Домашнее задание к занятию "6.2. SQL"

Задача 1

Используя docker поднимите инстанс PostgreSQL (версию 12) с 2 volume, в который будут складываться данные БД и бэкапы.

Приведите получившуюся команду или docker-compose манифест.

Ответ

файл docker-compose.yaml
листинг:

```
version: '3'
services:
db:
  container_name: pg12
  image: postgres:12
  environment:
    POSTGRES_USER: alexg36
    POSTGRES_PASSWORD: 12345678
    POSTGRES_DB: start_db
  ports:
    - "5432:5432"
  volumes:
    - database_volume:/home/database/
    - backup_volume:/home/backup/

volumes:
database_volume:
backup_volume:
```

выполняем
docker-compose up -d
Вывод с терминала:

```
   ~ docker-compose up -d                    ✔
[+] Running 14/14
 ⠿ db Pulled                                27.5s
   ⠿ a603fa5e3b41 Pull complete              8.0s
   ⠿ 02d7a77348fd Pull complete              8.7s
   ⠿ 16b62ca80c8f Pull complete              8.8s
   ⠿ fbd795da1fe1 Pull complete              9.0s
   ⠿ 9c68de39d930 Pull complete              9.8s
   ⠿ 2e441a95082c Pull complete              10.0s
   ⠿ 1c97f440fe14 Pull complete              10.1s
   ⠿ 87a3f78bc5d1 Pull complete              10.2s
   ⠿ 6f5522bdba19 Pull complete              19.9s
```

⠿ 3ffbed8daf3b Pull complete 19.9s
⠿ fe084ee65e13 Pull complete 20.0s
⠿ 3b4e12d98615 Pull complete 20.1s
⠿ f6c5d03edc85 Pull complete 20.2s
[+] Running 4/4
⠿ Network alex_default Created 0.1s
⠿ Volume "alex_backup_volume" Created 0.0s
⠿ Volume "alex_database_volume" Creat... 0.0s
⠿ Container pg12 Started 0.7s
~ ✔ 29s

затем
sudo docker exec -it pg12 bash

Вывод с терминала:
~ sudo docker exec -it pg12 bash 1 ✘ 34s
[sudo] пароль для alex:
root@d35538e00313:/#

Задача 2

В БД из задачи 1:

создайте пользователя test-admin-user и БД test_db
в БД test_db создайте таблицу orders и clients (спецификация таблиц ниже)
предоставьте привилегии на все операции пользователю test-admin-user на таблицы БД test_db
создайте пользователя test-simple-user
предоставьте пользователю test-simple-user права на SELECT/INSERT/UPDATE/DELETE данных таблиц БД test_db
Таблица orders:

id (serial primary key)
наименование (string)
цена (integer)
Таблица clients:

id (serial primary key)
фамилия (string)
страна проживания (string, index)
заказ (foreign key orders)
Приведите:

итоговый список БД после выполнения пунктов выше,
описание таблиц (describe)
SQL-запрос для выдачи списка пользователей с правами над таблицами test_db
список пользователей с правами над таблицами test_db

Ответ

создадим БД test_db и выполним подключение к созданной базе

Вывод с терминала:

```
root@d35538e00313:/# createdb test_db -U alexg36
root@d35538e00313:/# psql -d test_db -U alexg36
psql (12.13 (Debian 12.13-1.pgdg110+1))
Type "help" for help.
test_db=#
```

создадим пользователя test-admin-user

Вывод с терминала:

```
test_db=# CREATE USER test_admin_user;
CREATE ROLE
test_db-#
```

в БД test_db создадим таблицы orders и clients

Вывод с терминала:

```
test_db=# CREATE TABLE orders
(
  id SERIAL PRIMARY KEY,
  наименование TEXT,
  цена INTEGER
);
CREATE TABLE
test_db=# CREATE TABLE clients
(
   id SERIAL PRIMARY KEY,
   фамилия TEXT,
   "страна проживания" TEXT,
   заказ INTEGER,
   FOREIGN KEY (заказ) REFERENCES orders(id)
);
CREATE TABLE
test_db=# CREATE INDEX country_index ON clients ("страна проживания");
CREATE INDEX
test_db=#
```

предоставим привилегии на все операции пользователю test-admin-user на таблицы БД test_db

Вывод с терминала:

```
test_db=# GRANT ALL ON TABLE orders TO test_admin_user;
GRANT
test_db=# GRANT ALL ON TABLE clients TO test_admin_user;
GRANT
test_db-#
```

создадим пользователя test-simple-user

Вывод с терминала:

```
test_db=# CREATE USER test_simple_user;
CREATE ROLE
test_db-#
```

предоставим пользователю test-simple-user права на SELECT/INSERT/UPDATE/DELETE данных таблиц БД test_db
Вывод с терминала:
test_db=# GRANT SELECT,INSERT,UPDATE,DELETE ON TABLE orders TO test_simple_user;
GRANT
test_db=# GRANT SELECT,INSERT,UPDATE,DELETE ON TABLE clients TO test_simple_user;
GRANT
test_db=#

итоговый список БД после выполнения пунктов выше:
команда:
test_db=# \l+
Вывод с терминала:
test_db=# \l+

List of databases

Name | Owner | Encoding | Collate | Ctype | Access privileges | Size | Tablespace | Description

```
-----------+---------+----------+------------+------------+--------------------+---------+------------+--------------------------------
 postgres  | alexg36 | UTF8     | en_US.utf8 | en_US.utf8 |                    | 7969 kB | pg_default | default administrative connection database
 start_db  | alexg36 | UTF8     | en_US.utf8 | en_US.utf8 |                    | 7825 kB | pg_default |
 template0 | alexg36 | UTF8     | en_US.utf8 | en_US.utf8 | =c/alexg36         +| 7825 kB | pg_default | unmodifiable empty database
           |         |          |            |            | alexg36=CTc/alexg36 |         |            |
 template1 | alexg36 | UTF8     | en_US.utf8 | en_US.utf8 | =c/alexg36         +| 7825 kB | pg_default | default template for new databases
           |         |          |            |            | alexg36=CTc/alexg36 |         |            |
 test_db   | alexg36 | UTF8     | en_US.utf8 | en_US.utf8 |                    | 8121 kB | pg_default |
(5 rows)
```

test_db=#
описание таблиц (describe)
команда test_db=# \d+ orders
Вывод с терминала:
test_db=# \d+ orders

Table "public.orders"

Column | Type | Collation | Nullable | Default | Storage | Stats target | Description

```
--------------+---------+-----------+----------+-----------------------------------+----------+--------------+-------------
 id           | integer |           | not null | nextval('orders_id_seq'::regclass) | plain    |              |
 наименование | text    |           |          |                                   | extended |              |
 цена         | integer |           |          |                                   | plain    |              |
```

Indexes:
"orders_pkey" PRIMARY KEY, btree (id)
Referenced by:
TABLE "clients" CONSTRAINT "clients_заказ_fkey" FOREIGN KEY ("заказ") REFERENCES orders(id)
Access method: heap

test_db=#

команда test_db=# \d+ clients
Вывод с терминала:
test_db=# \d+ clients

Table "public.clients"

Column | Type | Collation | Nullable | Default | Storage | Stats target | Description

id | integer | | not null | nextval('clients_id_seq'::regclass) | plain | |
фамилия | text | | | | extended | |
страна проживания | text | | | | extended | |
заказ | integer | | | | plain | |

Indexes:
"clients_pkey" PRIMARY KEY, btree (id)
"country_index" btree ("страна проживания")
Foreign-key constraints:
"clients_заказ_fkey" FOREIGN KEY ("заказ") REFERENCES orders(id)
Access method: heap

test_db=#

SQL-запрос для выдачи списка пользователей с правами над таблицами test_db

test_db=# SELECT grantee, table_catalog, table_name, privilege_type FROM information_schema.table_privileges WHERE table_name IN ('orders','clients');

список пользователей с правами над таблицами test_db
Вывод с терминала:
test_db=# SELECT grantee, table_catalog, table_name, privilege_type FROM information_schema.table_privileges WHERE table_name IN ('orders','clients');

grantee | table_catalog | table_name | privilege_type

alexg36 | test_db | orders | INSERT
alexg36 | test_db | orders | SELECT
alexg36 | test_db | orders | UPDATE
alexg36 | test_db | orders | DELETE
alexg36 | test_db | orders | TRUNCATE
alexg36 | test_db | orders | REFERENCES
alexg36 | test_db | orders | TRIGGER

test_admin_user | test_db | orders | INSERT
test_admin_user | test_db | orders | SELECT
test_admin_user | test_db | orders | UPDATE
test_admin_user | test_db | orders | DELETE
test_admin_user | test_db | orders | TRUNCATE
test_admin_user | test_db | orders | REFERENCES
test_admin_user | test_db | orders | TRIGGER
test_simple_user | test_db | orders | INSERT
test_simple_user | test_db | orders | SELECT
test_simple_user | test_db | orders | UPDATE
test_simple_user | test_db | orders | DELETE
alexg36 | test_db | clients | INSERT
alexg36 | test_db | clients | SELECT
alexg36 | test_db | clients | UPDATE
alexg36 | test_db | clients | DELETE
alexg36 | test_db | clients | TRUNCATE
alexg36 | test_db | clients | REFERENCES
alexg36 | test_db | clients | TRIGGER
test_admin_user | test_db | clients | INSERT
test_admin_user | test_db | clients | SELECT
test_admin_user | test_db | clients | UPDATE
test_admin_user | test_db | clients | DELETE
test_admin_user | test_db | clients | TRUNCATE
test_admin_user | test_db | clients | REFERENCES
test_admin_user | test_db | clients | TRIGGER
test_simple_user | test_db | clients | INSERT
test_simple_user | test_db | clients | SELECT
test_simple_user | test_db | clients | UPDATE
test_simple_user | test_db | clients | DELETE
(36 rows)

test_db=#

Задача 3

Используя SQL синтаксис - наполните таблицы следующими тестовыми данными:

Таблица orders

Наименование	цена
Шоколад	10
Принтер	3000
Книга	500
Монитор	7000
Гитара	4000

Таблица clients

ФИО	Страна проживания
Иванов Иван Иванович	USA
Петров Петр Петрович	Canada
Иоганн Себастьян Бах	Japan
Ронни Джеймс Дио	Russia
Ritchie Blackmore	Russia

Используя SQL синтаксис:

вычислите количество записей для каждой таблицы
приведите в ответе:
запросы
результаты их выполнения.

Ответ

наполним таблицы требуемыми тестовыми данными
Вывод с терминала:

```
test_db=# INSERT INTO orders VALUES (1, 'Шоколад', 10), (2, 'Принтер', 3000), (3, 'Книга', 500), (4, 'Монитор', 7000), (5, 'Гитара', 4000);
INSERT 0 5
test_db=# INSERT INTO clients VALUES (1, 'Иванов Иван Иванович', 'USA'), (2, 'Петров Петр Петрович', 'Canada'), (3, 'Иоганн Себастьян Бах', 'Japan'), (4, 'Ронни Джеймс Дио', 'Russia'), (5, 'Ritchie Blackmore', 'Russia');
INSERT 0 5
test_db=#
```

SQL-запросы для вычисления количества записей в таблицах:

```
SELECT COUNT (*) FROM orders;
SELECT COUNT (*) FROM clients;
```

Вывод с терминала:

```
test_db=# SELECT COUNT (*) FROM orders;
 count
-------
     5
(1 row)

test_db=# SELECT COUNT (*) FROM clients;
 count
-------
     5
(1 row)

test_db=#
```

Задача 4

Часть пользователей из таблицы clients решили оформить заказы из таблицы orders.

Используя foreign keys свяжите записи из таблиц, согласно таблице:

ФИО	Заказ
Иванов Иван Иванович	Книга
Петров Петр Петрович	Монитор
Иоганн Себастьян Бах	Гитара

Приведите SQL-запросы для выполнения данных операций.

Приведите SQL-запрос для выдачи всех пользователей, которые совершили заказ, а также вывод данного запроса.

Подсказка - используйте директиву UPDATE.

Ответ

свяжем записи из таблиц следующими запросами

Вывод с терминала:

```
test_db=# UPDATE clients SET заказ=(select id from orders where наименование='Книга') WHERE фамилия='Иванов Иван Иванович';
UPDATE 1
test_db=# UPDATE clients SET заказ=(select id from orders where наименование='Монитор') WHERE фамилия='Петров Петр Петрович';
UPDATE 1
test_db=# UPDATE clients SET заказ=(select id from orders where наименование='Гитара') WHERE фамилия='Иоганн Себастьян Бах';
UPDATE 1
test_db=#
```

с помощью запроса SELECT* FROM clients WHERE заказ IS NOT NULL; выведем пользователей, которые совершили заказ

Вывод с терминала:

```
test_db=# SELECT* FROM clients WHERE заказ IS NOT NULL;
 id |      фамилия        | страна проживания | заказ
----+---------------------+-------------------+-------
  1 | Иванов Иван Иванович | USA               |     3
  2 | Петров Петр Петрович | Canada            |     4
  3 | Иоганн Себастьян Бах | Japan             |     5
(3 rows)

test_db=#
```

Задача 5

Получите полную информацию по выполнению запроса выдачи всех пользователей из задачи 4 (используя директиву EXPLAIN).

Приведите получившийся результат и объясните что значат полученные значения.

Ответ

Вывод с терминала:

```
test_db=# EXPLAIN SELECT* FROM clients WHERE заказ IS NOT NULL;
                        QUERY PLAN
-------------------------------------------------------------
 Seq Scan on clients  (cost=0.00..18.10 rows=806 width=72)
   Filter: ("заказ" IS NOT NULL)
(2 rows)

test_db=#
```

Чтение данных из таблицы clients происходит с использованием метода Seq Scan.
0.00 — ожидаемые затраты на получение первой строки.
18.10 — ожидаемые затраты на получение всех строк.
rows - ожидаемое число строк, которое должен вывести этот узел плана.
width - ожидаемый средний размер строк, выводимых этим узлом плана (в байтах).
Каждая запись сравнивается с условием "заказ" IS NOT NULL.
Если условие выполняется, запись вводится в результат.

применим директиву ANALYZE:
Вывод с терминала:

```
test_db=# EXPLAIN (ANALYZE) SELECT* FROM clients WHERE заказ IS NOT NULL;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------------------
 Seq Scan on clients  (cost=0.00..18.10 rows=806 width=72) (actual time=0.028..0.032
rows=3 loops=1)
   Filter: ("заказ" IS NOT NULL)
   Rows Removed by Filter: 2
 Planning Time: 0.117 ms
 Execution Time: 0.063 ms
(5 rows)

test_db=#
```

Здесь уже видны реальные затраты на обработку первой и всех строк,
количество выведенных строк (3), удовлетворяющих фильтру "заказ" IS NOT NULL,
количество проходов (1), количество строк, которые были удалены из запроса по фильтру (2),
планируемое и затраченное время, а также общее количество строк, по которым производилась выборка.

Задача 6

Создайте бэкап БД test_db и поместите его в volume, предназначенный для бэкапов (см. Задачу 1).

Остановите контейнер с PostgreSQL (но не удаляйте volumes).

Поднимите новый пустой контейнер с PostgreSQL.

Восстановите БД test_db в новом контейнере.

Приведите список операций, который вы применяли для бэкапа данных и восстановления.

Ответ

Создайте бэкап БД test_db и поместите его в volume, предназначенный для бэкапов
Вывод с терминала:

test_db=# pg_dump -U alexg36 test_db > /home/backup/test_db.backup
test_db-#

Остановим контейнер с PostgreSQL (pg12)
Вывод с терминала:

```
~ docker stop pg12
127 ✘ 1h 23m 4s
pg12
~ docker ps -a
✔
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
30dcdc5dc442 postgres:12 "docker-entrypoint.s…" About an hour ago Exited (0) 25
seconds ago pg12
~ docker ps
✔
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
~
```

Поднимем новый пустой контейнер с PostgreSQL
Вывод с терминала:

```
docker run --name pg12_new -e POSTGRES_PASSWORD=12345678 -d postgres:12
✔
02f330d77c243753be8dcae0d3daaf1a8058a255883969a4a8527fe8506b6ac2
~
```

Восстановите БД test_db в новом контейнере. Для этого копируем файл дампа из контейнера pg12 в контейнер pg12_new
Вывод с терминала:

docker cp pg12:/home/backup/test_db.backup backup/ && docker cp backup/test_db.backup pg12_new:/home/

восстановим базу:
Вывод с терминала:
root@d35538e00313:/# psql -U postgres -d test_db -f /home/test_db.backup