Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Информационных технологий и программирования

Лабораторная работа №2
Начало работы с БД

Выполнил: студент группы
М3203
Костыгов Андрей Константинович
Проверила:
Шевчик Софья Владимировна

Санкт-Петербург
2024 г.

Создание докер-контейнера

Напишем простой скрипт в файле docker-compose.yml:

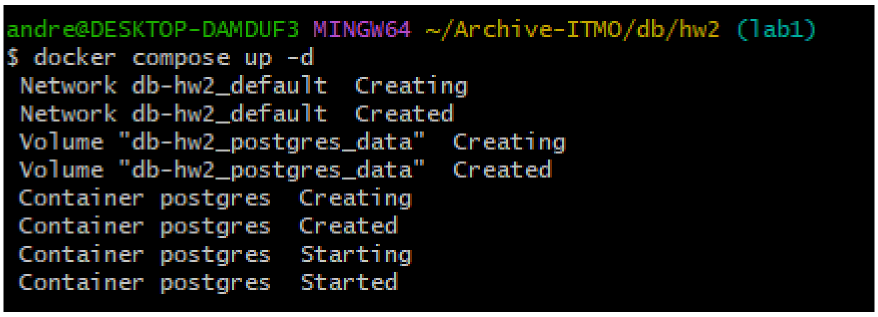
```
version: "3.9"

volumes:
  postgres_data:
  scripts:

name: "db-hw2"

services:
  postgres:
    container_name: postgres
    image: postgres:16.2
    hostname: postgres
    restart: unless-stopped
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./scripts:/docker-entrypoint-initdb.d
    ports:
      - "5432:5432"
    environment:
      - POSTGRES_USERS=user1,user2,user3
      - POSTGRES_DB=dbhw2
      - POSTGRES_PORT=5432
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_USER=postgres
```

Запустим его с помощью команды docker-compose up -d. В терминале появится сообщение, что все успешно собралось:



```
andre@DESKTOP-DAMDUF3 MINGW64 ~/Archive-ITMO/db/hw2 (lab1)
$ docker compose up -d
 Network db-hw2_default  Creating
 Network db-hw2_default  Created
 Volume "db-hw2_postgres_data"  Creating
 Volume "db-hw2_postgres_data"  Created
 Container postgres  Creating
 Container postgres  Created
 Container postgres  Starting
 Container postgres  Started
```

Создание таблиц

Создадим таблицы с помощью SQL-запросов и bash-программ. Для этого для каждой таблицы напишем скрипт, создающий свое отношение. С помощью команды psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c "$SQL" –quiet создадим таблицу, где $user, $db, $port – передаются в программу аргументами при запуске, $SQL – запрос для создания отношения (таблицы). Запросы приведены в приложении 1.

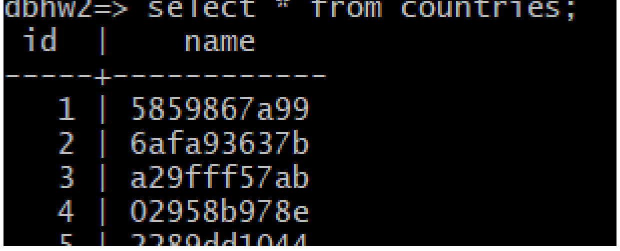
Для того, чтобы не запускать каждый запрос по отдельности, создадим init.sh файл, в котором последовательно будут создаваться отношения. Этот файл мы сможем запустить командой winpty docker exec -it postgres docker-entrypoint-initdb.d/init.sh dbhw2 postgres 5432.

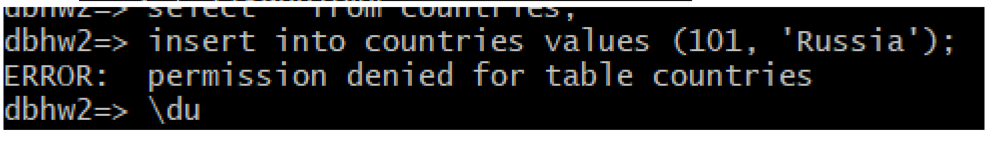
Заполнение таблиц

Для заполнения аналогично создадим bash-скрипты с SQL-запросами. Изначально, в таблицах, где число кортежей не превышает n = 10000, можно было воспользоваться циклом в bash-коде с n итерациями по INSERT в каждом. Однако при больших n, команда psql начинает суммарно занимать много времени. Поэтому я стал писать более сложные SQL-запросы, содержащие в себе многократную генерацию данных и вставки. Таким образом, за одну транзакцию можно создать всю таблицу, что будет гораздо быстрее. Подобные запросы я написал в скриптах, заполняющих порядка миллиона кортежей. Скрипты находятся в приложении 2.

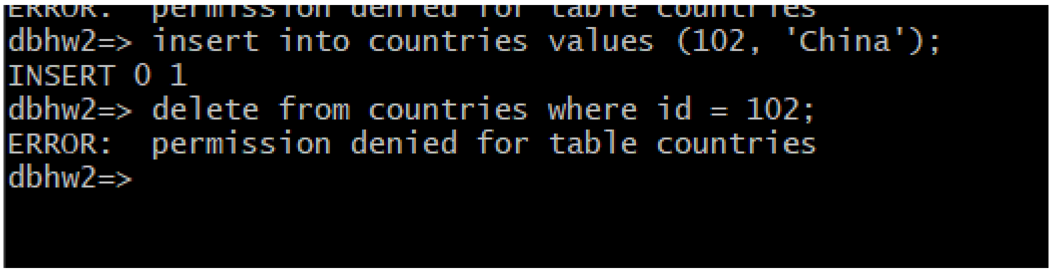
Создание ролей

Создадим скрипты для создания ролей. Reader может только читать данные, не изменяя их. Убедимся в этом, зайдя в базу под ролью reader. Мы можем спокойно выполнить чтение из таблицы, однако не можем добавить новые данные:


```
dbhw2=> select * from countries;
 id  |    name
-----+------------
   1 | 5859867a99
   2 | 6afa93637b
   3 | a29fff57ab
   4 | 02958b978e
```

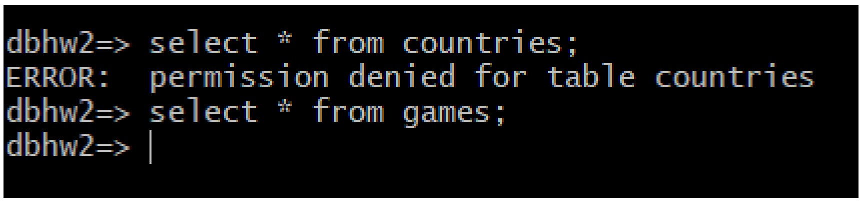

```
dbhw2=> insert into countries values (101, 'Russia');
ERROR:  permission denied for table countries
dbhw2=> \du
```

В роли writer мы можем помимо чтения править данные, добавляя и обновляя, но не удаляя.



```
ERROR:  permission denied for table countries
dbhw2=> insert into countries values (102, 'China');
INSERT 0 1
dbhw2=> delete from countries where id = 102;
ERROR:  permission denied for table countries
dbhw2=>
```

Роль аналитик: дает доступ к чтению определенной таблице (у нас Games), но не позволяет смотреть, изменять данные в других таблицах:



```
dbhw2=> select * from countries;
ERROR:  permission denied for table countries
dbhw2=> select * from games;
dbhw2=> |
```

Групповая роль: зайти под которой в базу данных нельзя. Она имеет все возможности манипуляций над базой данных: запись новых данных, удаление и чтение.

Приложение:

1. Создание отношения Referees:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
SQL="CREATE TABLE IF NOT EXISTS Referees (
    passport bigint PRIMARY KEY,
    firstname varchar(255),
    lastname varchar(255),
    number int,
    age int
);"
psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c
"$SQL" –quiet
```

2. Создание отношения Countries:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
SQL="CREATE TABLE IF NOT EXISTS Countries (
    id int UNIQUE,
    name varchar(255) UNIQUE,
    PRIMARY KEY (id, name)
);"
psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c "$SQL" –quiet
```

3. Создание отношения Cities:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
SQL="CREATE TABLE IF NOT EXISTS Cities (
    id INT UNIQUE,
    name varchar(255) UNIQUE,
    countryId int REFERENCES Countries(id),
    PRIMARY KEY (name, countryId)
);"
psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c "$SQL" –quiet
```

4. Создание отношения Stadiums:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
SQL="CREATE TABLE IF NOT EXISTS Stadiums (
    id INT UNIQUE,
    name varchar(255),
    adress varchar(255),
    PRIMARY KEY (id, adress)
);"
psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c "$SQL" –quiet
```

5. Создание отношения Clubs:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
SQL="CREATE TABLE IF NOT EXISTS Clubs (
    id INT UNIQUE,
    name varchar(255),
    cityId int references Cities(id),
    stadiumId int references Stadiums(id),
    PRIMARY KEY (id, name)
);"
psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c "$SQL" –quiet
```

6. Создание отношения Trainers:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
SQL="CREATE TABLE IF NOT EXISTS Trainers (
    passport bigint UNIQUE PRIMARY KEY,
    firstname varchar(255),
    lastname varchar(255),
    clubId int references Clubs(id),
    countryId int references Countries(id),
    age int
);"
psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c "$SQL" –quiet
```

7. Создание enum-типа PlayerPosition:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
SQL="create type PlayerPosition as enum('Goalkeaper', 'Defender', 'Attacker');"
psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c "$SQL" –quiet
```

Создание отношения Players:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
SQL="CREATE TABLE IF NOT EXISTS Players (
    passport bigint UNIQUE PRIMARY KEY,
    firstname varchar(255),
    lastname varchar(255),
        clubId int references Clubs(id),
        number int,
        countryId int references Countries(id),
        age int,
        playerposition PlayerPosition
    );"
    psql -h localhost -U "$2" -d "$1" -p "$port" -v ON_ERROR_STOP=1 -c
"$SQL"
```

Создание отношения Games:

```
    #!/bin/bash
    db="$1"
    user="$2"
    port="$3"
    SQL="CREATE TABLE IF NOT EXISTS Games (
        id SERIAL unique,
        homeClubId int references Clubs(id),
        guestClubId int references Clubs(id),
        homeTeamScore int,
        guestTeamScore int,
        dataTime timestamp,
        stadiumId int references Stadiums(id),
        PRIMARY KEY (id, dataTime, stadiumId)
    );"
    psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c
"$SQL" –quiet
```

Создание отношения Games_Referees:

```
    #!/bin/bash
    db="$1"
    user="$2"
    port="$3"
    SQL="CREATE TABLE IF NOT EXISTS Games_Referees (
        gameId int references Games(id),
        refereePassport bigint references Referees(passport),
        PRIMARY KEY (gameId, refereePassport)
    );"
    psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c
"$SQL" –quiet
```

Создание отношения Goals:

```
    #!/bin/bash
```

```
db="$1"
user="$2"
port="$3"
SQL="CREATE TABLE IF NOT EXISTS Goals (
    time time,
    gameId int references Games(id),
    scorerPassport bigint references Players(passport),
    assistentPassport bigint references Players(passport),
    PRIMARY KEY (time, gameId)
);"
psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c
"$SQL" –quiet
```

Создание отношения Penalties:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
SQL="CREATE TABLE IF NOT EXISTS Penalties (
    gameTime time,
    gameId int references Games(id),
    playerPassport bigint references Players(passport),
    penaltyTime time,
    PRIMARY KEY (gameTime, gameId, playerPassport)
);"
psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c
"$SQL" –quiet
```

Приложение 2

1. Заполнение отношения Referees:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
password="$4"

SQL="SELECT COUNT(*) FROM referees;"
count=$(PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet)
echo $count
```

```
count=$(echo $count | awk '{print $4}')
if [[ "$count" -gt "1" ]]; then
   exit 1
fi

for i in {1..1000}; do
   number=$((1 + $RANDOM % 100))
   age=$((1 + $RANDOM % 100))
   firstname=$(echo $RANDOM | md5sum | head -c 10)
   lastname=$(echo $RANDOM | md5sum | head -c 10)
   passport=$i
   SQL="INSERT INTO referees (passport, firstname, lastname, number, age)
VALUES ($passport, '$firstname', '$lastname', $number, $age);"
   PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet
done
```

2. Заполнение отношения Countries:

```
#!/bin/bash


db="$1"
user="$2"
port="$3"
password="$4"
SQL="SELECT COUNT(*) FROM Countries;"
count=$(PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet)
count=$(echo $count | awk '{print $4}')
if [[ "$count" -gt "1" ]]; then
   exit 1
fi


echo "5"
for i in {1..100}; do
   #echo "num"
   id=$i
   name=$(echo $RANDOM | md5sum | head -c 10)
   SQL="INSERT INTO countries (id, name)
   VALUES ($id, '$name');"
   PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet
done
```

3. Заполнение отношения Cities:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
password="$4"

SQL="SELECT COUNT(*) FROM Cities;"
count=$(PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet)
count=$(echo $count | awk '{print $4}')
if [[ "$count" -gt "1" ]]; then
   exit 1
fi
for i in {1..100}; do
   #echo "num"
   id=$i
   name=$(echo $RANDOM | md5sum | head -c 10)
   countryId=$((1 + $RANDOM % 100))
   SQL="INSERT INTO cities (id, name, countryId)
VALUES ($id, '$name', $countryId);"
   PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet
   #echo "$i"
done
```

4. Заполнение отношения Stadiums:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
password="$4"

SQL="SELECT COUNT(*) FROM Stadiums;"
count=$(PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet)
```

```
count=$(echo $count | awk '{print $4}')
if [[ "$count" -gt "1" ]]; then
   exit 1
fi
for ((i = 1; i <= $5; i++)); do
   id=$i
   name=$(echo $RANDOM | md5sum | head -c 10)
   adress=$(echo $RANDOM | md5sum | head -c 10)
   SQL="INSERT INTO Stadiums (id, name, adress)
VALUES ($id, '$name', '$adress');"
   PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet
   #echo "$i"
done
```

5. Заполнение отношения Clubs:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
password="$4"

SQL="SELECT COUNT(*) FROM Clubs;"
count=$(PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet)
count=$(echo $count | awk '{print $4}')
if [[ "$count" -gt "1" ]]; then
   exit 1
fi

for ((i = 1; i <= $5; i++)); do
   id=$i
   name=$(echo $RANDOM | md5sum | head -c 10)
   cityId=$((1 + $RANDOM % 100))
   stadiumId=$i
   SQL="INSERT INTO Clubs (id, name, cityId, stadiumId)
VALUES ($id, '$name', $cityId, $stadiumId);"
   PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet
done
```

6. Заполнение отношения Trainers:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
password="$4"

SQL="SELECT COUNT(*) FROM Trainers;"
count=$(PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet)
count=$(echo $count | awk '{print $4}')
if [[ "$count" -gt "1" ]]; then
   exit 1
fi

for ((i = 1; i <= $5; i++)); do
   clubId=$i
   age=$((1 + $RANDOM % 100))
   firstname=$(echo $RANDOM | md5sum | head -c 10)
   lastname=$(echo $RANDOM | md5sum | head -c 10)
   passport=$((i + 1000))
   countryId=${RANDOM:0:2}
   SQL="INSERT INTO trainers (passport, firstname, lastname, clubId, countryId,
age)
    VALUES ($passport, '$firstname', '$lastname', $clubId, $countryId, $age);"
   PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet
done
```

7. Заполнение отношения Players:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
password="$4"

echo "$db"
echo "$user"
echo "$port"
echo "$password"
echo "$5"
echo "$6"

SQL="SELECT COUNT(*) FROM Players;"
count=$(PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet)
count=$(echo $count | awk '{print $4}')
if [[ "$count" -gt "1" ]]; then
   exit 1
fi

   SQL="
   do \$\$
   DECLARE
   testCount integer := $6;
   DECLARE PlayerPosit PlayerPosition;
   DECLARE Club INT;
   DECLARE Country INT;
   DECLARE Number INT;
   DECLARE Age INT;
   DECLARE name VARCHAR(255);
   DECLARE surname VARCHAR(255);
   begin
      WHILE TestCount > 0 loop
         EXECUTE 'SELECT FLOOR(RANDOM() * ($5)) + 1' INTO Club;
         EXECUTE 'SELECT FLOOR(RANDOM() * (100)) + 1' INTO Country;
         EXECUTE 'SELECT FLOOR(RANDOM() * (100)) + 1' INTO Number;
         EXECUTE 'SELECT FLOOR(RANDOM() * (60)) + 1' INTO Age;
         EXECUTE 'SELECT LEFT(MD5(RANDOM()::text), 8)' INTO name;
         EXECUTE 'SELECT LEFT(MD5(RANDOM()::text), 8)' INTO surname;
         IF TestCount < 5000 THEN
            PlayerPosit := 'Goalkeaper';
         ELSIF TestCount > 500000 THEN
            PlayerPosit := 'Attacker';
         ELSE
            PlayerPosit := 'Defender';
         END IF;
```

```
          INSERT INTO players (passport, firstname, lastname, clubId, number,
countryId, age, playerposition)
          VALUES (TestCount, name, surname, Club, Number, Country, Age,
PlayerPosit);
          TestCount := TestCount - 1;
       end loop;
   end;
   \$\$;"
   PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet
#done
```

8. Заполнение отношения Games:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
password="$4"

SQL="SELECT COUNT(*) FROM Games;"
count=$(PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet)
count=$(echo $count | awk '{print $4}')
if [[ "$count" -gt "1" ]]; then
   exit 1
fi

SQL="
   do \$\$
   DECLARE
   testCount integer := $6;
   DECLARE homeTeamScore1 INT;
   DECLARE guestTeamScore1 INT;
   DECLARE homeClubId1 INT;
   DECLARE guestClubId1 INT;
   DECLARE stadiumId1 INT;
   DECLARE time TIMESTAMP;
   begin
      WHILE TestCount > 0 loop
```

```
          EXECUTE 'SELECT FLOOR(RANDOM() * (5)) + 1' INTO
homeTeamScore1;
          EXECUTE 'SELECT FLOOR(RANDOM() * (5)) + 1' INTO
guestTeamScore1;
          EXECUTE 'SELECT FLOOR(RANDOM() * ($5)) + 1' INTO
homeClubId1;
          EXECUTE 'SELECT FLOOR(RANDOM() * ($5)) + 1' INTO
guestClubId1;
          EXECUTE 'SELECT FLOOR(RANDOM() * ($5)) + 1' INTO stadiumId1;
          EXECUTE 'select timestamp "1000-01-10 20:00:00" + random() *
(timestamp "8014-01-20 20:00:00" - timestamp "1014-01-10 10:00:00")' INTO
time;

          INSERT INTO Games (homeTeamScore, guestTeamScore, homeClubId,
guestClubId, stadiumId, dataTime)
          VALUES (homeTeamScore1, guestTeamScore1, homeClubId1,
guestClubId1, stadiumId1, time);
          TestCount := TestCount - 1;
       end loop;
   end;
   \$\$;"

PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet
```

9. Заполнение отношения Games_Referees:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
password="$4"

SQL="SELECT COUNT(*) FROM Games_Referees;"
count=$(PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet)
count=$(echo $count | awk '{print $4}')
if [[ "$count" -gt "1" ]]; then
   exit 1
fi
```

```
SQL="
    do \$\$
    DECLARE
    testCount integer := 1000000;
    DECLARE referee1 INT;
    DECLARE referee2 INT;
    begin
        WHILE TestCount > 0 loop
            EXECUTE 'SELECT FLOOR(RANDOM() * (1000)) + 1' INTO referee1;
            EXECUTE 'SELECT FLOOR(RANDOM() * (1000)) + 1' INTO referee2;
            IF referee1 = referee2 THEN
                EXECUTE 'SELECT FLOOR(RANDOM() * (1000)) + 1' INTO
referee2;
            END IF;
            IF referee1 = referee2 THEN
                EXECUTE 'SELECT FLOOR(RANDOM() * (1000)) + 1' INTO
referee2;
            END IF;

            INSERT INTO Games_Referees (gameId, refereePassport)
            VALUES (TestCount, referee1);
            INSERT INTO Games_Referees (gameId, refereePassport)
            VALUES (TestCount, referee2);
            TestCount := TestCount - 1;
        end loop;
    end;
    \$\$;"

PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" –quiet
```

10. Заполнение таблицы Goals

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
password="$4"

SQL="SELECT COUNT(*) FROM Goals;"
```

```
count=$(PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet)
count=$(echo $count | awk '{print $4}')
if [[ "$count" -gt "1" ]]; then
exit 1
fi

SQL="
do \$\$
DECLARE
testCount integer := 3 * $6;
DECLARE assistentPassport1 INT;
DECLARE scorerPassport1 INT;
DECLARE gameId1 INT;
DECLARE time TIME;
begin
WHILE TestCount > 0 loop
EXECUTE 'SELECT FLOOR(RANDOM() * ($6)) + 1' INTO scorerPassport1;
EXECUTE 'SELECT FLOOR(RANDOM() * ($6)) + 1' INTO
assistentPassport1;
EXECUTE 'select time "00:00:00" + random() * (time "10:00:00" - time
"00:00:00")' INTO time;

INSERT INTO Goals (time, gameId, scorerPassport, assistentPassport)
VALUES (time, TestCount % $6 + 1, scorerPassport1, assistentPassport1);
TestCount := TestCount - 1;
end loop;
end;
\$\$;"

PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet
```

11.Заполнение отношения Penalties:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
password="$4"
```

```
# SQL="SELECT COUNT(*) FROM Penalties;"
# count=$(PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet)
# count=$(echo $count | awk '{print $4}')
# if [[ "$count" -gt "1" ]]; then
#    exit 1
# fi

SQL="
   do \$\$
   DECLARE
   testCount integer := 5 * $6;
   DECLARE playerPassport1 INT;
   DECLARE gameId1 INT;
   DECLARE gametime1 TIME;
   DECLARE penaltytime1 TIME;
   begin
      WHILE TestCount > 0 loop
         EXECUTE 'SELECT FLOOR(RANDOM() * ($6)) + 1' INTO
playerPassport1;
         EXECUTE 'select time "00:00:00" + random() * (time "10:00:00" - time
"00:00:00")' INTO gametime1;
         EXECUTE 'select time "00:00:00" + random() * (time "00:20:00" - time
"00:00:00")' INTO penaltytime1;

         INSERT INTO Penalties (gameTime, penaltyTime, gameId,
playerPassport)
         VALUES (gametime1, penaltytime1, TestCount % $6 + 1,
playerPassport1);
         TestCount := TestCount - 1;
      end loop;
   end;
   \$\$;"

PGPASSWORD="$password" psql -U "$user" -d "$db" -p "$port" -v
ON_ERROR_STOP=1 -c "$SQL" --quiet
```

Приложение 3

1. Создание роли Writer:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
SQL="
   CREATE ROLE writer LOGIN;
   GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA public
TO writer;
   "

psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c
"$SQL"  --quiet
```

2. Создание роли Reader:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
SQL="
   CREATE ROLE reader LOGIN;
   GRANT SELECT ON ALL TABLES IN SCHEMA public TO reader;
   "

psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c
"$SQL"  --quiet
```

3. Создание роли Analytic:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
SQL="
   CREATE ROLE analytic LOGIN;
   GRANT SELECT ON Games TO analytic;
   "

psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c
"$SQL"  --quiet
```

4. Создание роли group_role:

```
#!/bin/bash
db="$1"
user="$2"
port="$3"
users="$4"
IFS=' ' read -ra ADDR <<< "${users[@]}"

SQL="
   CREATE ROLE group_role NOLOGIN;
   GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN
SCHEMA public TO group_role;
   "

psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c
"$SQL"  --quiet


for i in "${ADDR[@]}"; do
 if [[ "$i" != "" ]]; then

   SQL="
   CREATE USER $i with password '1234';
   grant group_role to $i;
   "
   psql -h localhost -U "$user" -d "$db" -p "$port" -v ON_ERROR_STOP=1 -c
"$SQL"  --quiet

 fi
done
```