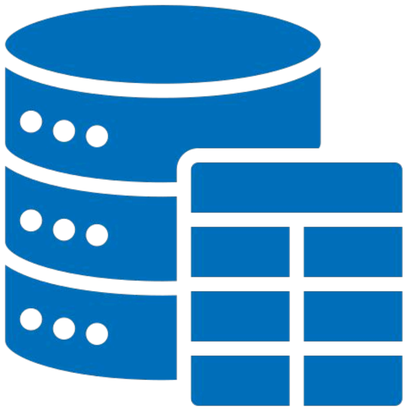
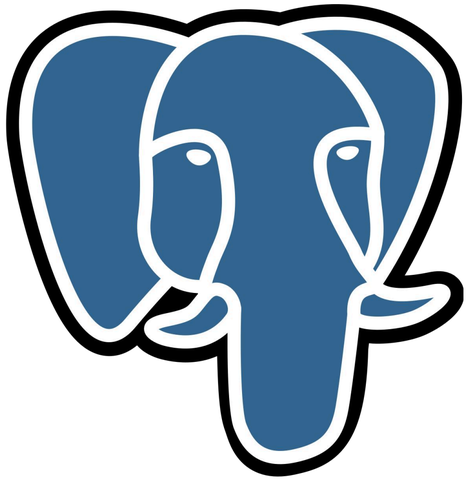
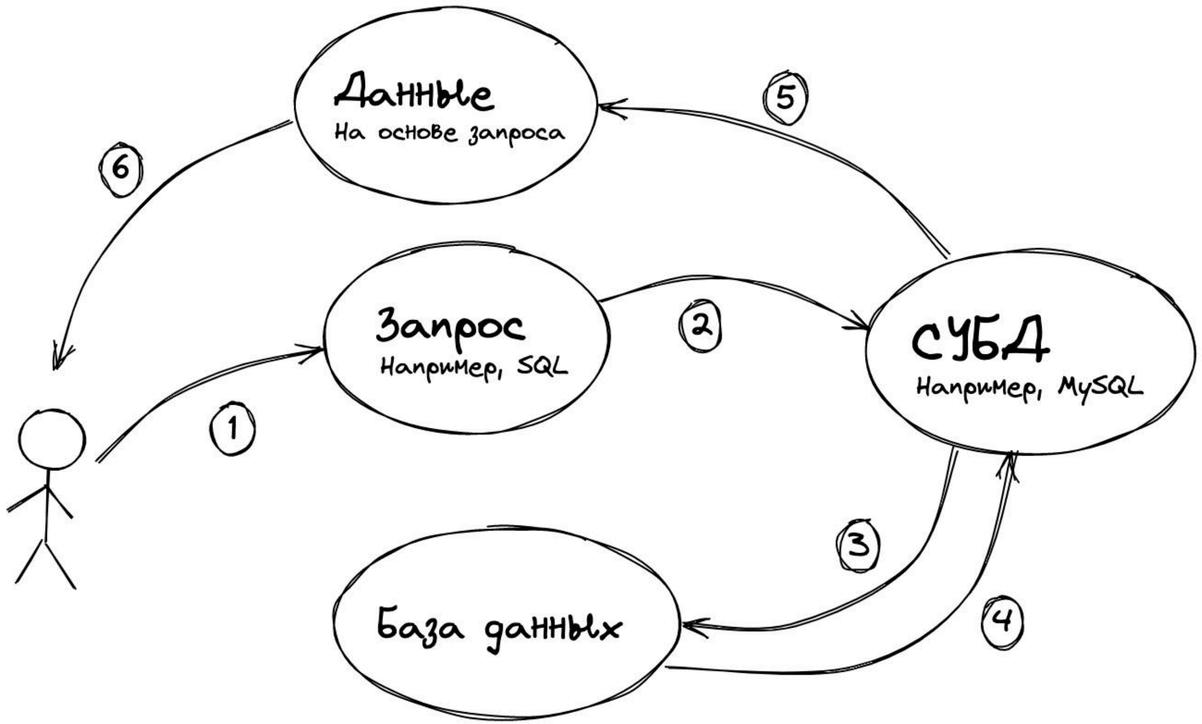




Введение в базы данных

“ИТ-класс”. Бэкенд-разработка на Python.
Бусыгин Дмитрий, 2025 год.

Что это и с чем его едят?



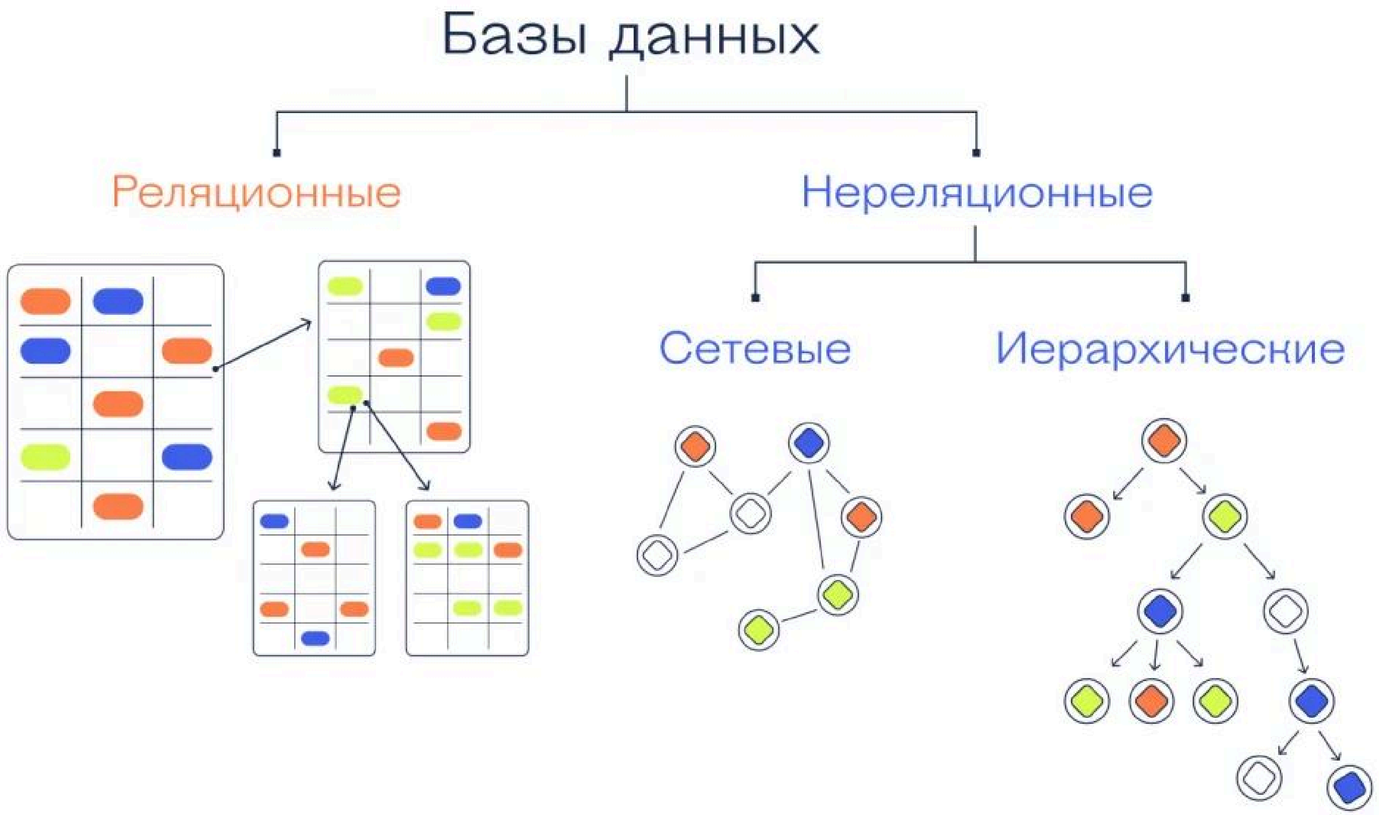
Как всё устроено

1. Клиент отправляет запрос на сервер
2. Сервер обращается к СУБД
3. СУБД обрабатывает информацию из своей БД.
4. Когда результат получен, ответ идёт в обратном направлении к клиенту

База данных (БД) — это организованное хранилище данных, которое позволяет их систематизировать, управлять ими, а также хранить, извлекать и обрабатывать информацию.

Система управления базами данных (СУБД) — это программное обеспечение, которое позволяет работать с этим хранилищем

Виды баз данных



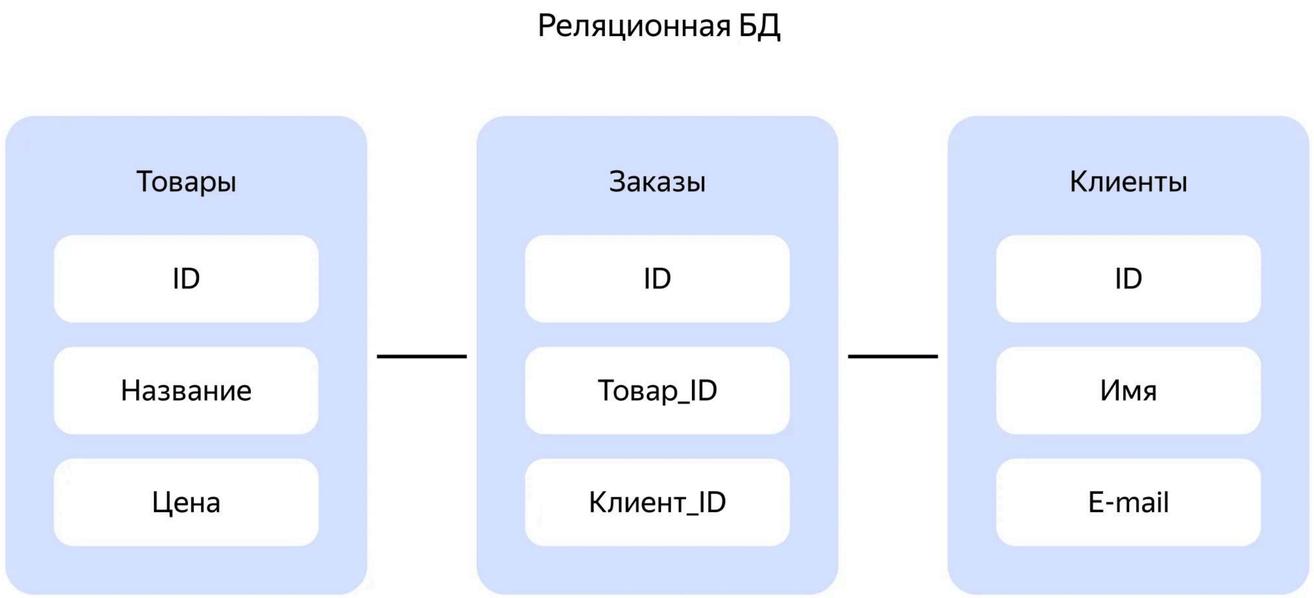
Реляционные — используют строгие таблицы со связями

Нереляционные — имеют более сложную структуру

- Дерево
- Граф
- Документы
- Столбцы (а не строки)

Но я почти уверен, что нереляционные БД вам не пригодятся в проектах :)

Таблицы — главное в БД



Основные типы данных

- INTEGER
- FLOAT
- SERIAL
- VARCHAR(length), TEXT
- DATE, TIMESTAMP, DATETIME
- BOOLEAN
- BYTES

Любая реляционная база данных состоит из таблиц,
каждая из которых состоит из колонок (столбцов), в которых
хранится конкретный тип данных.
Аналогия: БД это .xlsx файл, а таблица — это книга в ней

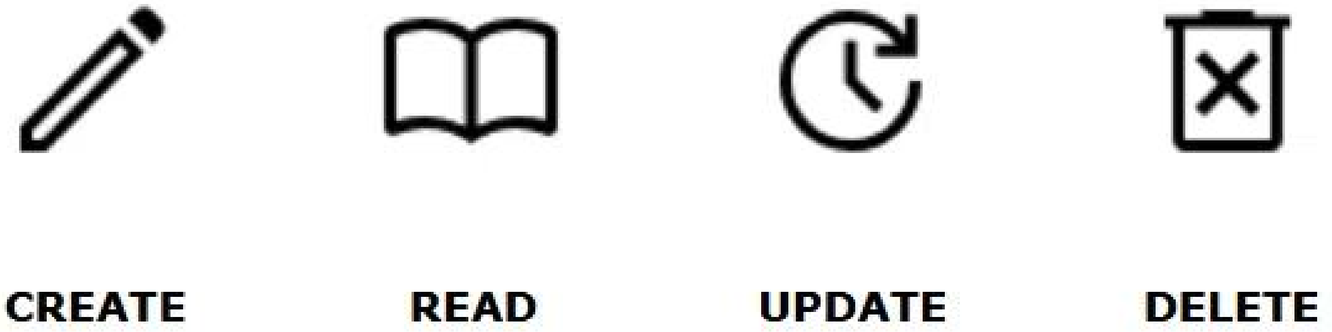
Язык запросов к БД


```
CREATE TABLE students (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE,
    grade INTEGER,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT NOW()
);
```

SQL (Structured Query Language) — это язык структурированных запросов, используемый для управления и взаимодействия с реляционными БД. С его помощью можно извлекать, добавлять, изменять и удалять данные, а также создавать и изменять структуру БД.

SQL является стандартным языком для большинства СУБД

Create, Read, Update, Delete



Эти четыре основные операции используются для управления данными в постоянных хранилищах, таких как базы данных.

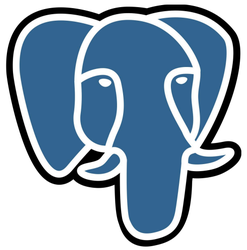
Применение

- Базы данных: CRUD является основой для работы с большинством реляционных и NoSQL-баз данных.
- Веб-API: Позволяет приложениям взаимодействовать с данными через веб-интерфейс, где каждая операция соответствует определенному HTTP-методу.
- Пользовательский интерфейс: Может использоваться для описания соглашений интерфейса, которые упрощают работу с данными с помощью форм и отчетов.

Виды СУБД

SQLite — для обучения и простых проектов
Плюсы: нулевая настройка, один файл
Минусы: нет настоящих подключений, слабая производительность
Дурной тон, но если не хотите заморачиваться с настройкой БД, то на первое время сойдёт

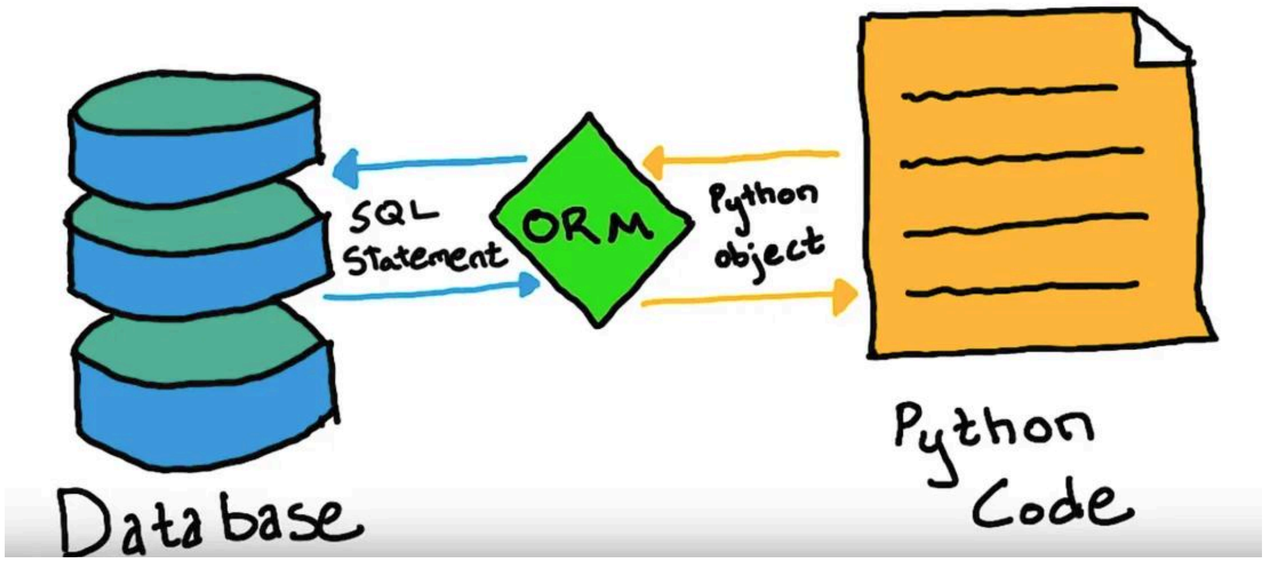
MySQL — классический выбор
Плюсы: много хостингов, простота
Минусы: менее строгий к данным
Для простых учебных проектов сойдёт, но при масштабировании придется переходить на...



НАШ СЛОНЯРА

PostgreSQL — наш выбор для проектов
Плюсы: строгий к типам, богатые возможности
Минусы: требует больше ресурсов
Если у вас комплексная бизнес-логика и вам важна надежность, то этот вариант лучший

А как с этим работать на Python?



Ручной запрос

```
cursor.execute("\
SELECT * \
FROM users \
WHERE age > 18\
")
```

Запрос через ORM (SQLAlchemy)

```
users = session
.query(User)
.filter(User.age > 18)
.all()
```

ORM (Object-Relational Mapping) — технология в программировании. Позволяет разработчикам работать с базами данных с помощью объектно-ориентированных языков.

Плюсы ORM: ниже вероятность ошибки из-за человеческого фактора. Надёжность.

Драйвера для подключения к БД

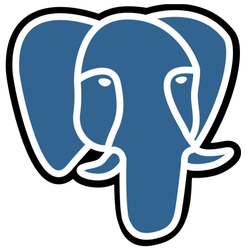
psycopg2 (синхронный)
- Блокирующие операции
- Проще в понимании
- Подходит для скриптов

asyncpg (асинхронный)
- Неблокирующие операции
- Выше производительность
- Совместим с FastAPI
- Современный стандарт

Для SQLAlchemy используем:
postgresql+asyncpg://user:pass@host/db

Стандартная связка

1. СУБД на PostgreSQL
2. Подчключение через asyncpg
3. ORM на SQLAlchemy

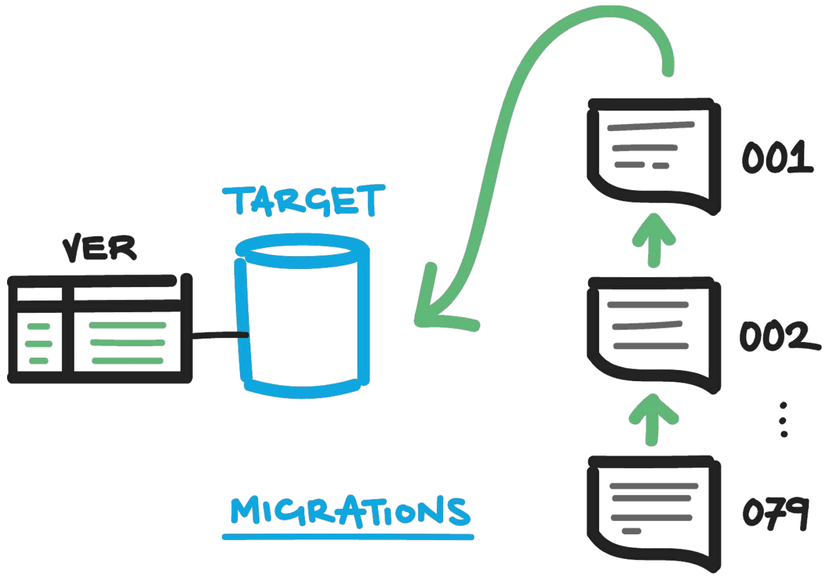


и asyncpg

Базы данных тоже обновляют и расширяют

Напрямую менять структуру таблиц некорректно:

- Можно лишиться ценных данных
- Будет тяжело вернуть всё как было, если будет надо
- Если каждый запускает БД локально, то сложно разобраться, как надо обновить таблицы и данные в них, чтобы всё работало правильно.

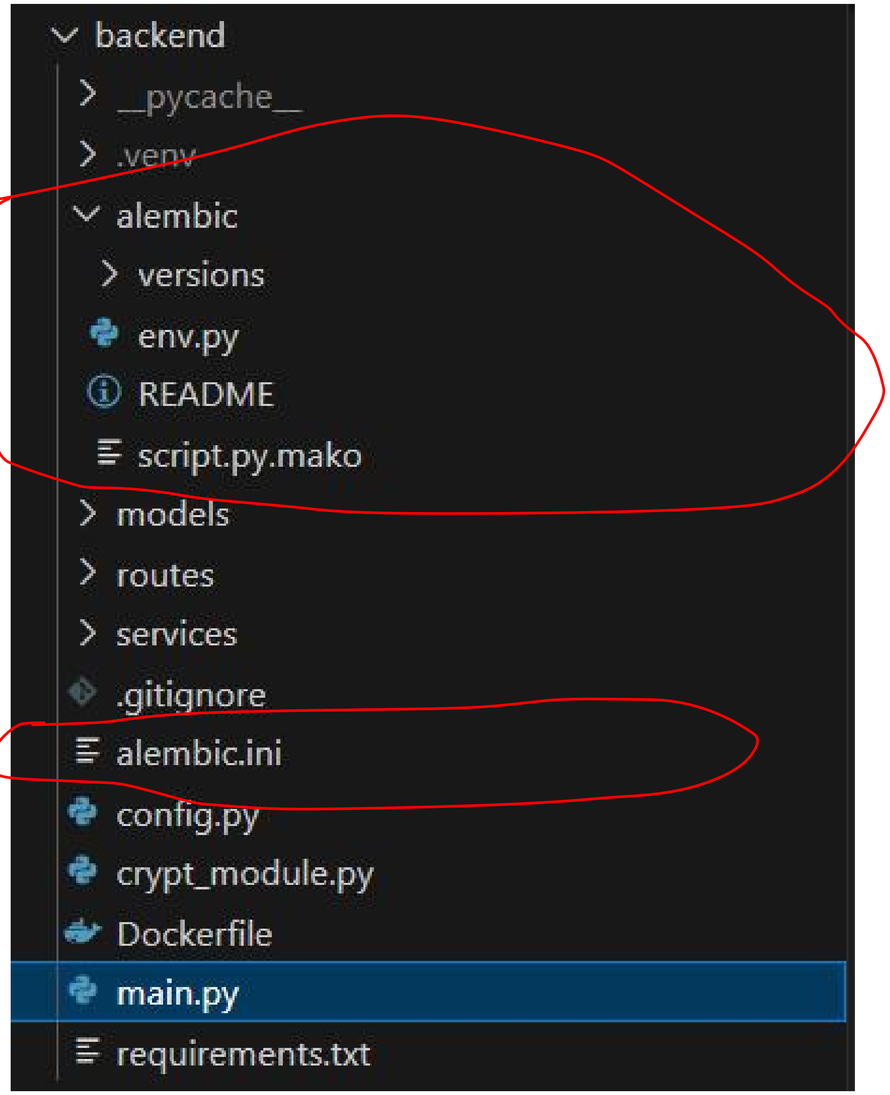


Для обновления схем и данных делают **миграции**:

- Своего рода репозиторий для базы данных
- Любое изменение прописывается кодом в файле
- Любую миграцию можно откатить (если написать rollback)

Data migrations — изменение данных в столбцах (UPDATE, INSERT, DELETE)
Schema migrations — изменение структуры таблиц (CREATE TABLE, ADD COLUMN, DROP INDEX)

Alembic



🏗️ Schema Migrations (Схемы):
• CREATE TABLE, ADD COLUMN, DROP INDEX
• Меняют СТРУКТУРУ БД
• Пример: добавить колонку "phone"

📊 Data Migrations (Данные):
• UPDATE, INSERT, DELETE
• Меняют СОДЕРЖИМОЕ БД
• Пример: заполнить "phone" на основе других полей

💡 Best Practices:
• Всегда пишите downgrade()
• Тестируйте на копии продакшн-данных
• Разделяйте schema и data миграции
• Для больших данных используйте batch-операции

Самостоятельное обучение

https://sql-academy.org/ru/trainer
— бесплатный тренажер по SQL. Подойдёт, чтобы закрепить материал и набить руку.

https://roadmap.sh/sql
— полный гайд на глубокое изучение SQL (CRUD — лишь верхушка айсберга)