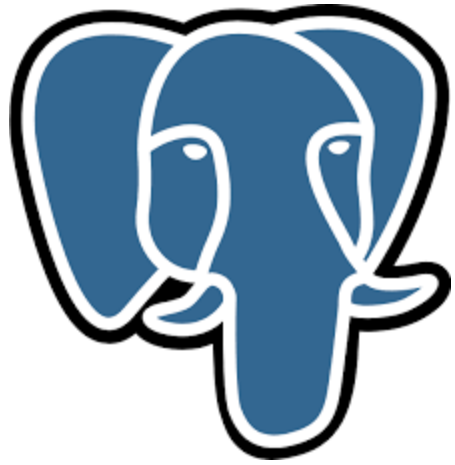
PostgreSQL

Установка PostgreSQL

Установка

```
sudo apt install postgresql-9.5
```



Проверка подключения

```
sudo -u postgres psql
```

Для пользователя `postgres` - `peer` авторизация

`/etc/postgresql/9.5/main/pg_hba.conf`

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`

Пользователь и база проекта

Создать пользователя и базу

```
postgres=# CREATE USER quack PASSWORD 'quack';
CREATE ROLE

postgres=# CREATE DATABASE quack 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;
```

Использование psql (2)

- `\?` показать список команд
- `\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

Язык SQL

Типы данных

- `integer`, `bigint` - целое, 4/8 байт
- `serial`, `bigserial` - целое, 4/8 байт, авто-инкремент
- `timestamp` - дата и время, с точностью до микросекунд
- `text` - строка произвольной длинны
- `uuid` - UUID
- `jsonb` - JSON документ

https://www.postgresql.org/docs/9.5/static/datatype.html

Создание таблиц

```
CREATE TABLE users (

    user_id SERIAL PRIMARY KEY,

    user_nick TEXT NOT NULL UNIQUE
        CHECK (length(user_name) < 32),

    user_name TEXT NOT NULL
        CHECK (length(user_name) < 32)
)
```

Создание таблиц (2)

```
CREATE TABLE messages (

    message_id SERIAL PRIMARY KEY,

    user_id INTEGER NOT NULL
        REFERENCES users(user_id),

    message_text TEXT NOT NULL
        CHECK (length(message_text) < 65536),

    message_added_at TIMESTAMP NOT NULL DEFAULT NOW()
)
```

Изменение и удаление таблиц

```
ALTER TABLE users
    ADD COLUMN user_avatar TEXT DEFAULT NULL,
    DROP CONSTRAINT users_user_name_check,
    ADD CONSTRAINT users_user_name_check
        CHECK (length(user_name) < 64);

DROP TABLE users;

DROP TABLE users CASCADE; -- не повторять в production :)
```

Добавление данных

```
INSERT INTO users (user_nick, user_name)
VALUES ('vasily.pupkin', 'Василий Пупкин'),
       ('another.user', 'Другой Юзер');

INSERT INTO users
VALUES (5, 'the.good', 'Клинт Иствуд');
```

Обновление и удаление данных

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

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

Выборка из таблицы

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

Выборка из нескольких таблиц

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

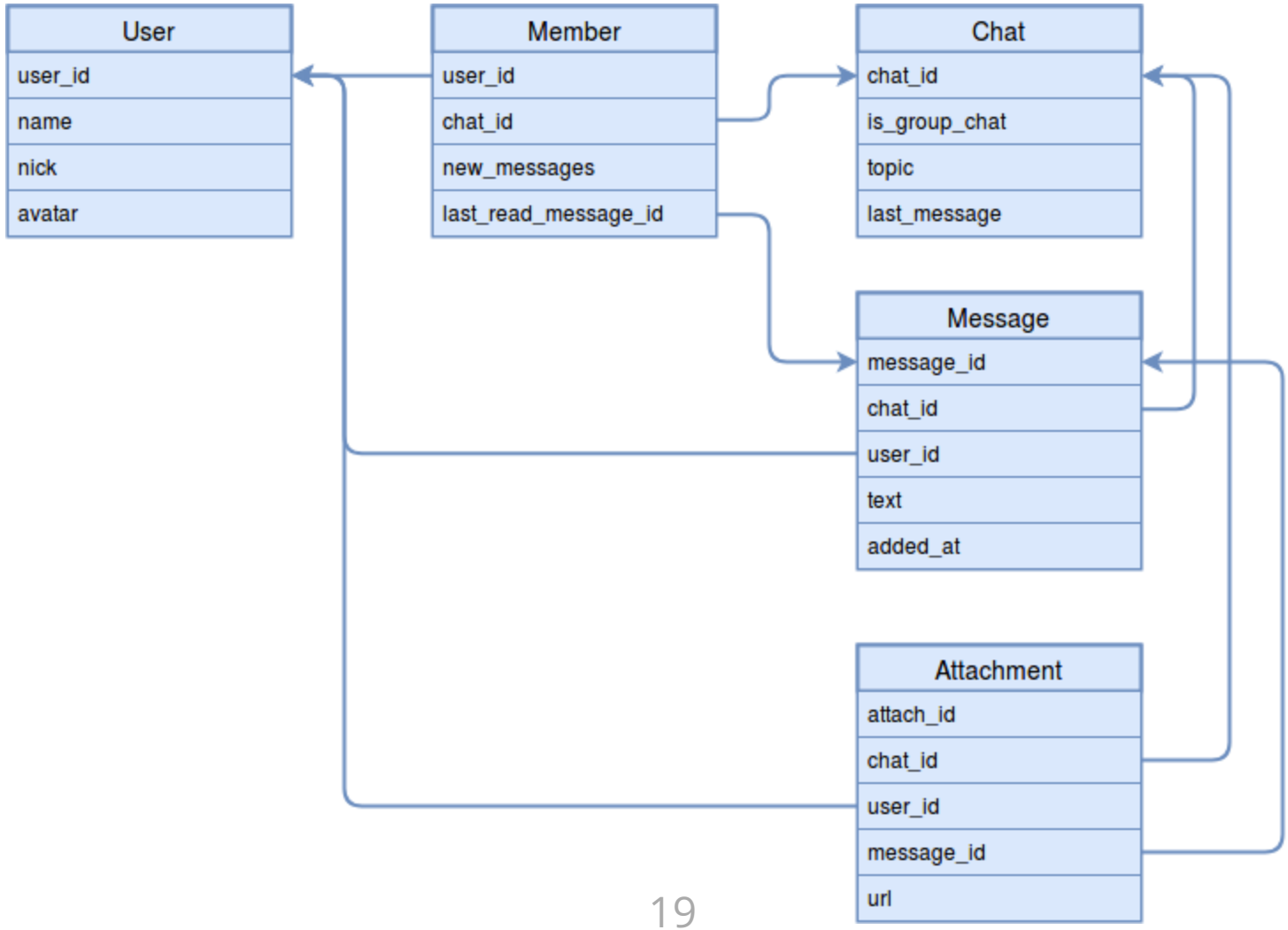
I KNOW SQL

http://www.sql-ex.ru

Проектирование базы данных

Правила проектирования

- Отдельная сущность - отдельная таблица
- Использовать синтетические первичные ключи
- Отношение 1:N - внешний ключ
- Отношение N:M - промежуточная таблица


User
user_id
name
nick
avatar
Member
user_id
chat_id
new_messages
last_read_message_id
Chat
chat_id
is_group_chat
topic
last_message
Message
message_id
chat_id
user_id
text
added_at
Attachment
attach_id
chat_id
user_id
message_id
url

Работа с СУБД в Python

Использование СУБД в Python

PEP-249 - спецификация интерфейса к СУБД

`psycopg2` - Python библиотека, реализующая этот интерфейс

Основные объекты

- **Connection** - соединение с базой данных
- **Cursor** - промежуточный объект для осуществления запросов и получения результатов

Пример использования

```
import psycopg2
try:
    connection = psycopg2.connect(
        user = "quack", password = "***",
        host = "127.0.0.1", database = "quack")
    cursor = connection.cursor()
    cursor.execute("SELECT version();")
    row = cursor.fetchone()
    print("You are connected to - ", row, "\n")
except psycopg2.Error as error :
    print("Error while connecting to PostgreSQL", error)
finally:
    if connection:
        connection.close()
        if cursor:
            cursor.close()
```

Методы Connection

- `cursor()` - создать новый объект Cursor
- `cursor(cursor_factory=psycopg2.extras.DictCursor)`
- `commit()` - COMMIT, сохранить транзакцию в СУБД
- `rollback()` - ROLLBACK, отменить транзакцию в СУБД
- `close()` - закрыть соединение с СУБД

Методы Cursor

- `execute(sql, params)` - выполнить SQL запрос
- `callproc(procname, params)` - вызвать хранимую процедуру
- `fetchone()` - получить первую строку результата
- `fetchall()` - получить все строки результата
- `close()` - закрыть курсор, освободить ресурсы СУБД
- `.lastrowid` - (аттрибут) значение SERIAL ключа после INSERT

Транзакции

Транзакция - набор изменений в базе данных, которые либо применяются одновременно, либо не применяются вообще. Все изменения в СУБД осуществляются в рамках транзакций.

`COMMIT` - сохранение транзакции в СУБД. После завершения СУБД дает гарантию того что данные не будут потеряны. Дорогостоящая операция, осуществляет запись на диск.

`ROLLBACK` - отмена транзакции. Все изменения, произведенные в рамках транзакции, отменяются.

Особенности СУБД в Web

Особенности Web приложений

- HTTP запросы - быстрые (~50мс)
- Но их много (~100 / сек)
- В рамках одного HTTP запроса ~ 10-20 SQL запросов
- HTTP запросы могут быть успешными (2xx/3xx) или завершаться ошибкой (4xx/5xx)

Следствия

- Соединения с СУБД нужно кешировать
  - В течение запроса
  - В течение жизни Application Server
  - Использовать внешний пул соединений (pg_bouncer)
- Удобно: 1 HTTP запрос = 1 транзакция

Организация кода во Flask

!!! Отделить код для работы с базой от контроллеров **!!!**

```
├── app
│   ├── app.py              # Flask приложение
│   ├── handlers.py         # Контроллеры
│   ├── db.py               # Утилиты для работы с СУБД
│   ├── model.py            # Модели, конкретные запросы к СУБД
│   └── settings.py         # Настройки подключения
└── sql
    ├── 001_initial.sql
    └── 002_migration.sql
```

Утилиты для СУБД

```
# app/db.py

import flask
import psycopg2
import settings

def get_connection():
    if not hasattr(flask.g, 'dbconn'):
        flask.g.dbconn = psycopg2.connect(
            database=settings.DB_NAME, host=settings.DB_HOST,
            user=settings.DB_USER, password=settings.DB_PASS)
    return flask.g.dbconn
```

Утилиты для СУБД (2)

```
# app/db.py

import psycopg2.extras

def get_cursor():
    return get_connection().cursor(
        cursor_factory=psycopg2.extras.DictCursor)

def query_one(sql, **params):
    with get_cursor() as cur:
        cur.execute(sql, params)
        return dict(cur.fetchone())

# TODO: query_all => [ {} ]
```

Транзакции и HTTP запросы

```
# app/db.py
import flask

def _rollback_db(sender, exception, **extra):
    if hasattr(flask.g, 'dbconn'):
        conn = flask.g.dbconn
        conn.rollback()
        conn.close()
        delattr(flask.g, 'dbconn')

flask.got_request_exception.connect(_rollback_db, app)

# TODO: _commit_db
```

Модели в Web приложении

```
# app/model.py
import db

def list_messages_by_chat(chat_id, limit):
    return db.query_all("""
        SELECT user_id, user_nick, user_name,
               message_id, message_text, message_added_at
        FROM messages
        JOIN users USING (user_id)
        WHERE chat_id = %(chat_id)s
        ORDER BY message_added_at DESC
        LIMIT %(limit)s
    """, chat_id=int(chat_id), limit=int(limit))
```

И тогда в контроллере

```
# app/handlers.py
from flask import request, jsonify
import model
from app import app

@app.route('/messages/')
def messages():
    chat_id = int(request.args.get('chat_id'))
    messages = model.list_messages_by_chat(chat_id)
    return jsonify(messages)
```

Спасибо за внимание!