ИТМО

Проектирование баз данных
Этап 2. Начало работы

Кравченкова Елизавета М3203

15 апреля 2024 г.

Задача: Развернуть СУБД, создать таблиц и заполненить данными

1 Развертывание СУБД PostgreSQL

Для того, чтобы развернуть СУБД PostgreSQL надо прописать в файл docker-compose.yaml следующие строки:

```
version: "3.9"

name: "balroom-service"

services:
  postgres:
    container_name: postgresDB
    image: postgres:14.5
    restart: unless-stopped
    env_file:
      - .env
    ports:
      - 5432:5432
    environment:
      - POSTGRES_USER=${DB_USER}
      - POSTGRES_DB=${DB_NAME}
      - POSTGRES_PASSWORD=${DB_PASSWORD}
```

И надо создать файл .env, через который подаются переменные среды:

```
DB_NAME=postgres
DB_USER=postgres
DB_PASSWORD=postgres
```

Чтобы база заработала-надо прописать в консоль

```
docker-compose up
```

2 Создание таблиц

Таблицы будем создавать, используя механизм миграций. Так, через файл.env передадим директорию с файлами миграции. Чтобы была возможность мигрировать до конкретной

версии добавим переменную MIG_VERSION в .env , в также bash скрипт init.sh, который будет выбирать и запускать нужные миграции.(Но сначала создаст роль, от которой эти миграции будут запускаться) Чтобы это сделать пропишем в volumes в docker-compose.yaml:

```
    volumes:
      - ./migrations:/docker-entrypoint-initdb.d/migrations
      - ./.env:/docker-entrypoint-initdb.d/env
      - ./init.sh:/docker-entrypoint-initdb.d/init.sh
```

В .env:

```
    DB_NAME=postgres
    DB_USER=postgres
    DB_PASSWORD=postgres
    MIG_VERSION=0.0.0
```

В init.sh:

```
    #!/bin/bash
db_name=$(grep -E 'DB_NAME' /docker-entrypoint-initdb.d/env | awk 'BEGIN { FS = "=" } ; {print $2}')
db_user=$(grep -E 'DB_USER' /docker-entrypoint-initdb.d/env | awk 'BEGIN { FS = "=" } ; {print $2}')
psql -U $db_user -d $db_name -c "CREATE ROLE admin WITH LOGIN CREATEDB;"

last_version="0.0.0"
mig_path="/docker-entrypoint-initdb.d/migrations"
num=$(grep -E 'MIG_VERSION' /docker-entrypoint-initdb.d/env | awk 'BEGIN { FS = "=" } ; {print $2}')
if [[ "$num" == "" ]]
then
    psql -U admin -d $db_name -f "$mig_path/$last_version.sql"
else
    array=$(ls $mig_path | sort )
    for file in "$array"
    do
    psql -U admin -d $db_name -f "$mig_path/$file"
    if [[ "$file" == "$num.sql" ]]
    then
        break
    fi

    done;
fi
```

Теперь в файлах директории migrations можно написать первый файл миграции 0.0.0.sql . Вот его часть:

```
DROP TYPE IF EXISTS ParticipantStatus;
CREATE TYPE ParticipantStatus AS ENUM ('participant', 'viewer');

CREATE TABLE IF NOT EXISTS Part (
    partID      integer PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    date_start   timestamptz NOT NULL,
    date_finish  timestamptz NOT NULL
);
```

3 Генерация данных

По схожей выше схеме реализуем генерацию тестовых данных. Через переменную в .env передадим требуемое количество кортежей в основных отношениях:

```
   COUNT=1000000
```

Создадим директорию generator и файл для генерации 0.0.0.sql:

```
...
INSERT INTO Category(partId,dancer_type, competition_program,dance_count,
    age_category)
SELECT
    FLOOR(RANDOM() * FLOOR(:'count'/3)+1 )::INTEGER,
  (enum_range(NULL::DancerType))[random_choice(array[1,2])],
  (enum_range(NULL::CompetitionsProgram))[random_choice(array[1,2,3])],
    random_choice(array[2, 3, 4, 5, 6,8,10]),
    (enum_range(NULL::AgeCategory))[random_choice(array[1,2,3,4,5,6,7,8,9])]
FROM GENERATE_SERIES(1, FLOOR(:'count'/2)::INTEGER);
...
```

Допишем в docker-compose.yaml добавленную директорию:

```
    volumes:
      - ./generator:/docker-entrypoint-initdb.d/generator
```

Запуском файлов генерации будет заниматься написанный ранее init.sh скрипт. Допишем в него следующие строки:

```
gen_path="/docker-entrypoint-initdb.d/generator"
count=$(grep -E 'COUNT' /docker-entrypoint-initdb.d/env | awk 'BEGIN { FS =
    "=" } ; {print $2}')

if [[ "$num" == "" ]]
then
    psql  -U admin -d $db_name -v count=$count -f "$gen_path/$last_version.sql
    "
else
    array=$(ls $gen_path | sort )
    for file in "$array"
    do
    psql  -U admin -d $db_name -v count=$count -f "$gen_path/$file"
    if [[ "$file" == "$num.sql" ]]
    then
        break
    fi

    done;
fi
```

4 Добавление ролей

По условию требуется это сделать в bash скрипте. Создадим скрипт roles.sh и добавим его в в docker-compose.yaml:

```
    volumes:
    - ./roles.sh:/docker-entrypoint-initdb.d/roles.sh
```

В .env добавим переменную-массив, через которую будут передаваться имена ролей для групповой роли:

```
   NAMES=Liza,Morgenshtern,Stich
```

В самом скрипте напишем создание ролей:

```
    #!/bin/bash
db_name=$(grep -E 'DB_NAME' /docker-entrypoint-initdb.d/env | awk 'BEGIN { FS
    = "=" } ; {print $2}')
```

```
db_user=$(grep -E 'DB_USER' /docker-entrypoint-initdb.d/env | awk 'BEGIN { FS = "=" } ; {print $2}')

psql -U $db_user -d $db_name -c "CREATE ROLE reader WITH LOGIN; GRANT SELECT ON ALL TABLES IN SCHEMA public TO reader;"
psql -U $db_user -d $db_name -c "CREATE ROLE writer WITH LOGIN; GRANT SELECT, UPDATE, INSERT ON ALL TABLES IN SCHEMA public TO writer;"
psql -U $db_user -d $db_name -c "CREATE ROLE analytic WITH LOGIN; GRANT SELECT ON ProtocolFinal TO analytic;"


psql -U $db_user -d $db_name -c "CREATE ROLE group_role; GRANT CONNECT ON DATABASE $db_name TO group_role;"

names_str=$(grep -E 'NAMES' /docker-entrypoint-initdb.d/env | awk 'BEGIN { FS = "=" } ; {print $2}')
names=($(echo $names_str | tr ',' '\n'))

for i in "${names[@]}"
do
psql -U $db_user -d $db_name -c "CREATE ROLE $i; GRANT group_role TO $i;"
done
```