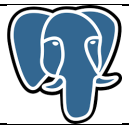
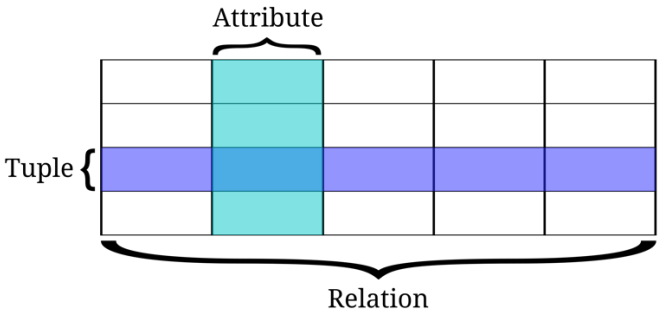
Data Engineering 0: основы баз данных и SQL - Семинар 1

Модели данных

Существуют разные модели данных, каждая из них помогает удобнее работать со своими задачами

Модель данных	**Сценарии использования**	**Примеры**	**Логотип**
Реляционная	Составные запросы и транзакции	PostgreSQL	
Документная	Данные с большой вложенностью	MongoDB	
Колоночная	Аналитические запросы	Clickhouse	
Ключ-значение	Кэширование	Redis	
Графовая	Анализ связей между сущностями	Neo4j	

Реляционная модель



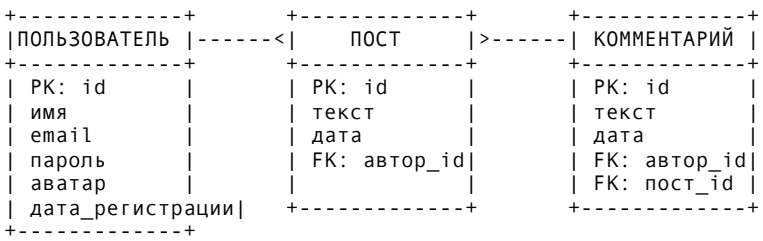
Отношение (Relation) – таблица
Кортеж (Tuple, Строка) – запись в таблице, представляющая конкретный объект
Столбец (Attribute, Поле) – отдельная характеристика объекта
Первичный ключ (Primary Key, PK) – уникальный идентификатор записи в таблице
Внешний ключ (Foreign Key, FK) – ссылка на PK из другой таблицы (для установления связи между записями в разных таблицах)

Индекс (Index) – дополнительная структура для ускорения поиска по таблице
Ограничения (Constraints) – правила для обеспечения согласованности данных. Примеры: UNIQUE, NOT NULL

Логическая модель БД

Отвечает на вопрос: «Что храним?»
- Какие таблицы будут?
- Какие поля в таблицах будут?
- Какие связи между таблицами будут?

Часто рисуется в виде ER-диаграммы. Пример:



Физическая модель БД

Отвечает на вопрос: «Как храним?»
- Конкретные типы данных
- Как оптимизировать типовые запросы (индексы)
- Конкретные описания таблиц, связей и ограничений

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    avatar VARCHAR(255),
    registration_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE posts (
    id SERIAL PRIMARY KEY,
    text TEXT NOT NULL,
    creation_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    author_id INTEGER NOT NULL,
    CONSTRAINT fk_posts_author
        FOREIGN KEY (author_id)
        REFERENCES users(id)
        ON DELETE CASCADE
);

CREATE TABLE comments (
    id SERIAL PRIMARY KEY,
    text TEXT NOT NULL,
    creation_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    author_id INTEGER NOT NULL,
    post_id INTEGER NOT NULL,
    CONSTRAINT fk_comments_author
        FOREIGN KEY (author_id)
        REFERENCES users(id)
        ON DELETE CASCADE,
    CONSTRAINT fk_comments_post
        FOREIGN KEY (post_id)
        REFERENCES posts(id)
        ON DELETE CASCADE
);
```

```
CREATE INDEX idx_posts_author ON posts(author_id);
CREATE INDEX idx_comments_author ON comments(author_id);
CREATE INDEX idx_comments_post ON comments(post_id);
```

PlantUML

Инструмент, позволяющий рисовать диаграммы из текстового описания.
Связи между сущностями обозначаются так:

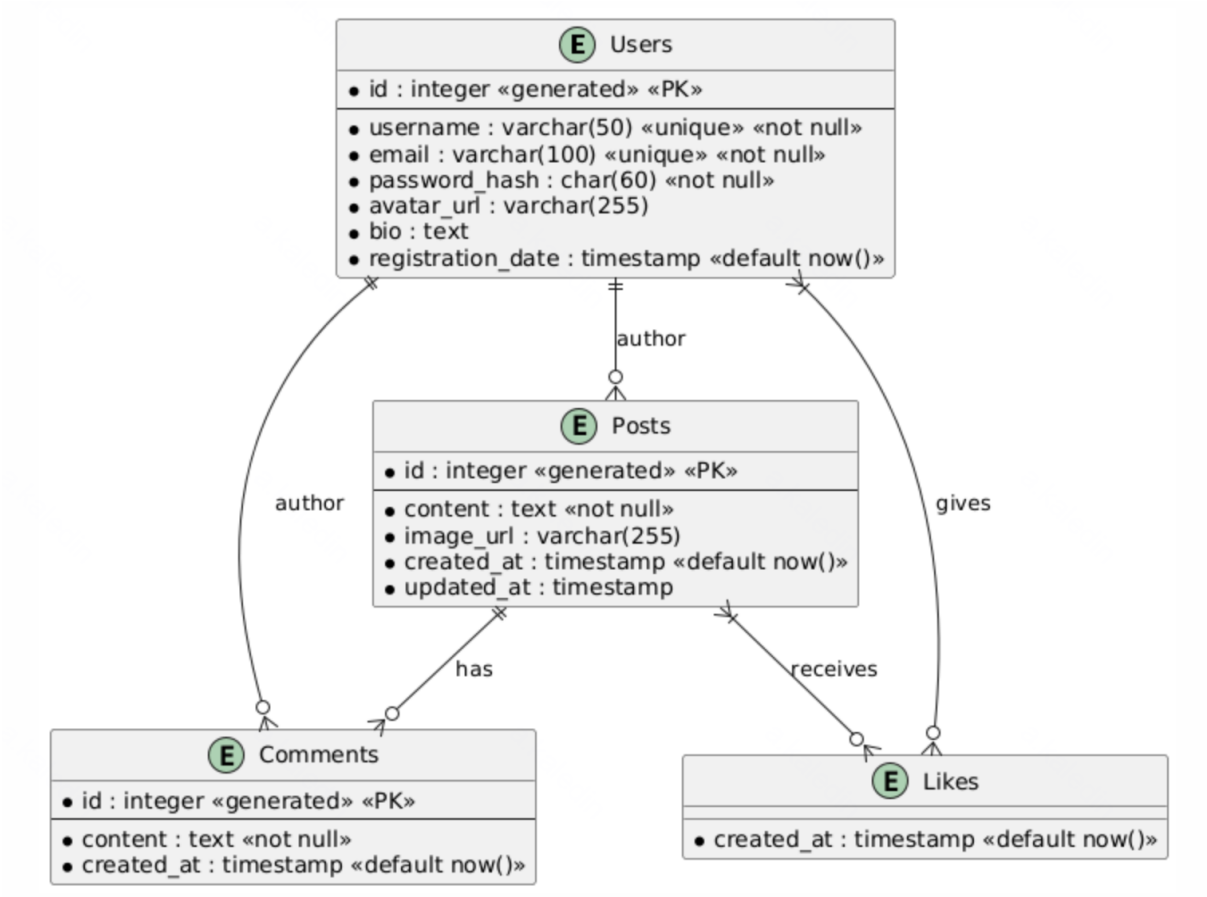
Type	Symbol
Zero or One	|o--
Exactly One	||--
Zero or Many	}o--
One or Many	}|--

Определим для социальной сети такие сущности:

- Users
- Posts
- Comments
- Likes

```
@startuml SocialNetworkERD

entity Users {
  * id : integer <<generated>> <<PK>>
  --
  * username : varchar(50) <<unique>> <<not null>>
  * email : varchar(100) <<unique>> <<not null>>
  * password_hash : char(60) <<not null>>
  * avatar_url : varchar(255)
  * bio : text
  * registration_date : timestamp <<default now()>>
}

entity Posts {
  * id : integer <<generated>> <<PK>>
  --
  * content : text <<not null>>
  * image_url : varchar(255)
  * created_at : timestamp <<default now()>>
  * updated_at : timestamp
}

entity Comments {
  * id : integer <<generated>> <<PK>>
  --
  * content : text <<not null>>
  * created_at : timestamp <<default now()>>
}

entity Likes {
  * created_at : timestamp <<default now()>>
}

Users ||--o{ Posts : "author"
```

```
Users ||--o{ Comments : "author"
Posts ||--o{ Comments : "has"
Users }|--o{ Likes : "gives"
Posts }|--o{ Likes : "receives"


@enduml
```



ДЗ

Спроектировать схему для интернет-магазина. Сдать в формате PlantUML + схема (как в примере выше схема генерируется автоматически на основе PlantUML описания). Задание творческое, какие атрибуты есть у каждой из сущностей и как должны быть сущности связаны между собой определяете самостоятельно. Должны присутствовать следующие сущности:

- Товар
- Пользователь
- Категория товара
- Заказ
- Корзина
- Избранные товары у пользователя

Что еще сделать дома?

На следующем занятии будем использовать DataGrip / DBeaver. Установите его заранее на компьютеры, чтобы могли повторять действия у себя на компьютерах.

Еще будем использовать Docker + Docker Compose, тоже установите и проверьте их работу.