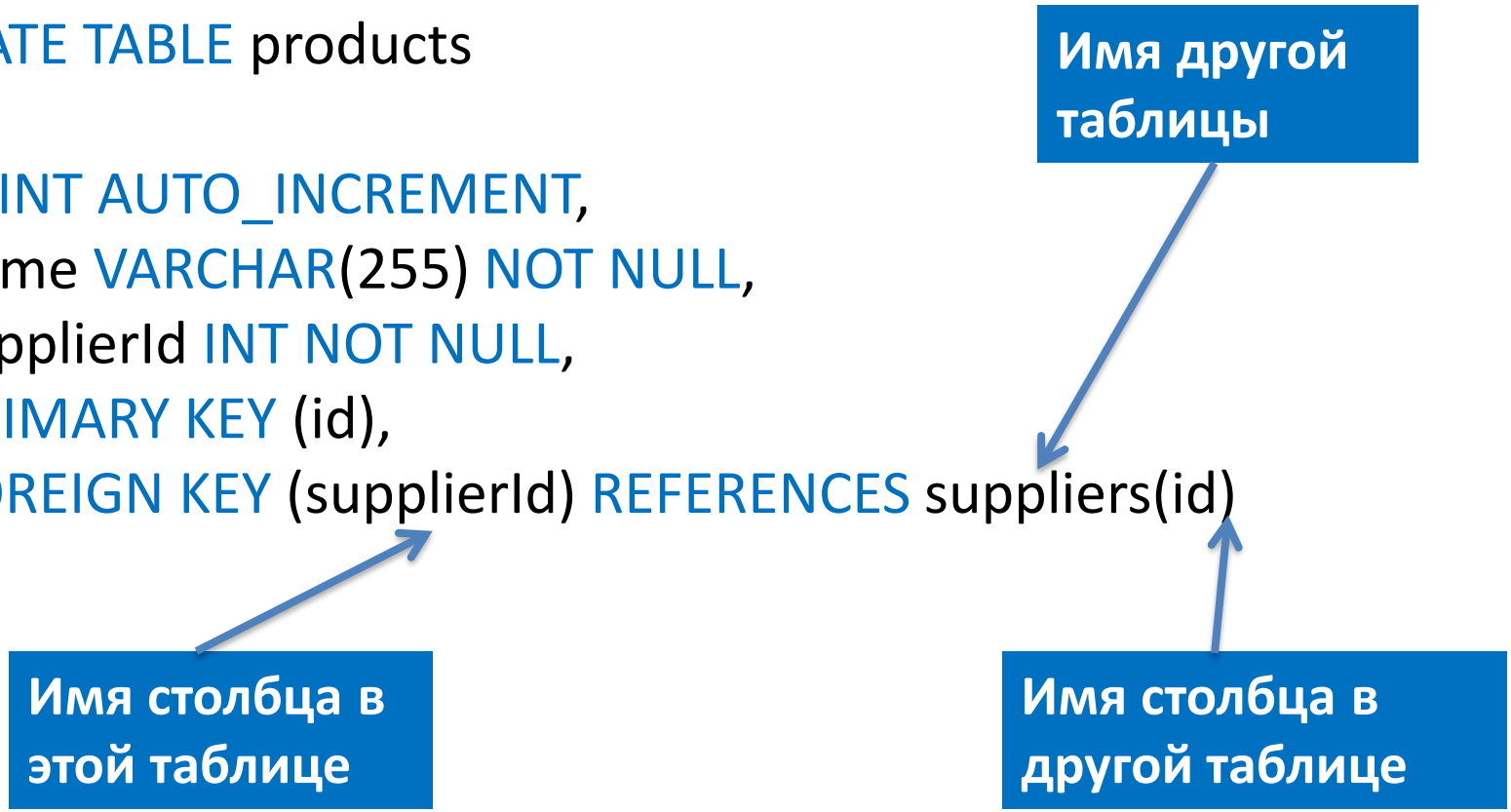
- Удаление таблицы
- USE shop;
  DROP TABLE products;

Внешние ключи

- Как связать 2 таблицы между собой?
- Для этого есть внешние ключи
- Создать внешний ключ можно либо сразу в CREATE TABLE, либо потом отдельно

Внешние ключи

- Допустим, есть отдельная таблица поставщиков продуктов, и мы для продукта хотим хранить поставщика (пусть он один для продукта)
-
```
USE shop;
CREATE TABLE products
(
    id INT AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    supplierId INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (supplierId) REFERENCES suppliers(id)
);
```



Внешний ключ

- Если вы добавили FOREIGN KEY, то СУБД добавляет некоторые проверки корректности данных
- Например, если вы будете заполнять поле **supplierId** числом 5, а в таблице **suppliers** не будет строки с **id** = 5, то вставка не произойдет – выдастся ошибка
- Кроме того, вы не сможете удалить поставщика, если есть товары, которые на него ссылаются
- Сначала нужно будет удалить эти связи (например, задать товарам другого поставщика или NULL – если это разрешено) или удалить сами товары (если это подходит по смыслу)

Порядок создания таблиц

- Если у вас БД из нескольких таблиц, то очень важен порядок создания таблиц
- В объявлении внешних ключей нельзя ссылаться на таблицы, которые еще не существуют
- И, в целом, это общее правило – нельзя работать с тем, чего еще нет

ON UPDATE, ON DELETE

- При настройке внешнего ключа можно указывать варианты поведения при удалении и обновлении строки из главной таблицы (на которую ссылается внешний ключ)
-
```
USE shop;
CREATE TABLE products
(
    id INT AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    supplierId INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (supplierId) REFERENCES suppliers(id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

ON DELETE

- Варианты для ON DELETE:
  - NO ACTION – выдает ошибку если кто-то ссылается на строку главной таблицы, удаление не делается
  - CASCADE – **каскадное удаление**. Если из главной таблицы удаляется строка, то удаляются все связанные строки других таблиц
  - RESTRICT – аналогично NO ACTION
  - SET NULL – выставить NULL во внешнем ключе при удалении строки из главной таблицы
  - SET DEFAULT – выставить значение по умолчанию или NULL, если значения по умолчанию нет

ON UPDATE

- Варианты для ON UPDATE. Задает действие, которое надо выполнить, если строке в главной таблице меняют первичный ключ
  - NO ACTION – выдает ошибку если кто-то ссылается на строку главной таблицы, обновление не делается
  - CASCADE – **каскадное обновление**. Если в главной таблице меняется первичный ключ, то меняется внешний ключ всех связанных строк других таблиц
  - RESTRICT – аналогично NO ACTION
  - SET NULL – выставить NULL во внешнем ключе при изменении первичного ключа строки из главной таблицы
  - SET DEFAULT – выставить значение по умолчанию или NULL, если значения по умолчанию нет

Типы данных

Типы данных

- Типы можно разделить на группы:
  - Числовые
  - Строковые
  - Дата и время

Числовые типы

- Целые числа:

Тип	Размер	Диапазон
TINYINT	1 байт	-128..127
SMALLINT	2 байта	-32768..32767
MEDIUMINT	3 байта	-8388608..8388607
INT или INTEGER	4 байта	-2147483648..2147483647
BIGINT	8 байт	$-2^{63}..2^{63}-1$

- Вещественные числа с плавающей точкой:

Тип	Размер
FLOAT	4 байта
DOUBLE	8 байт

Тип DECIMAL

- DECIMAL(M, D) – это вещественное число с фиксированным количеством цифр до запятой и после запятой
- M – общее количество цифр, в MySQL максимум 65
- D – количество цифр после запятой, в MySQL максимум 30

Строковые типы

- Перечислены не все типы и не все их возможности

Тип	Особенности
CHAR(M) M от 0 до 255	Строка фиксированной длины. Если длина строки меньше указанной, то она все равно будет занимать в памяти как строка указанной длины
VARCHAR(M) M от 0 до 65535	Строка переменной длины. Занимает почти ровно, сколько требуется – более эффективно по памяти. Дополнительно хранит внутри себя длину строки – 1 байт если M от 0 до 255, иначе 2 байта

- Рекомендуется использовать VARCHAR, а CHAR только если данные действительно фиксированной длины
- Если при вставке данных превысить длину поля, то операция упадет

Дата и время

- Перечислены не все типы и не все их возможности

Тип
DATE
TIME
DATETIME
TIMESTAMP

- Значения дат записывают в виде строк, обычно в таком формате: ‘2021-12-31’ (год-месяц-день)
- Время записывают в таком формате: ’12:34:31’
- Дата и время: ‘2021-12-31 12:34:31’

Вставка, изменение, удаление данных

Вставка данных в таблицу

- Для вставки строк в таблицу используется оператор INSERT

- INSERT INTO имя_таблицы(имя_столбца1, имя_столбца2)
  VALUES (значение_столбца1, значение_столбца2);

- Например, есть таблица products с полями id, name и price
- INSERT INTO products(name, price)
  VALUES (‘Intel Core i5’, 10000);

- При этом id сгенерируется сам, если он объявлен как AUTO_INCREMENT

Вставка данных в таблицу

- Можно вставлять несколько строк за раз
- INSERT INTO products(name, price)
  VALUES (‘Intel Core i5’, 10000), (‘AMD Athlon’, 5000);

Удаление данных из таблицы

- Удалить все строки таблицы
- DELETE FROM products;

- Обычно удаляют не все строки, а те, которые удовлетворяют условию
- DELETE FROM products
  WHERE id = 3;
- DELETE FROM products
  WHERE price <= 100;

- В WHERE может идти любое логическое условие, рассмотрим их позднее

LIMIT при удалении строк

- DELETE – довольно опасная операция, потому что удаляются данные
- Чаще всего удаляют по id и чтобы перестраховаться, указывают, что должна быть удалена только 1 строка

- ```
  DELETE FROM products
  WHERE id = 3
  LIMIT 1;
  ```

- Вместо 1 может быть любое нужное число, но это частый пример
  ```

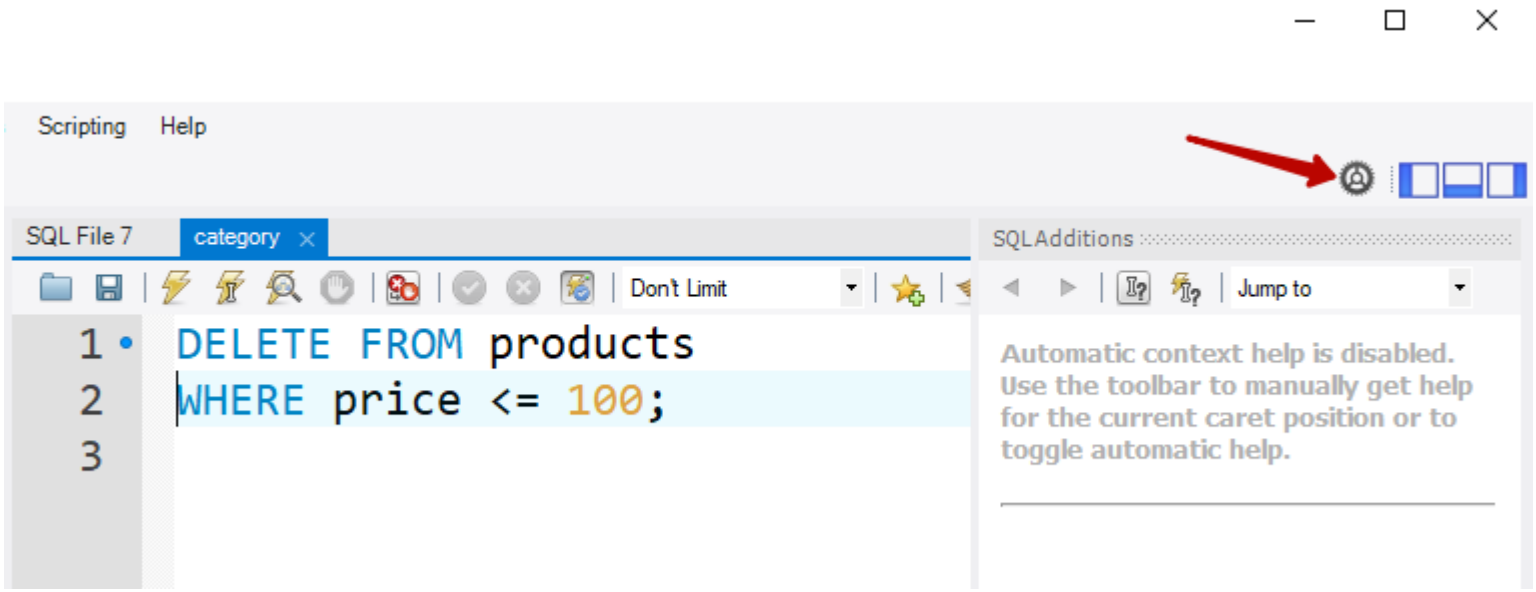
Удаление в Workbench

- Если попробовать выполнить подобный DELETE, то **Workbench** выдаст ошибку:
- DELETE FROM products
  WHERE price <= 100;
- Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column To disable safe mode, toggle the option in Preferences -> SQL Editor and reconnect.

- По умолчанию **Workbench** считает, что такой запрос небезопасный, т.к. вы можете удалить что-то лишнее
- Безопасными **Workbench** считает удаления по **id**

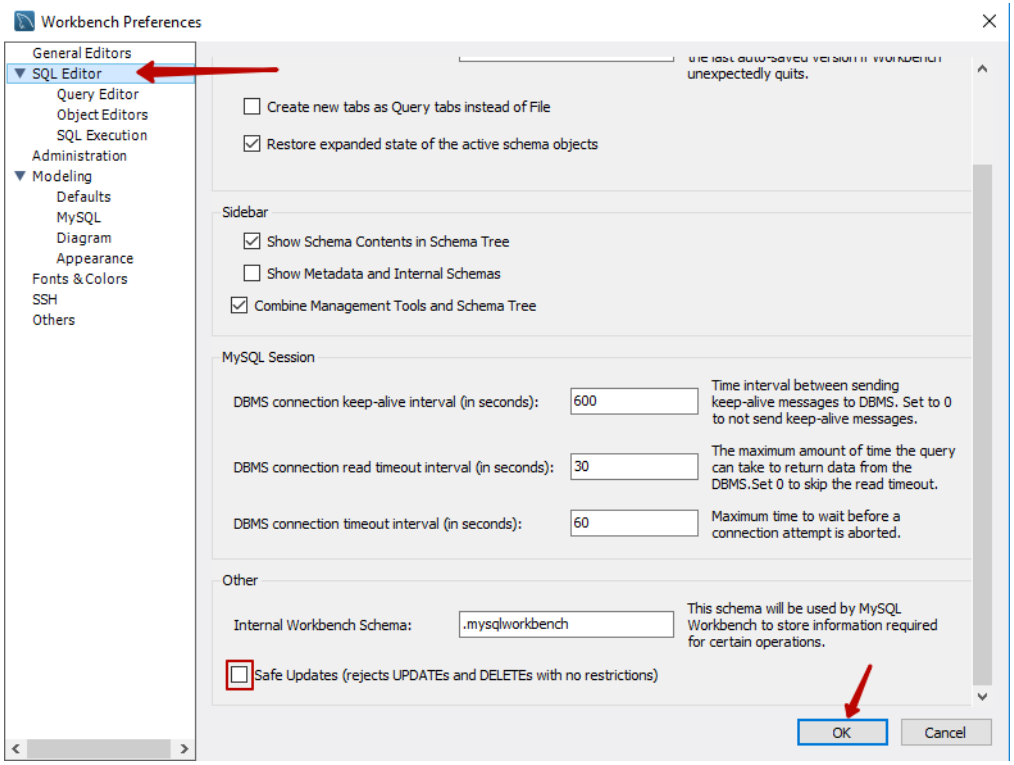
Удаление в Workbench

- Можно отключить эту проверку в настройках:



Удаление в Workbench

- Выберите вкладку **SQL Editor**, потом пролистайте диалог:



- Снимите галочку **Safe Updates**, нажмите **OK**
- После этого нужно выключить и включить **Workbench** заново

Обновление данных строк

- Чтобы изменить данные в строках, используется оператор UPDATE

- UPDATE products
  SET price = price * 2, name = ‘Inter Core i5 3.4 Ghz’
  WHERE id = 4;

- Часть WHERE не обязательна, тогда обновятся все строки
- В правой части от = могут быть выражения, использующие константы, поля, вызовы функций и т.д.

Арифметические операторы

Операция	Оператор
+	Сложение
-	Вычитание
*	Умножение
/	Деление

- В зависимости от СУБД логика работы деления может отличаться
- Например, в MySQL деление всегда вещественное. Например, 5 / 2 дает 2.5
- А в MS SQL такое деление будет целочисленным, т.к. оба числа целые: 5 / 2 дает 2
  - А если хотя бы одно число вещественное, то деление вещественное: 5.0 / 2 дает 2.5

Задача «Shop»

- Создать базу данных магазина из 2 таблиц - продукты (товары) и категории товаров
- Каждый товар принадлежит некоторой категории
- Вставьте данные в таблицы (3-5 строк)

- Попробовать UPDATE, DELETE:
  - Удалить товары с ценой выше 100
  - Изменить название и цену некоторому товару

Требования по сдаче заданий по SQL

- Задачи нужно присылать в виде текстовых файлов с SQL запросами/командами
  - Т.е. не Word файлы и не скриншоты
- Файлам делать расширение .sql
- Дополнительно можно указывать некоторый поясняющий текст (например, номера запросов или условие задачи)
  - Если это есть, то лучше оформлять в виде комментариев
- Файлы должны хорошо открываться в обычном блокноте и Notepad++ и/или IDEA (т.е. не должно ехать форматирование)

Хороший стиль форматирования SQL

- Ключевые слова нужно писать заглавными буквами
- Между отдельными командами ставьте пустую строку для читаемости
- Вокруг операторов вроде = * и т.д. нужно ставить по пробелу для читаемости
- После запятой в перечислении для читаемости нужно ставить пробел
- В операторах UPDATE, DELETE нужно писать SET, WHERE, LIMIT с новой строки
- В INSERT нужно писать VALUES с новой строки
- Слишком длинные строки нужно переносить

Форматирование CREATE TABLE

- CREATE TABLE нужно форматировать так:

- ```
  CREATE TABLE products
  (
     id INT AUTO_INCREMENT,
     name VARCHAR(255) NOT NULL,
     PRIMARY KEY (id)
  );
  ```
- Строки внутри круглых скобок сдвинуты вправо на 1 табуляцию
  ```

Хороший стиль имен

- Для имен должна использоваться единая нотация – либо везде нотация с _, либо везде верблюжья нотация
- Для внешних ключей лучше давать имя вида **названиеДругойТаблицыId**
  - Например: **categoryId** или **category_id**