**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное

образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

ОТЧЕТ по лабораторной работе

«Этап 2. Начало работы»

по дисциплине «**Проектирование баз данных**»

|  |  |
| --- | --- |
| Автор: | Константинова Елизавета Анатольевна |
| Факультет: | ФИТиП |
| Группа: | M3203 |
| Преподаватели: | Шевчик Софья Владимировна |
|  |  |

ИТМО

Санкт-Петербург 2024

Этап 2. Начало работы

**Задачи:** развернуть СУБД, создание таблиц и заполнение данными.

На втором этапе необходимо развернуть СУБД PostgreSQL на сервере или

локальной машине внутри docker контейнера:

docker-compose.yml, с помощью которого развертывается бд

version**:** '3.8'

services**:** db**:** build**:** .  
 restart**:** always  
 env\_file**:  
 -** .env  
 environment**:** POSTGRES\_DB**:** ${POSTGRES\_DB}  
 POSTGRES\_PASSWORD**:** ${POSTGRES\_PASSWORD}  
 DB\_USERNAME**:** ${DB\_USERNAME}  
 DB\_SUPERUSER**:** ${DB\_SUPERUSER}  
 DB\_VERSION**:** ${DB\_VERSION}  
 FILLING\_AMOUNT**:** ${FILLING\_AMOUNT}  
 ports**:  
 -** "5432:5432"  
 volumes**:  
 -** ./migrations:/migrations  
 **-** ./bashScripts:/bashScript  
 **-** ./bashScripts/entrypoint.sh:/docker-entrypoint-initdb.d/entrypoint.sh  
 **-** ./bashScripts/start\_migrations.sh:/docker-entrypoint-initdb.d/start\_migrations.sh

.env:

POSTGRES\_DB="discord-db"  
POSTGRES\_PASSWORD="mypassword"  
DB\_USERNAME="postgres"  
DB\_SUPERUSER="postgres"  
FILLING\_AMOUNT=100000

Идемпотентный файл миграции:

CREATE TABLE IF NOT EXISTS users (  
 user\_id SERIAL PRIMARY KEY,  
 username VARCHAR(50) NOT NULL UNIQUE,  
 email VARCHAR(100) NOT NULL UNIQUE,  
 password\_hash VARCHAR(255) NOT NULL,  
 online\_status BOOLEAN,  
 registration\_date TIMESTAMP  
 );  
  
INSERT INTO users (username, email, password\_hash, online\_status, registration\_date)  
SELECT  
 'user' || generate\_series,  
 'user' || generate\_series || '@example.com',  
 md5(random()::text),  
 random() < 0.5,  
 NOW() - (random() \* INTERVAL '365 days')  
FROM  
 generate\_series(1, ${FILLING\_AMOUNT});  
  
CREATE TABLE IF NOT EXISTS servers (  
 server\_id SERIAL PRIMARY KEY,  
 server\_name VARCHAR NOT NULL,  
 owner\_id INTEGER NOT NULL REFERENCES users(user\_id),  
 creation\_date TIMESTAMP NOT NULL  
 );  
  
INSERT INTO servers (server\_name, owner\_id, creation\_date)  
SELECT  
 'server' || generate\_series,  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 NOW() - (random() \* INTERVAL '365 days')  
FROM  
 generate\_series(1, ${FILLING\_AMOUNT});  
  
CREATE TABLE IF NOT EXISTS channels (  
 channel\_id SERIAL PRIMARY KEY,  
 channel\_name VARCHAR NOT NULL,  
 server\_id INTEGER NOT NULL REFERENCES servers(server\_id),  
 channel\_type VARCHAR NOT NULL  
 );  
  
INSERT INTO channels (channel\_name, server\_id, channel\_type)  
SELECT  
 'channel' || generate\_series,  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 CASE WHEN random() < 0.5 THEN 'text\_channel' ELSE 'voice\_channel' END  
FROM  
 generate\_series(1, ${FILLING\_AMOUNT});  
  
CREATE TABLE IF NOT EXISTS roles (  
 role\_id SERIAL PRIMARY KEY,  
 role\_name VARCHAR NOT NULL,  
 server\_id INTEGER NOT NULL REFERENCES servers(server\_id)  
 );  
  
INSERT INTO roles (role\_name, server\_id)  
SELECT  
 'role' || generate\_series,  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1  
FROM  
 generate\_series(1, ${FILLING\_AMOUNT});  
  
CREATE TABLE IF NOT EXISTS permissions (  
 permission\_id SERIAL PRIMARY KEY,  
 role\_id INTEGER NOT NULL REFERENCES roles(role\_id),  
 channel\_id INTEGER NOT NULL REFERENCES channels(channel\_id),  
 can\_send\_messages BOOLEAN,  
 can\_delete\_messages BOOLEAN,  
 can\_edit\_messages BOOLEAN,  
 can\_create\_roles BOOLEAN,  
 can\_ban\_users BOOLEAN  
 );  
  
INSERT INTO permissions (role\_id, channel\_id, can\_send\_messages, can\_delete\_messages, can\_edit\_messages, can\_create\_roles, can\_ban\_users)  
SELECT  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 random() < 0.5,  
 random() < 0.5,  
 random() < 0.5,  
 random() < 0.5,  
 random() < 0.5  
FROM  
 generate\_series(1, ${FILLING\_AMOUNT});  
  
CREATE TABLE IF NOT EXISTS invitations (  
 invitation\_id SERIAL PRIMARY KEY,  
 server\_id INTEGER NOT NULL REFERENCES servers(server\_id),  
 inviter\_id INTEGER REFERENCES users(user\_id),  
 invited\_user\_id INTEGER NOT NULL REFERENCES users(user\_id),  
 invitation\_date TIMESTAMP  
 );  
  
INSERT INTO invitations (server\_id, inviter\_id, invited\_user\_id, invitation\_date)  
SELECT  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 NOW() - (random() \* INTERVAL '365 days')  
FROM  
 generate\_series(1, ${FILLING\_AMOUNT});  
  
CREATE TABLE IF NOT EXISTS moderation\_logs (  
 log\_id SERIAL PRIMARY KEY,  
 moderator\_id INTEGER NOT NULL REFERENCES users(user\_id),  
 user\_id INTEGER REFERENCES users(user\_id),  
 action VARCHAR NOT NULL,  
 reason VARCHAR,  
 timestamp TIMESTAMP  
 );  
  
INSERT INTO moderation\_logs (moderator\_id, user\_id, action, reason, timestamp)  
SELECT  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 'action' || generate\_series,  
 'reason' || generate\_series,  
 NOW() - (random() \* INTERVAL '365 days')  
FROM  
 generate\_series(1, ${FILLING\_AMOUNT});  
  
CREATE TABLE IF NOT EXISTS emojis (  
 emoji\_id SERIAL PRIMARY KEY,  
 emoji\_name VARCHAR NOT NULL,  
 emoji\_image VARCHAR NOT NULL,  
 server\_id INTEGER REFERENCES servers(server\_id),  
 creator\_id INTEGER NOT NULL REFERENCES users(user\_id),  
 creation\_date TIMESTAMP  
 );  
  
INSERT INTO emojis (emoji\_name, emoji\_image, server\_id, creator\_id, creation\_date)  
SELECT  
 'emoji' || generate\_series,  
 'https://example.com/emoji/' || generate\_series || '.png',  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 NOW() - (random() \* INTERVAL '365 days')  
FROM  
 generate\_series(1, ${FILLING\_AMOUNT});  
  
CREATE TABLE IF NOT EXISTS user\_roles (  
 user\_role\_id SERIAL PRIMARY KEY,  
 user\_id INTEGER NOT NULL REFERENCES users(user\_id),  
 role\_id INTEGER NOT NULL REFERENCES roles(role\_id),  
 server\_id INTEGER NOT NULL REFERENCES servers(server\_id)  
 );  
  
INSERT INTO user\_roles (user\_id, role\_id, server\_id)  
SELECT  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1  
FROM  
 generate\_series(1, ${FILLING\_AMOUNT});  
  
CREATE TABLE IF NOT EXISTS user\_settings (  
 settings\_id SERIAL PRIMARY KEY,  
 user\_id INTEGER NOT NULL REFERENCES users(user\_id),  
 theme\_preference VARCHAR,  
 notification\_settings VARCHAR,  
 privacy\_settings VARCHAR,  
 other\_preferences VARCHAR,  
 language VARCHAR  
 );  
  
INSERT INTO user\_settings (user\_id, theme\_preference, notification\_settings, privacy\_settings, other\_preferences, language)  
SELECT  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 CASE WHEN random() < 0.5 THEN 'Light' ELSE 'Dark' END,  
 'notification\_setting' || generate\_series,  
 'privacy\_setting' || generate\_series,  
 'other\_preference' || generate\_series,  
 'English'  
FROM  
 generate\_series(1, ${FILLING\_AMOUNT});  
  
CREATE TABLE IF NOT EXISTS bans (  
 ban\_id SERIAL PRIMARY KEY,  
 banned\_user\_id INTEGER NOT NULL REFERENCES users(user\_id),  
 banned\_by\_user\_id INTEGER NOT NULL REFERENCES users(user\_id),  
 server\_id INTEGER NOT NULL REFERENCES servers(server\_id),  
 reason VARCHAR,  
 timestamp TIMESTAMP NOT NULL  
 );  
  
INSERT INTO bans (banned\_user\_id, banned\_by\_user\_id, server\_id, reason, timestamp)  
SELECT  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 FLOOR(RANDOM() \* ${FILLING\_AMOUNT}) + 1,  
 'reason' || generate\_series,  
 NOW() - (random() \* INTERVAL '365 days')  
FROM  
 generate\_series(1, ${FILLING\_AMOUNT});

Dockerfile:

**FROM** postgres:latest  
  
  
**EXPOSE** 5432

Точка входа (entrypoint):

**#!/bin/bash**execute\_sql() {  
 psql -U "$DB\_SUPERUSER" -d "$POSTGRES\_DB" -c "$1"  
}  
  
database\_exists() {  
 psql -U "$DB\_SUPERUSER" -lqt | cut -d \| -f 1 | grep -qw "$1"  
}  
  
**if !** database\_exists "$POSTGRES\_DB"; **then** execute\_sql "CREATE DATABASE $POSTGRES\_DB;"  
**fi**exec "$@"

Bash-скрипт, который запускает миграции, заполнение данными и создает роли и пользователей:

**#!/bin/bash**execute\_sql() {  
 psql -U "$DB\_SUPERUSER" -d "$POSTGRES\_DB" -c "$1"  
}  
  
if [[ -z "$DB\_USERNAME" ]]; then  
 echo "Username for connection doesn't specified"  
 exit 1  
fi  
  
version=$DB\_VERSION  
  
if [ -z "$version" ]; then  
 versions\_to\_run=$(find /migrations -mindepth 1 -maxdepth 1 -type d | sort | awk -F '/' '{ print $3 }')  
else  
 versions\_to\_run=$(find /migrations -mindepth 1 -maxdepth 1 -type d | sort | awk -F '/' -v v="$version" '{ print $3; if ($3 == v) exit }')  
fi  
  
for v in $versions\_to\_run; do  
 for script in $(find migrations/$v/ -name "\*.sql" -type f); do  
 TEMP\_SQL\_FILE=$(mktemp)  
 sed "s/\${FILLING\_AMOUNT}/$FILLING\_AMOUNT/g" $script > "$TEMP\_SQL\_FILE"  
 psql -U "$POSTGRES\_USER" -d "$POSTGRES\_DB" -f "$TEMP\_SQL\_FILE"  
 rm "$TEMP\_SQL\_FILE"  
 done  
done  
  
if ! execute\_sql "\\du" | grep -qw "reader"; then  
 execute\_sql "CREATE USER reader WITH ENCRYPTED PASSWORD 'readerpassword';"  
fi  
  
if ! execute\_sql "\\du" | grep -qw "writer"; then  
 execute\_sql "CREATE USER writer WITH ENCRYPTED PASSWORD 'writerpassword';"  
fi  
  
if ! execute\_sql "\\du" | grep -qw "analytic"; then  
 execute\_sql "CREATE ROLE analytic;"  
fi  
  
execute\_sql "CREATE ROLE group\_role;"  
execute\_sql "GRANT ALL PRIVILEGES ON DATABASE $POSTGRES\_DB TO group\_role;"  
  
for ((i=1; i<=4; i++)); do  
 username="user$i"  
 if ! execute\_sql "\\du" | grep -qw "$username"; then  
 execute\_sql "CREATE USER $username WITH ENCRYPTED PASSWORD 'password$i';"  
 fi  
 execute\_sql "GRANT group\_role to $username;"  
done  
  
execute\_sql "GRANT SELECT ON TABLE roles TO analytic;"  
  
execute\_sql "ALTER USER reader WITH NOCREATEDB;"  
execute\_sql "ALTER USER writer WITH CREATEDB;"

Итоговая структура проекта:

A black and white text

Description automatically generated

A screenshot of a computer

Description automatically generated