

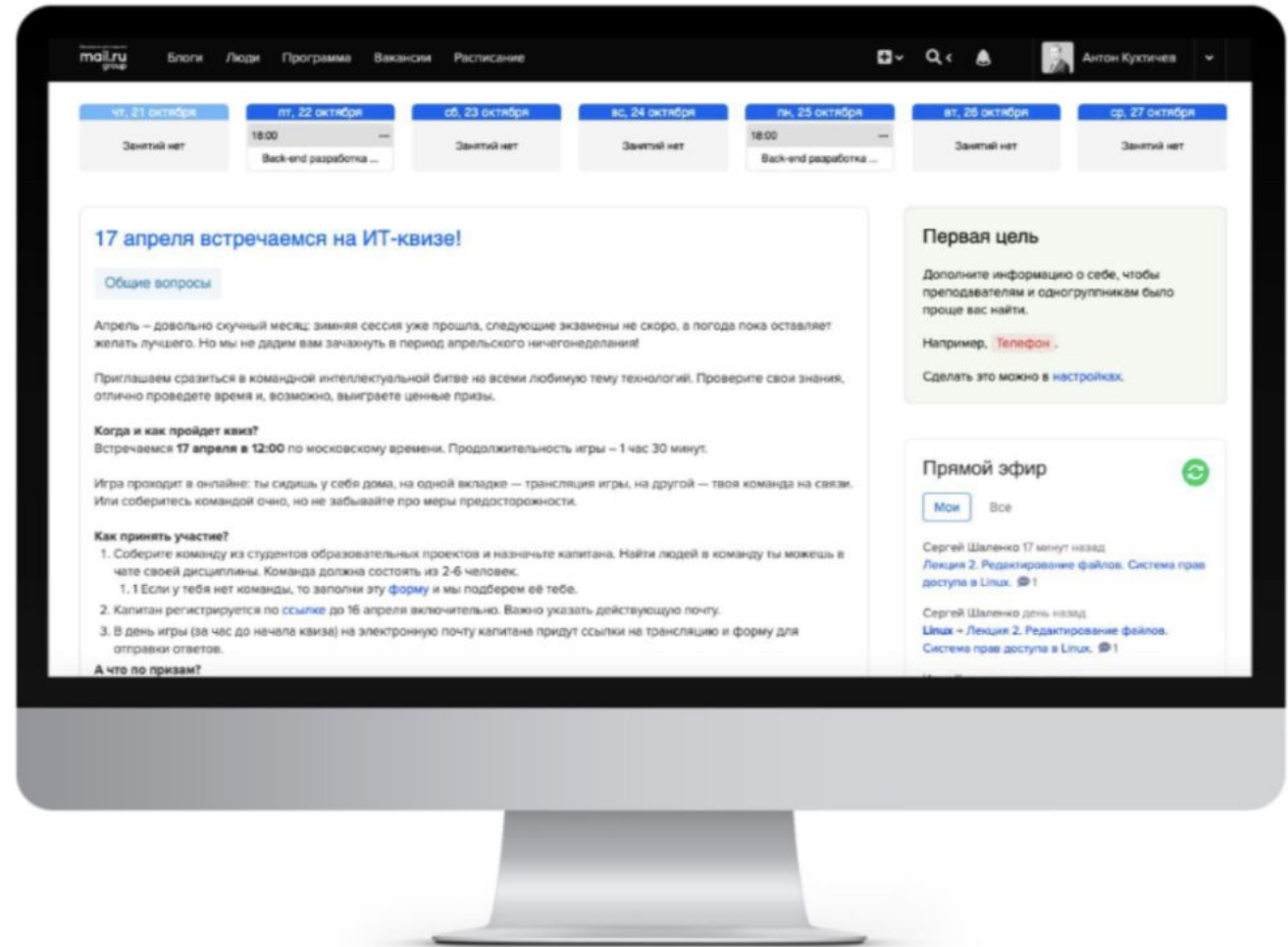
Бэкенд-разработка на Python. Лекция №5. Введение в базы данных

Алена Елизарова



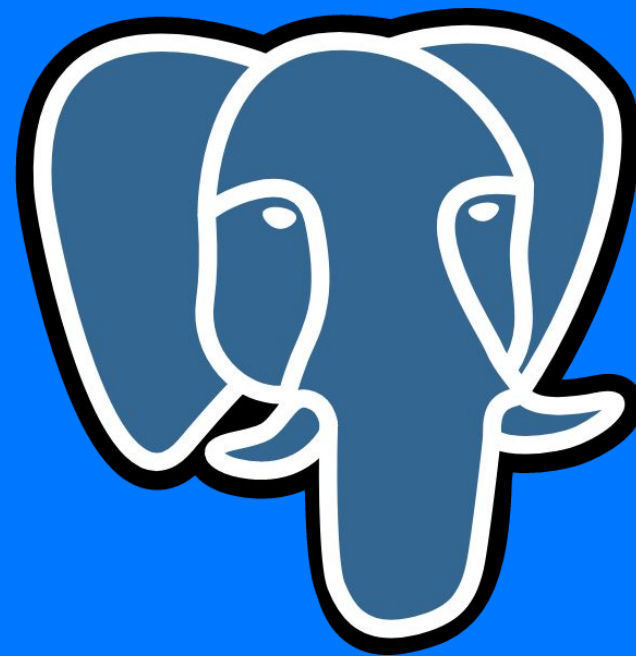
образование

Напоминание отметиться на портале

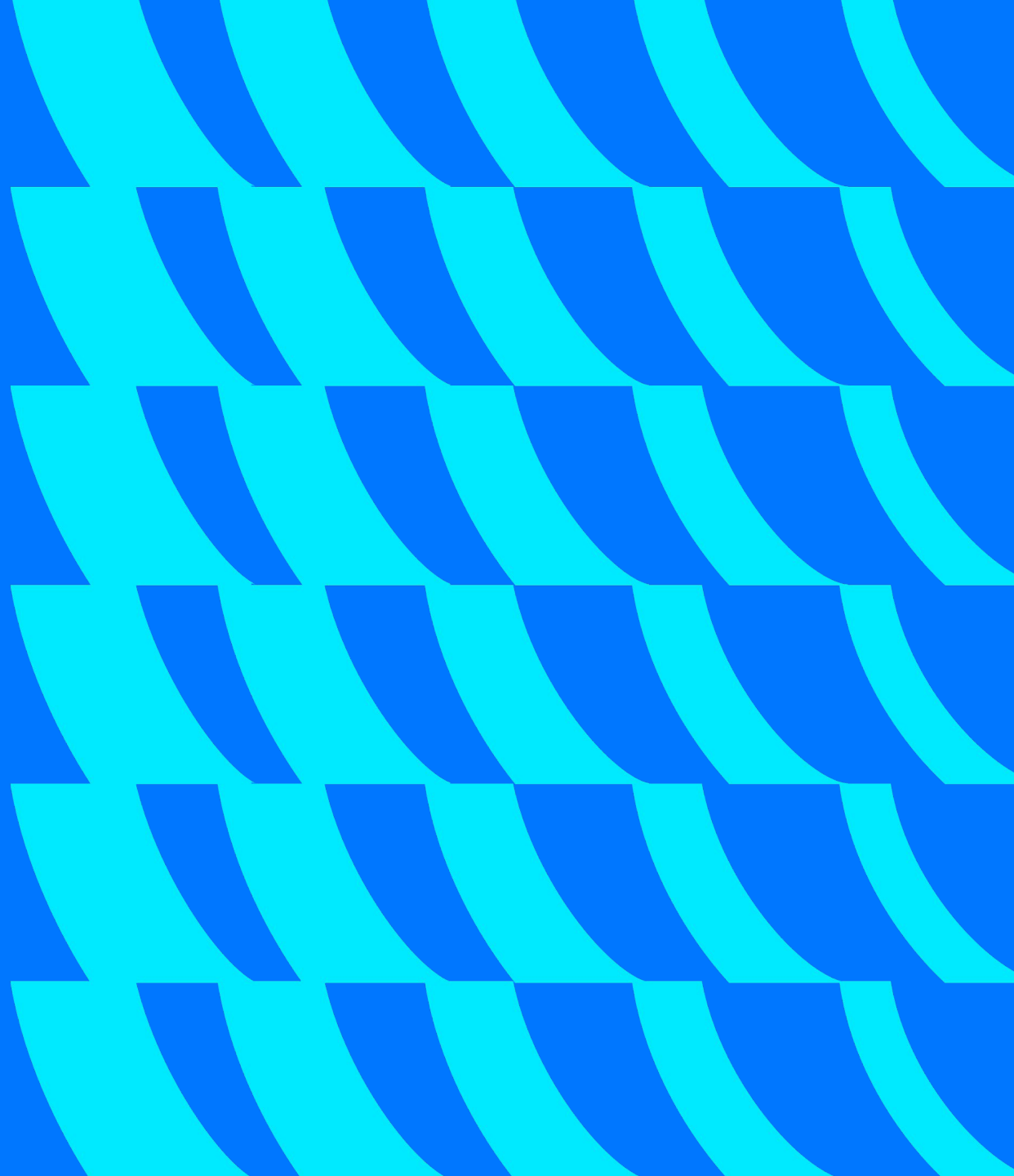


План занятия

1. БД. Определение и понятия
2. Виды БД
3. PostgreSQL
4. Язык SQL
5. Django и PostgreSQL
6. Подключение БД к проекту
7. Миграции



Базы данных



БД/СУБД

База данных — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных

Типы БД

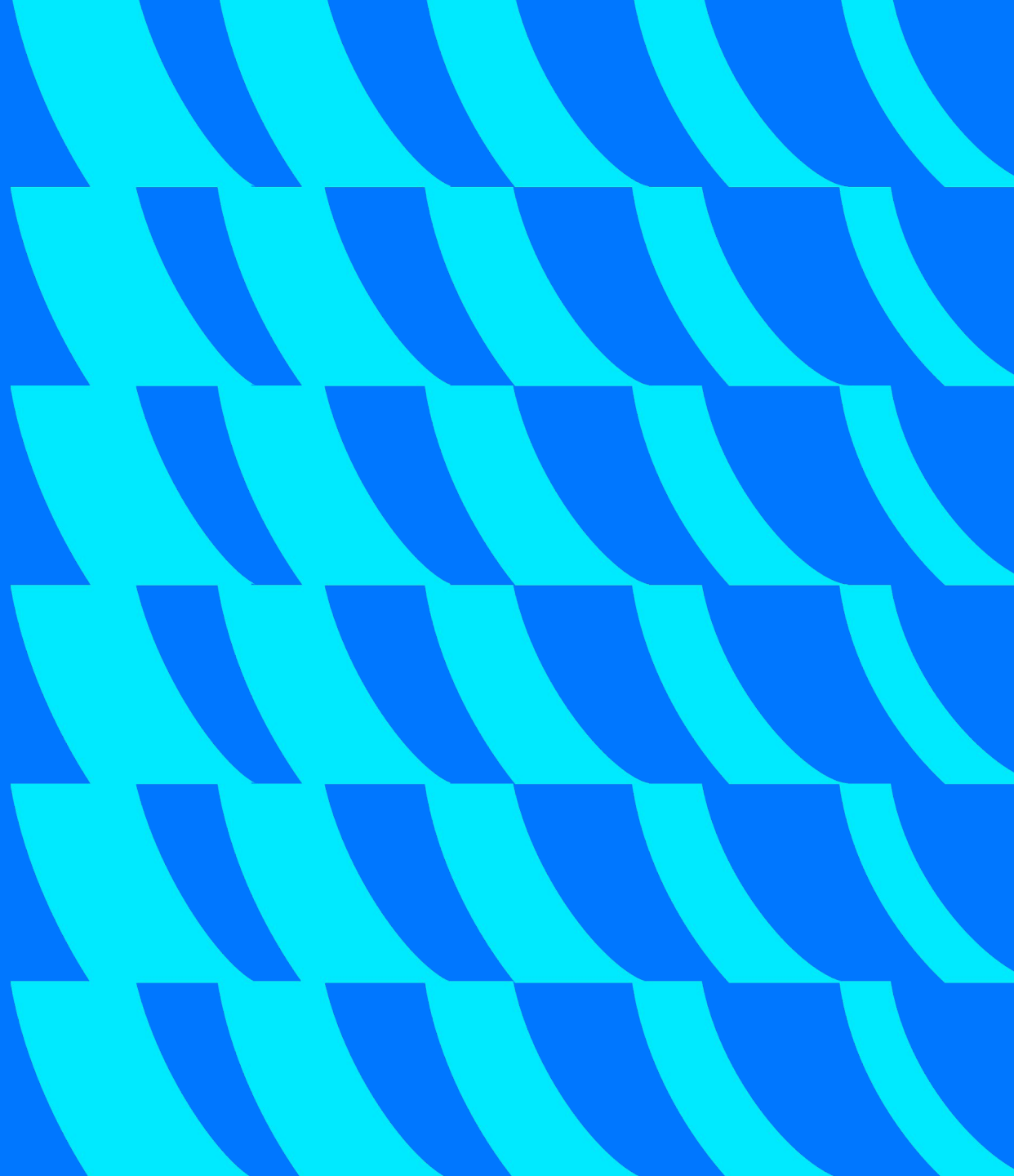
Реляционные

MySQL, Oracle DB, PostgreSQL

Нереляционные

MongoDB, Cassandra, Google BigTable, Vertica

PostgreSQL



psql

Клиент psql:

- установка, запуск, создание БД и новых пользователей

PostgreSQL

Установка на Ubuntu

```
sudo apt install postgresql
```

Установка на MacOS

```
brew install postgresql
```

Проверка подключения

```
sudo -u <USER_NAME> psql
```

PostgreSQL. Преимущества

- надежность и устойчивость
- кроссплатформенность;
- доступность
- расширяемость

Пользователи и БД и 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.

Создание БД

Создать пользователя и базу:

```
postgres=# CREATE USER forum_user WITH password 'pass';
```

```
CREATE ROLE
```

```
postgres=# CREATE DATABASE forum_db OWNER forum_user;
```

```
CREATE DATABASE
```

Проверить подключение

```
$ psql --host=localhost --user=forum_user forum_db
```

```
Password for user forum_user: *****
```

Подключение к БД

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:

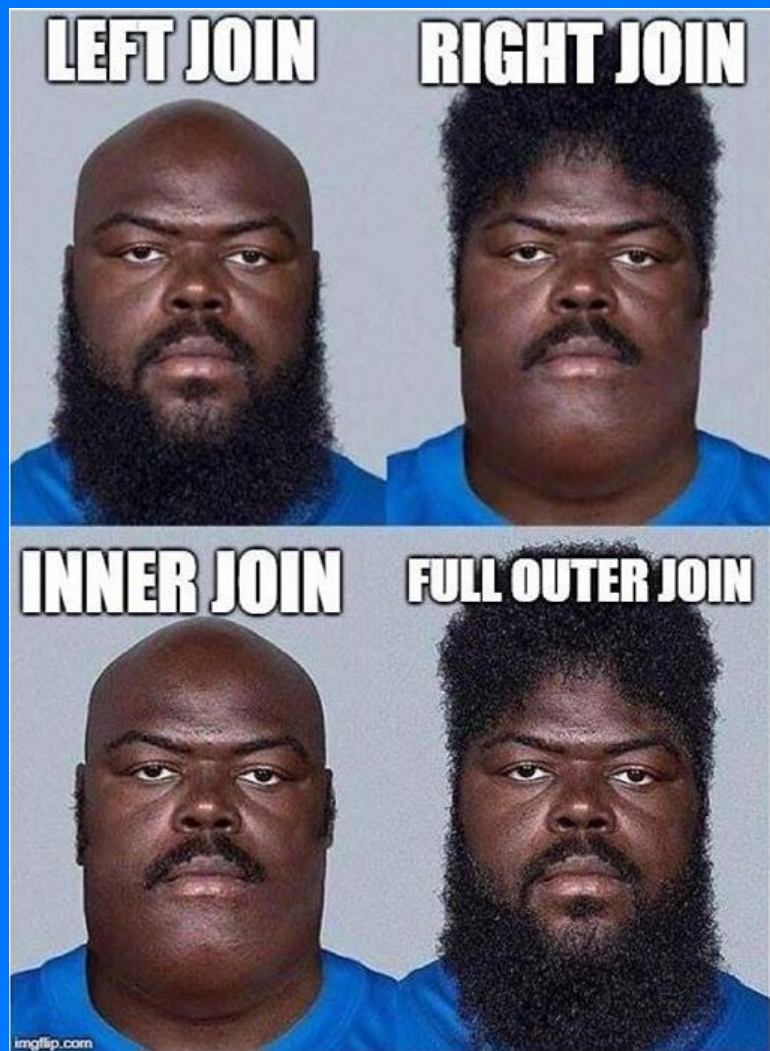
```
postgres=# CREATE USER your_linux_user;
```

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

Команды psql

- `\?` – 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



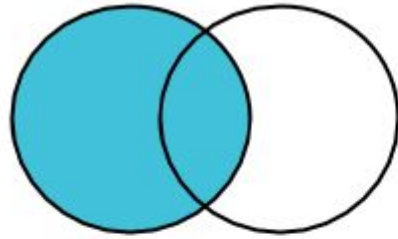
Типы данных

- 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 ограниченной длины;
- char — строка ограниченной длины, дополненная пробелами;
- uuid — UUID;
- jsonb — JSON документ;

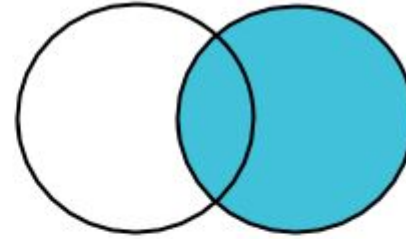
Типы операций

- 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

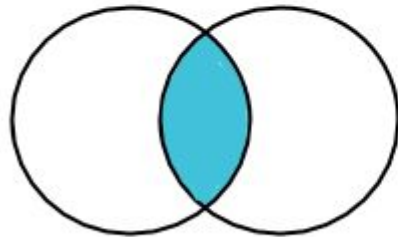
JOIN



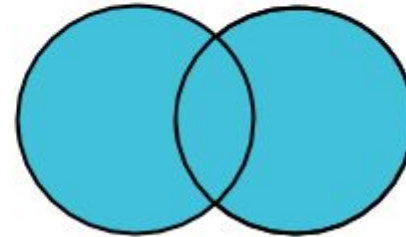
Left Join



Right Join



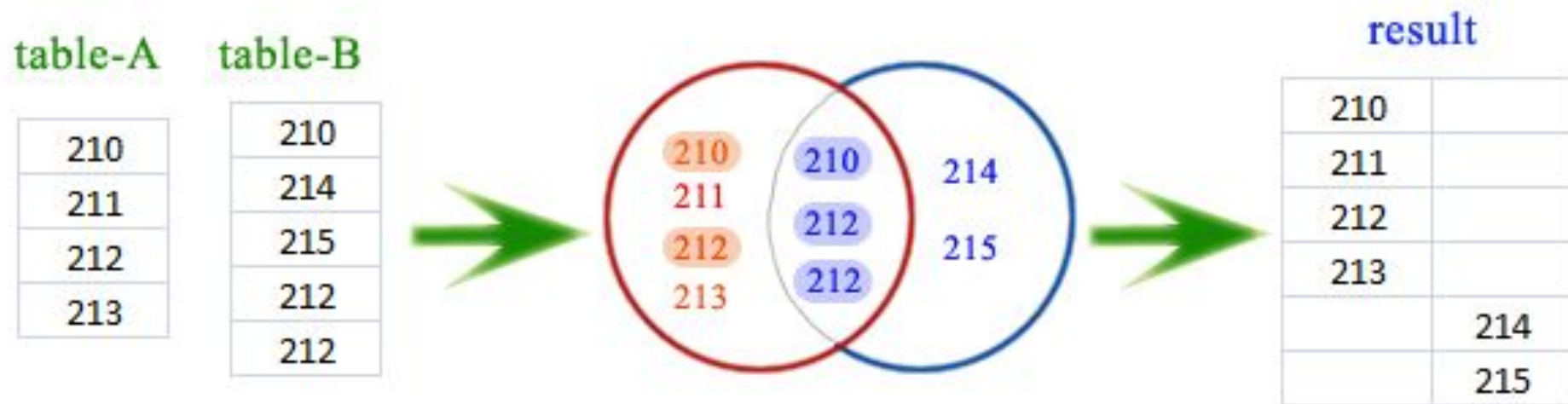
Inner Join



**Full Outer
Join**

JOIN

Left Join or Left outer Join



****** all rows from left table table-A and matching rows from right table table-B and unmatched rows from right table table-B.

Создание таблиц

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

Создание таблиц

```
CREATE TABLE posts (  
    id SERIAL PRIMARY KEY,  
    user_id INTEGER NOT NULL REFERENCES users(user_id),  
    content TEXT NOT NULL CHECK (length(content) < 65536),  
    created_at TIMESTAMP NOT NULL DEFAULT NOW()  
);
```

Изменение и удаление таблиц

```
ALTER TABLE users ADD COLUMN info TEXT DEFAULT NULL;
```

```
DROP TABLE users;
```

Добавление данных

```
INSERT INTO users (username, name)  
  VALUES ('a.elizarova', 'Alena Elizarova'),  
          ('s.matveeva', 'Svetlana Matveeva');
```

```
INSERT INTO users VALUES (5, 'i.ivanov', 'Ivan Ivanov');
```

Обновление данных

```
UPDATE users SET name = 'Svetlana Ivanova'  
WHERE user_id = 2;
```

```
DELETE FROM users WHERE user_id % 5 = 0;
```


Выборка данных

```
SELECT id, content, created_at::DATE FROM posts
WHERE user_id = 3 AND id < 100500
ORDER BY created_at DESC
LIMIT 10;
```

Выборка из нескольких таблиц

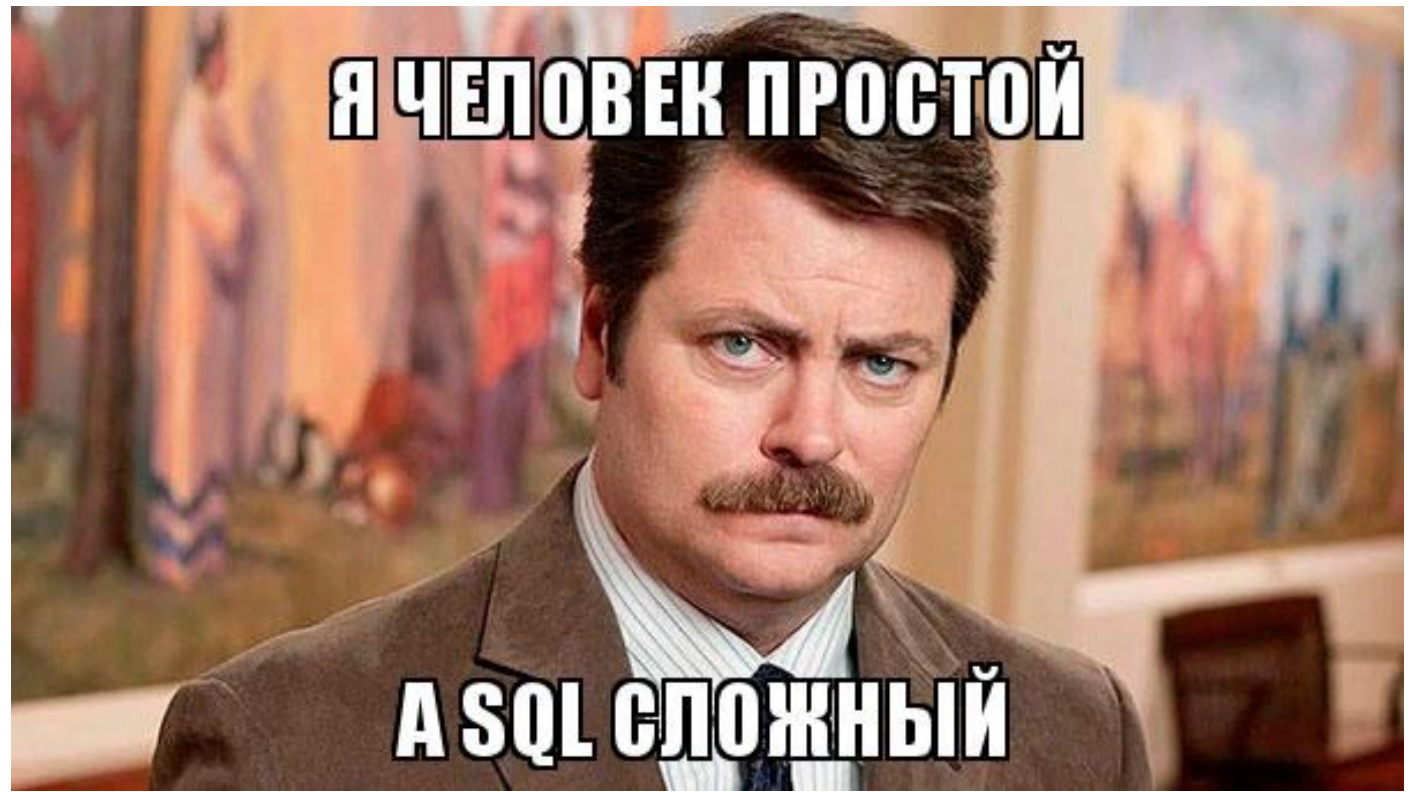
```
SELECT username AS author, name, posts.*  
FROM posts  
JOIN users USING (user_id)  
WHERE user_id = 2 AND id < 100500  
ORDER BY created_at DESC  
LIMIT 10;
```

Выборка из нескольких таблиц

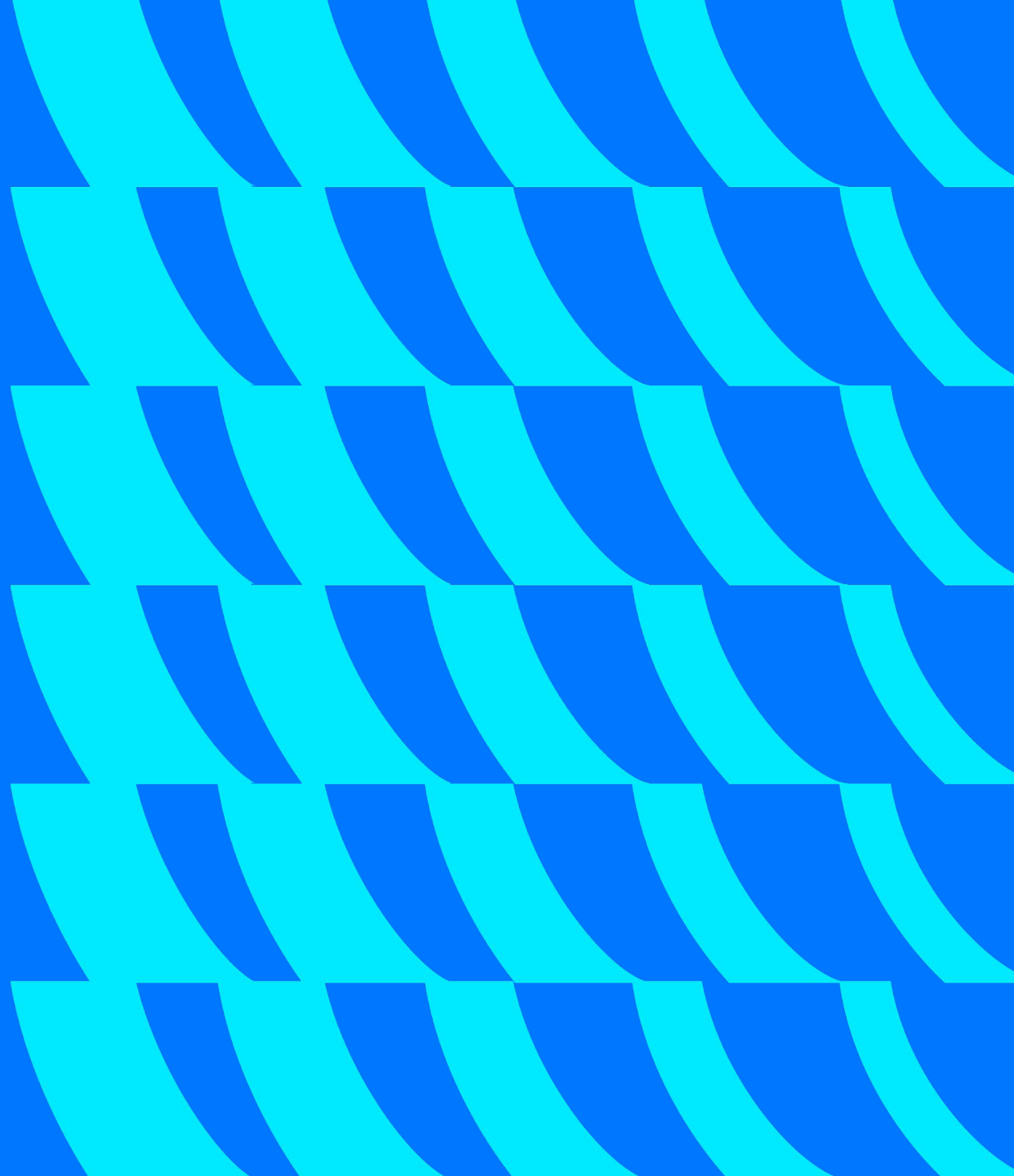
```
SELECT username AS author, name, posts.*  
FROM posts  
JOIN users ON posts.user_id = users.user_id  
WHERE users.user_id = 2 AND id < 100500  
ORDER BY created_at DESC  
LIMIT 10;
```

Тренировка SQL

<https://www.sql-ex.ru/>



Django и PostgreSQL



Типы полей в моделях Django

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

Типы полей. Связи

- Один-к-одному → `OneToOneField`
- Один-ко-многим → `ForeignKey`
- Многие ко многим → `ManyToManyField`

ForeignKey

```
class Movie(models.Model):  
    title = models.CharField()  
    category = models.ForeignKey(Category, null=True, on_delete=models.CASCADE)
```

- `models.CASCADE` – при удалении категории удаляются и фильмы с такой категорией;
- `models.PROTECT` – запрещает удалять категории, пока есть фильмы с такой категорией;
- `models.SET_NULL` – фильмы останутся в БД даже при удалении категории, но значение в поле `category` у фильмов изменится на `None`.
- `models.SET_DEFAULT` – работает как `SET_NULL`, но вместо `None` выставляет значение по-умолчанию.

Параметры у поля в модели

`verbose_name` — человекочитаемое название поля;

`null` — может ли быть поле NULL в БД, по умолчанию — False

`blank` — может ли быть поле пустым в форме, по умолчанию — False

`choices` — выбор значения из списка;

`default` — значение по умолчанию;

`db_index` — нужно ли создать индекс в БД;

`help_text` — подробное описание поля;

`primary_key` — первичный ключ;

`unique` — уникальное ли поле;

`editable` — можно ли поле изменять;

Настройка БД проекта

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.postgresql_psycopg2',  
        'NAME': 'forum_db',  
        'USER': 'forum_user',  
        'PASSWORD': 'pass',  
        'HOST': '127.0.0.1',  
        'PORT': 5432,  
    }  
}
```

Настройка БД проекта

Делаем миграцию

```
./manage.py migrate
```

Создаем суперпользователя

```
./manage.py createsuperuser
```

и запускаем сервер

```
./manage.py runserver
```

Полезные команды

- `./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 sqlmigrate <app> <migration id>` – вывести SQL-код для миграции;
- `./manage.py shell` – запустить python shell;
- `./manage.py validate` – проверить структуру моделей.

Откат миграций

В общем случае команда выглядит так:

```
$ ./manage.py migrate <ваше приложение> <название предыдущей миграции>
```

Откат всех миграций!

```
$ ./manage.py migrate <ваше приложение> zero
```

Примерчик:

```
$ ./manage.py showmigrations movies
```

movies

```
[X] 0001_initial
```

```
[X] 0002_movie_category
```

```
[X] 0003_auto_20201005_1456
```

```
$ ./manage.py migrate movies 0002_movie_category
```

Создание пустой миграции

```
$ ./manage.py makemigrations --empty <ваше приложение>
```

```
from django.db import migrations
class Migration(migrations.Migration):
    dependencies = [
        ('yourappname', '0001_initial'),
    ]
    operations = [
        migrations.RunPython(forwards_func, reverse_func),
    ]
```

Выполнение SQL из Django

```
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
```

Выполнение SQL из Django

```
>>> 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

1. Установить Postgres, создать нового пользователя и БД и настроить доступ;
2. Спроектировать базу данных проекта, подготовить модели и мигрировать их в БД.
3. Необходимо реализовать как минимум одно из отношений:
 - Один-ко-многим;
 - Многие-ко-многим;
 - Один-к-одному.

Бонус. Что спросят на собеседовании

ACID

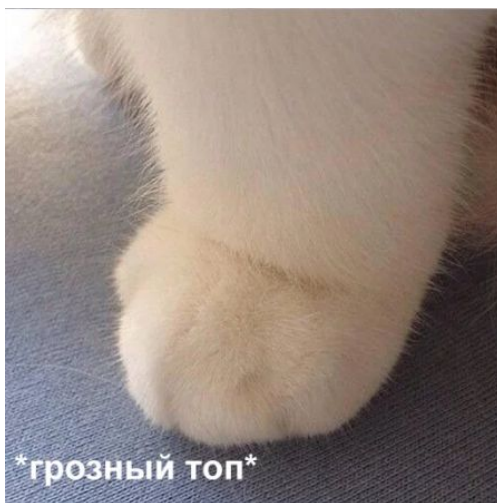
Locking, deadlocks

Транзакции. Уровни изоляции транзакций

Репликация и шардирование

Индексы. Как устроены, плюсы и минусы

**Не забудьте оставить
отзыв на портале**



Спасибо
за внимание

