

# DB Introduction, PostgreSQL

# НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

# ЦЕЛЬ

Настроить среду для работы с PostgreSQL. Приступить к изучению основ синтаксиса SQL.

# ПЛАН ЗАНЯТИЯ

- Установка сервера и Beekeeper Studio
- БД и СУБД
- Таблица, строки, столбцы, primary key, нормализация в общих словах
- SQL запросы - практика

# Установка сервера PostgreSQL, Beekeeper Studio для **Windows**

(инструкция для мак на следующем слайде)

Инструкция по установке:

<https://winitpro.ru/index.php/2019/10/25/ustanovka-nastrojka-postgresql-v-windows/>

Ссылка на установщик: <https://www.postgresql.org/download/>

Имя юзера: postgres

Выберите пароль для суперюзера: qwerty007

P.S. на реальных проектах не используйте такие пароли!

Beekeeper:

<https://github.com/beekeeper-studio/beekeeper-studio/releases/tag/v4.0.3>

Перейдите по ссылке и скачайте  
**Beekeeper-Studio-Setup-4.0.3.exe**

(можете выбрать другой из списка, если он подходит лучше для вашей операционной системы)

# Установка сервера PostgreSQL, Beekeeper Studio для **Mac OS**

Установите два приложения по ссылкам ниже:

- <https://postgresapp.com/>
- <https://github.com/beekeeper-studio/beekeeper-studio/releases/tag/v4.0.3> переходите по ссылке и скачиваете и устанавливаете Beekeeper-Studio-Setup-4.0.3.dmg (можете выбрать другой из списка, если он подходит лучше)

# Основы реляционных БД

**Система управления базами данных, СУБД** — специальная программа-сервер позволяющая использовать и управлять базами данных. СУБД позволяет читать и записывать данные, искать по ним и выполнять сложные выборки

# Основы реляционных БД

## База данных

**БД** — фактически создаваемые на диске файлы, в которых хранится информация записанная с помощью СУБД

**Важно!** Часто значение БД и СУБД путают, можно услышать фразу “БД PostgreSQL” когда имелось ввиду “СУБД PostgreSQL”



# Основы реляционных БД

## Что такое РСУБД?

**Реляционная система управления базами данных, РСУБД / Relational Database Management System (RDBMS)** — СУБД которая хранит данные в виде таблиц и строк в этих таблицах. Любая данные в РСУБД должны быть структурированы в соответствии с реляционной моделью. Примеры РСУБД:

- MySQL
- PostgreSQL
- MSSQL
- Oracle

# Основы реляционных БД

SQL - язык структурированных запросов (Structured Query Language)

**SQL** — декларативный язык программирования, предназначенный для описания, изменения и извлечения данных из СУБД. Чаще всего используется в реляционных СУБД.

# Data Definition Language, Data Control Language

**DDL** — подмножество языка SQL, предназначенное для создания, изменения и удаления баз данных и таблиц в них. К нему относятся все команды, начинающиеся с CREATE, ALTER и DROP.

**DCL** — подмножество языка SQL, предназначенное для управления правами доступа к базам данным и таблицам в них. С помощью него можно разрешить и запретить пользователям создавать, изменять и удалять базы данных или таблицы. К нему относятся все команды, начинающиеся на GRANT и REVOKE.

# SQL: создание пользователя СУБД

Пользователь создаётся именно в СУБД!

```
CREATE USER test_user WITH PASSWORD 'qwerty';
```

# SQL: создание базы данных

Создание базы данных с названием  
**test\_db**

Любая база должна быть создана,  
прежде чем к ней можно будет  
подключиться.

Пользователь, указанный как owner,  
будет иметь максимальные права  
доступа к этой базе данных.

```
CREATE DATABASE test_db OWNER test_user;
```

# SQL: права пользователя в БД

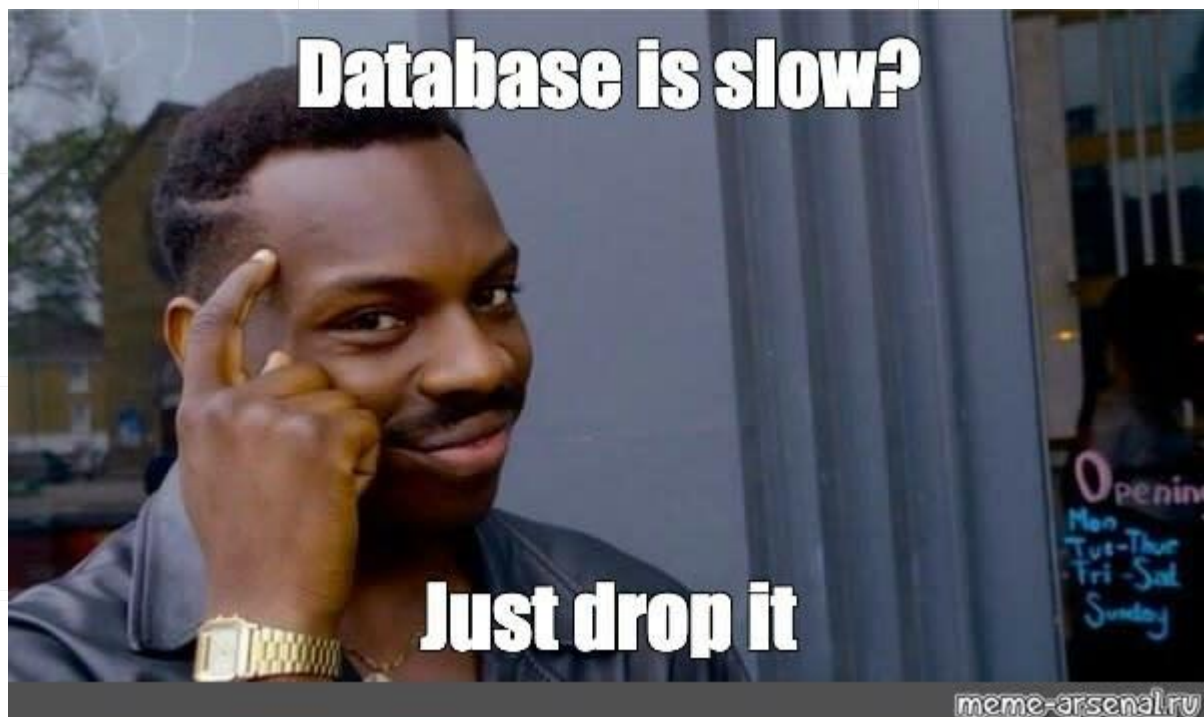
По умолчанию доступ к базе имеет только owner и супер-администратор СУБД. Всем остальным пользователям доступ нужно добавлять явным образом.

```
GRANT ALL PRIVILEGES ON DATABASE test_db  
TO test_user;
```

# SQL: удаление базы данных

Удалить базу данных может её владелец, супер-администратор СУБД или пользователь, которому явным образом даны на это права.

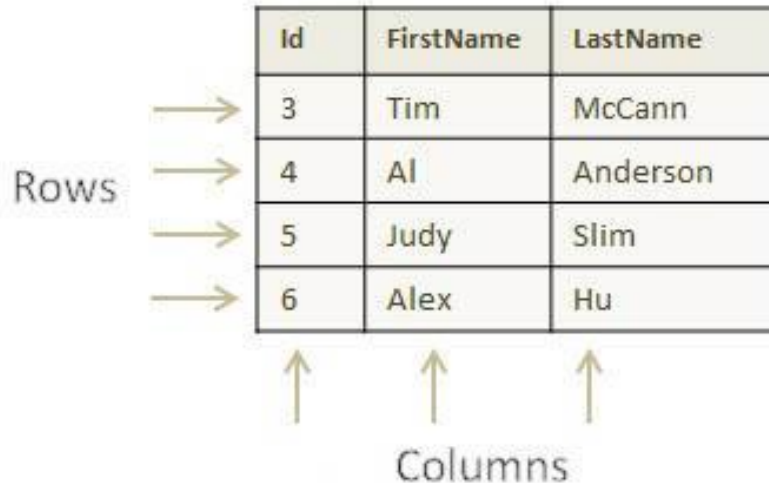
```
DROP DATABASE test_db;
```





# Таблица, строка, столбец

- Таблица (table) — совокупность строк и столбцов
- Строка (row) — запись в БД о конкретной сущности
- Столбец (column) — любой атрибут сущности хранящейся в строке



The diagram illustrates a database table with four rows and three columns. The columns are labeled 'Id', 'FirstName', and 'LastName'. The rows contain data for four individuals: Tim McCann, Al Anderson, Judy Slim, and Alex Hu. Arrows labeled 'Rows' point to each row, and arrows labeled 'Columns' point to each column header.

Id	FirstName	LastName
3	Tim	McCann
4	Al	Anderson
5	Judy	Slim
6	Alex	Hu

# База данных

SQL

Retrieve data  
Update data  
Remove data

Database

User



Product

# Типы данных: числовые

- **serial** — целое с автоувеличением, от 1 до 2147483647
- **smallint, int2** — целое, от -32768 до +32767;
- **integer, int, int4** — целое, от -2147483648 до +2147483647
- **bigint, int8** — целое, от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
- **double precision, float8** — с плавающей точкой, от 1E-307 до 1E+308

# Типы данных: строковые

- `character(n)`, `char(n)` — представляет строку из фиксированного количества символов. С помощью параметра задается количество символов в строке
- `character varying(n)`, `varchar(n)` - представляет строку переменной длины с ограничением. С помощью параметра задается ограничение символов в строке
- `text` — представляет текст произвольной длины

# Типы данных: время и даты

- **timestamp** — хранит дату и время, с учётом часового пояса или без. Для дат самое нижнее значение 4713 год до н. э., самое верхнее значение 294276 год н. э.
- **date** — представляет дату от 4713 год до н. э. до 5874897 года н.э
- **time** — хранит время суток, без даты, с учётом часового пояса или без. Принимает значения от 00:00:00 до 24:00:00

# Типы данных: прочие

- **boolean** — булево значение, true или false
- **jsonb** — данные произвольного формата в формате JSON

# SQL: создание таблицы

Создание таблицы с полями

**students** с полями id, name, age

```
CREATE TABLE students (  
  id serial PRIMARY KEY, name varchar(80), age  
  integer  
);
```

## PRIMARY KEY ограничение

Ограничение первичного ключа указывает, что столбец или группа столбцов могут использоваться в качестве уникального идентификатора строк в таблице. Для этого необходимо, чтобы значения были уникальными и не были нулевыми. Итак, следующие два определения таблицы принимают одни и те же данные

```
CREATE TABLE products (  
    product_id serial PRIMARY KEY,  
    name text,  
);
```



# SQL: удаление таблицы

Удаление таблицы с названием  
**students**

```
DROP TABLE students;
```

# Data Manipulation Language

**DML** — подмножество языка SQL, предназначенное для создания, изменения и удаления данных в таблицах внутри базы данных. К нему относятся все команды, начинающиеся с SELECT, INSERT, UPDATE и DELETE. (CRUD аббревиатура CREATE READ UPDATE DELETE)

В целом, SQL DML очень похожи среди разных баз данных, но иногда бывают серьезные отличия, поэтому всегда полезно сверяться с документацией по конкретной БД в случае проблем.

# SQL: добавление строк в таблицу

Обратите внимание, что поле id существует в таблице, но указывать его значение не обязательно. Тип данных serial позволяет ему заполняться автоматически.

```
INSERT INTO students (name, age) VALUES  
( 'Anna' , 25) ,  
( 'Maria' , 23) ,  
( 'Roman' , 28) ;
```

# SQL: извлечение данных

С помощью `SELECT` можно не просто вытаскивать данные, но и проводить фильтрацию, сортировать и проводить несложные агрегации.

```
SELECT * FROM students;
```

```
SELECT name, age FROM students;
```

```
SELECT * FROM students WHERE age < 18;
```

```
SELECT * FROM students ORDER BY age ASC;
```

# SQL: оператор WHERE

Оператор `WHERE` используется для фильтрации записей на основе заданных условий. Он используется в операторе `SELECT`, а также в операторах `UPDATE`, `DELETE` и `INSERT`

```
SELECT * FROM students WHERE age > 25;
```

# SQL: оператор ORDER BY

Оператор `ORDER BY` используется для сортировки записей по одному или нескольким столбцам. По умолчанию сортировка выполняется в порядке возрастания (ASC), но можно явно указать порядок убывания (DESC).

```
SELECT * FROM products ORDER BY price ORDER DESC;
```

# SQL: изменение данных

С помощью `UPDATE` можно изменять любые данные в таблице.

Использование `WHERE` не обязательно, хотя и крайне рекомендуется. Без `WHERE` будут обновлены все существующие строки в таблице.

```
UPDATE students SET age = 26 WHERE name = 'Anna';
```

# SQL: удаление данных

Чаще всего удаление происходит по id, но можно удалять и по любому другому полю:

Возможно использование DELETE вообще без WHERE, в таком случае будут удалены вообще все данные в таблице.

```
DELETE FROM students WHERE id = 1;
```

```
DELETE FROM students WHERE name = 'Anna';
```



# Игра SQL Island для изучения SQL

Ссылка на игру на немецком:

[ссылка](#)

Ссылка на игру на английском:

[ссылка](#)



# **Ваша новая IT-профессия – Ваш новый уровень жизни**

Программирование с нуля в  
немецкой школе AIT TR GmbH