МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

Урок №4

PostgreSQL

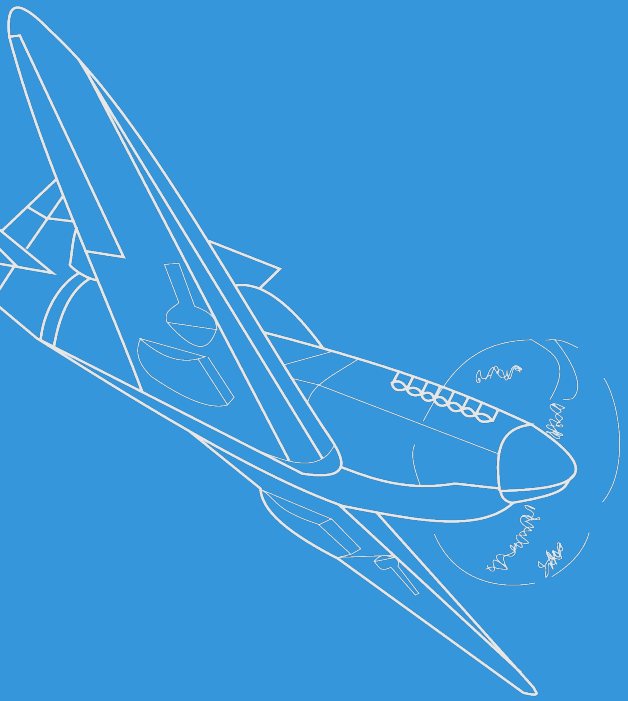
на котором расскажут про psql, синтаксис SQL, виды операций в SQL, как подключить в Django базу данных.

04 октября 2021 года

Антон Кухтичев

Содержание занятия

1. PostgreSQL;
2. Что такое SQL;
3. ORM в Django;



Реляционные базы данных

Решаемые проблемы

- Структура хранения;
- Эффективный поиск данных,
- Управление памятью,
- Совместный доступ к данным,
- Атомарные операции-транзакции,
- Язык управления базой и данным - SQL.

Реляционная модель данных

Данные хранятся в виде таблиц. У каждой таблицы фиксированное число столбцов.

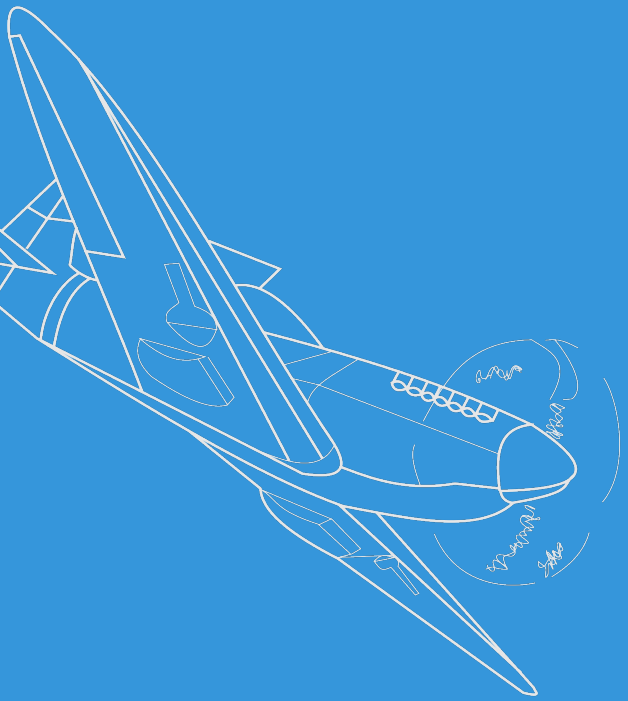
Все данные в столбце одного типа.

blog_post

id	title	category_id
1	Python	1
3	Python for dummies	1
15	Perl	2
7	C++ in 21 day	3

blog_category

id	title	auditory
1	Python	4000
2	Perl	2000
3	C++	10000
4	Java	8000

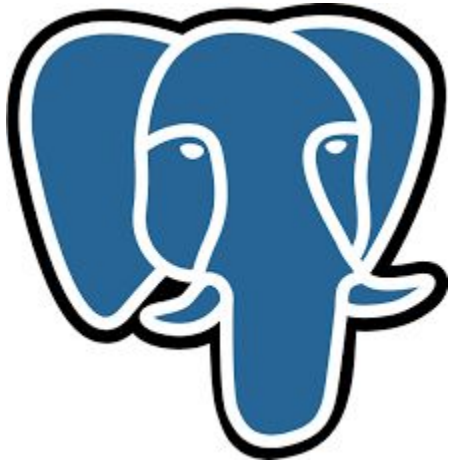


PostgreSQL

Клиент psql: установка, запуск, создание БД и новых пользователей.

Установка PostgreSQL

```
# Установка на Ubuntu
sudo apt install postgresql-10
# Установка на MacOS
brew install postgresql@10
# Проверка подключения
sudo -u <USER_NAME> psql
```



Сильные стороны PostgreSQL

- высокопроизводительные и надёжные механизмы транзакций и репликации;
- расширяемая система встроенных языков программирования;
- наследование;
- возможность индексирования геометрических объектов и наличие базирующегося на ней расширения PostGIS;
- встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации;
- расширяемость.

Пользователи и базы из коробки

Пользователи:

- `postgres` — супер-пользователь внутри Postgres, может все;

Базы данных:

- `template1` — шаблонная база данных;
- `template0` — шаблонная база данных на всякий случай;
- `postgres` — база данных по-умолчанию, копия `template1`.

Пользователи и базы из коробки

Создать пользователя и базу:

```
postgres=# CREATE USER quack WITH password 's3cr3t';
CREATE ROLE
postgres=# CREATE DATABASE quack_db OWNER quack;
CREATE DATABASE
```

Проверить подключение

```
$ psql --host=localhost --user=quack quack
Password for user quack: *****
```

Использование psql

Подключение через UNIX сокет, имя пользователя совпадает с пользователем Linux:

```
$ psql db_name
```

Подключение через TCP сокет:

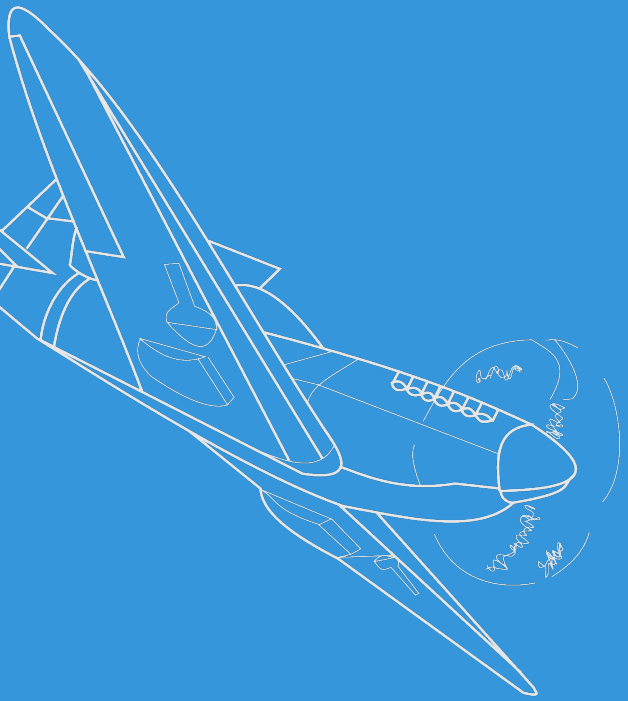
```
$ psql --host=127.0.0.1 --user=db_user db_name
```

Если вы хотите подключаться к своей базе без ввода пароля:

```
postgres=# CREATE USER your_linux_user;
postgres=# GRANT ALL ON DATABASE db_name TO your_linux_user;
```

Язык SQL

- `\?` — показать список команд;
- `\du` — показать список пользователей с привилегиями;
- `\l` — показать список баз данных;
- `\c db_name2` — подключиться к другой базе данных;
- `\dt` — показать список таблиц;
- `\d table_name` — показать колонки таблицы
- `\x` — переключить режим вывода
- `\q` — выход из psql



Язык SQL

Типы данных, синтаксис, основные виды операции.

Типы данных

- `smallint`, `integer`, `bigint` — целое, 2/4/8 байт;
- `smallserial`, `serial`, `bigserial` — целое, 2/4/8 байт, автоувеличение;
- `timestamp` — дата и время, с точностью до микросекунд;
- `text` — строка произвольной длины;
- `varchar` — строка ограниченное длины;
- `char` — строка ограниченное длины, дополненная пробелами;
- `uuid` — UUID;
- `jsonb` — JSON документ;

Виды операций

- `SELECT` — выборка данных;
- `UPDATE` — обновление значений столбцов;
- `DROP` — удаление;
- `ALTER` — изменение;
- `INSERT` — вставка;
- `CREATE` — создание;
- `JOIN` — объединение;
  - `INNER JOIN` (или просто JOIN);
  - `LEFT OUTER JOIN`
  - `RIGHT OUTER JOIN`
  - `FULL OUTER JOIN`
  - `CROSS JOIN`

Создание таблиц


```
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    nick TEXT NOT NULL UNIQUE CHECK (length(nick) < 32),
    name TEXT NOT NULL CHECK (length(name) < 32)
)
```

Создание таблиц (2)


```
CREATE TABLE messages (
    message_id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL REFERENCES users(user_id),
    content TEXT NOT NULL CHECK (length(content) < 65536),
    added_at TIMESTAMP NOT NULL DEFAULT NOW()
)
```

Изменение и удаление таблиц


```
ALTER TABLE users
    ADD COLUMN avatar TEXT DEFAULT NULL,
    DROP CONSTRAINT users_name_check,
    ADD CONSTRAINT users_name_check
        CHECK (length(name) < 64);
DROP TABLE users;
DROP TABLE users CASCADE; -- не повторять в production :)
```

Добавление данных


```
INSERT INTO users (nick, name)
            VALUES  ('mort.rainey', 'Морт Рейни'),
                    ('john.shooter', 'Кокни Шутер');

INSERT INTO users VALUES (5, 'todd.downey', 'Тодд Дауни');
```

Обновление данных


```sql
UPDATE users SET name = 'Не такой как все'
WHERE user_id = 2;

DELETE FROM users WHERE user_id % 2 = 0;
```

Выборка данных

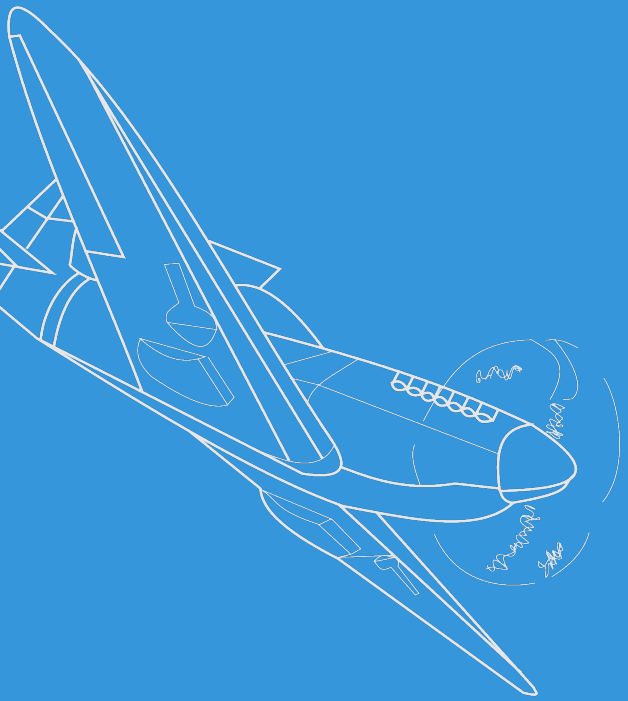

```
SELECT message_id, content, added_at::DATE
FROM messages
WHERE user_id = 3
  AND message_id < 100500
ORDER BY added_at DESC
LIMIT 10
```

Выборка из нескольких таблиц


```
SELECT nick AS author, name, messages.*
FROM messages
JOIN users USING (user_id)
WHERE user_id = 3
  AND message_id < 100500
ORDER BY added_at DESC
LIMIT 10
```

I KNOW SQL

http://www.sql-ex.ru



Django и PostgreSQL

ORM, как подключить БД к Django-приложению, создание и миграция моделей.

Добавить БД в settings.py


```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'quack_db',
        'USER': 'quack',
        'PASSWORD': 's3cr3t',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

Добавить БД в settings.py


```
# Делаем миграцию
./manage.py migrate

# Создаем суперпользователя
./manage.py createsuperuser

# и запускаем сервер
./manage runserver
```

Некоторые полезные опции

- `./manage.py dbshell` — запустить клиент базы данных;
- `./manage.py showmigrations` — показать историю миграций;
- `./manage.py makemigrations` — создать миграции;
- `./manage.py migrate` — применить миграции;
- `./manage.py shell` — запустить python shell;
- `./manage.py validate` — проверть структуру моделей.

Типы полей (1)

- `IntegerField`
- `AutoField`
- `BooleanField`
- `CharField`
- `EmailField`
- `DateField`
- `DateTimeField`
- `FloatField`
- `TextField`

Типы полей (2)

- Один-к-одному → OneToOneField
- Один-ко-многим → ForeignKey
- Многим ко многим → ManyToManyField

SQL-запросы напрямую из Django (1)


```
from django.db import connection

def my_custom_sql(self):
    with connection.cursor() as cursor:
        cursor.execute("SELECT foo FROM bar WHERE baz =
%s", [self.baz])
        row = cursor.fetchone()

    return row
```

SQL-запросы напрямую из Django (2)


```
>>> people = Person.objects.raw('SELECT *,
age(birth_date) AS age FROM myapp_person')
>>> for p in people:
...     print(f"{p.first_name} is {p.age}.")
```

Домашнее задание № 4

1. Установить Postgres, создать нового пользователя и БД и настроить доступ — 5 баллов;
2. Спроектировать базу данных проекта, подготовить модели и мигрировать их в БД (должна присутствовать как минимум одна из связей OneToOne, ForeignKey, ManyToMany) — 5 баллов.

Рекомендуемая литература

Документация Django
Документация PostgreSQL

Для саморазвития (опционально)
Чтобы не набирать двумя пальчиками

Спасибо за внимание!

Антон Кухтичев

a.kukhtichev@mail.ru

@toshunster