

# 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)

# Повторим

В чем разница БД и СУБД?

Что такое таблица?

Что такое строка?

Что такое столбец?

Что такое shema?

Что значит аббревиатура CRUD?

# ЦЕЛЬ

**Изучить, какие существуют типы ключей, способы связи между таблицами и научиться проектировать db в графическом редакторе.**

# ПЛАН ЗАНЯТИЯ

■ Типы ключей, связи между таблицами

■ Проектирование БД при помощи <https://drawsql.app/>

■ JOIN-операции

# Проектирование базы данных



# Зарегистрируйтесь на drawsql

<https://drawsql.app/>


(можно воспользоваться гугл почтой)


# Database diagrams



31.1 31.2


 My diagrams

 Shared with me

 Experiment

 Create new folder

 My favorites

 Browse templates 

## New database diagram



### Diagram settings

Configure your new database diagram. The database type cannot be changed once the diagram has been created.

Team

 31.1 31.2

Diagram name

Diagram visibility 

☒ Public

☐ Private

 Pro

Database type

MySQL



PostgreSQL



SQL Server



Create diagram



# Первичный ключ (Primary key)

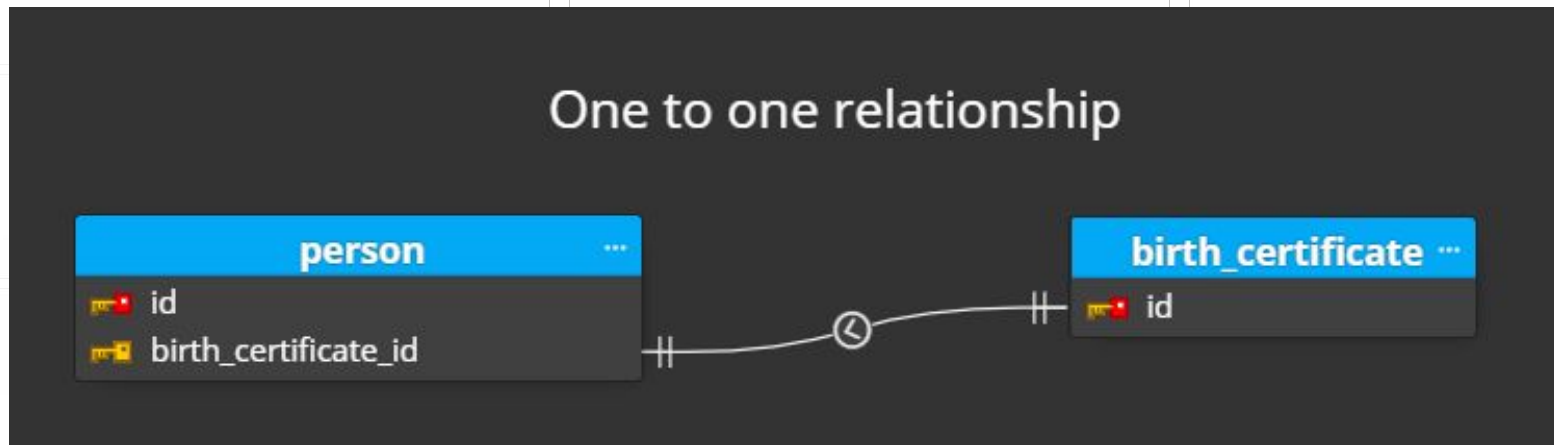
В PostgreSQL первичный ключ - это отдельное поле или комбинация полей, которые однозначно определяют запись. Ни одно из полей, являющихся частью первичного ключа, не может содержать значение NULL. Таблица может иметь только один первичный ключ.

# Внешний ключ (Foreign key)

Для связывания таблиц в PostgreSQL используются внешние ключи. Внешний ключ в таблице связывает ее с другой таблицей, указывая идентификатор связанной записи.

# Способы связывания таблиц: Один-к-одному (One-to-One)

Каждая запись в одной таблице связана с одной записью в другой таблице.



# Способы связывания таблиц: Один-к-одному (One-to-One)

Какие примеры связи one-to-one вы знаете в реальной жизни?

# Один-ко-многим (One-to-Many)

Каждая запись в одной таблице связана с несколькими записями в другой таблице.

## One to many relationship



# Один-ко-многим (One-to-Many)

Какие примеры связи one-to-many вы знаете в реальной жизни?

# Многие-ко-многим (Many-to-Many)

Каждая запись в одной таблице может быть связана с несколькими записями в другой таблице, и наоборот.

Many to many relationship



# Многие-ко-многим (One-to-Many)

Какие примеры связи many-to-many вы знаете в реальной жизни?



# Один-к-одному (One-to-One)

Между клиентом и контактной информацией связь 1:1

```
CREATE TABLE employees (  
    id integer PRIMARY KEY,  
    name varchar(100)  
);  
  
CREATE TABLE contact_info (  
    employee_id integer REFERENCES employees(id) UNIQUE,  
    email text,  
    phone_number varchar(9)  
);
```

# Один-ко-многим (One-to-Many)

У одного клиента может быть много заказов.

```
CREATE TABLE orders (  
    order_number integer,  
    customer_id integer REFERENCES customers(id),  
);
```

```
CREATE TABLE customers (  
    id integer PRIMARY KEY,  
    name text  
);
```

# Многие-ко-многим (Many-to-Many)

Одну песню могут написать несколько артистов. У артиста может быть много песен, которые он написал.

```
CREATE TABLE songs (  
  id integer PRIMARY KEY,  
  name varchar(100)  
);  
  
CREATE TABLE artists (  
  id integer PRIMARY KEY,  
  name varchar(100)  
);  
  
CREATE TABLE songs_artists (  
  artist_id integer REFERENCES artists(id),  
  song_id integer REFERENCES songs(id),  
  PRIMARY KEY (artist_id, song_id)  
);
```

# Join

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

```
SELECT
```

```
*
```

```
FROM
```

```
product
```

```
JOIN category ON category.id = product.category_id;
```

# Преимущества связывания таблиц:

- **Исключение повторений данных:** связывание позволяет избежать дублирования информации в разных таблицах. Например, если в базе данных есть таблица «Пользователи» и таблица «Заказы», то связывание позволит связать каждый заказ с соответствующим пользователем, не дублируя информацию о пользователе в каждой записи заказа.
- **Улучшение эффективности запросов:** при связывании таблиц можно выполнять запросы, отбирая данные из нескольких таблиц одним запросом. Это уменьшает количество запросов к базе данных и повышает производительность системы.
- **Обеспечение целостности данных:** связывание таблиц позволяет определить связи между разными сущностями в базе данных и задать ограничения на значения ключевых полей. Например, при связывании таблиц «Пользователи» и «Заказы» можно задать ограничение, что каждый заказ должен быть связан с существующим пользователем.
- Связывание таблиц является одним из важнейших инструментов реляционных баз данных, таких как PostgreSQL. Оно **помогает организовать структуру** данных, улучшить производительность системы и обеспечить целостность информации.



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

Программирование с нуля в  
немецкой школе AIT TR GmbH