

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №8

по дисциплине «Операционная система Linux»

Создание дампа БД и восстановление

Студент

Посаднев В.В.

Группа АС-18

Руководитель

Кургасов В.В.

Липецк 2020 г.

Цель работы

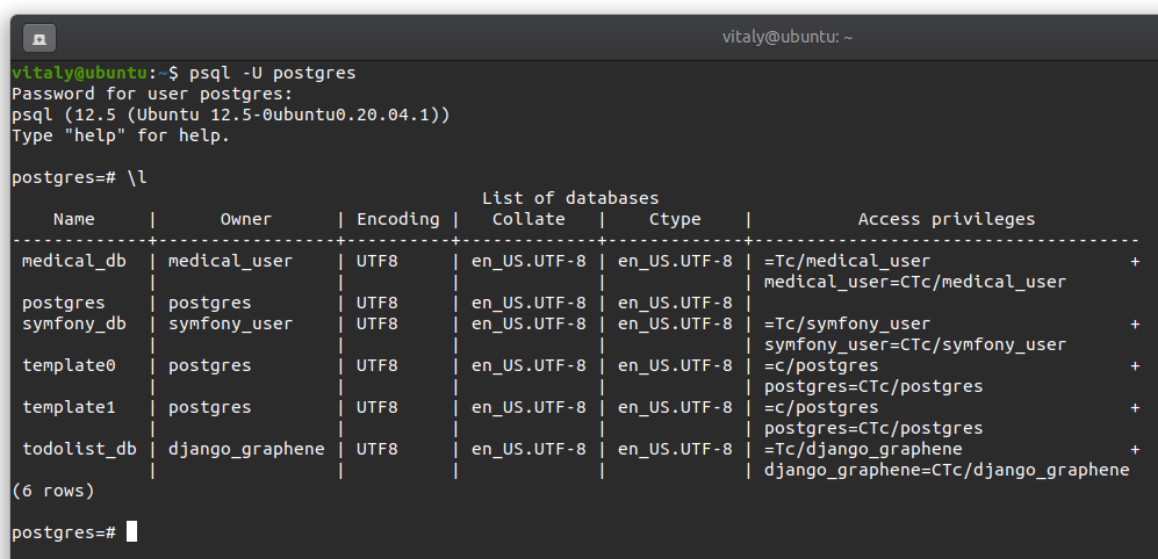
Изучить работу со средствами создания резервных копий и восстановления БД в выбранной СУБД.

Задание кафедры

Создать в выбранной СУБД (PostgreSQL) базу данных, добавить в неё таблицы, заполнить их данными. Создать резервную копию БД, удалить текущую базу данных и восстановить резервную копию в новую базу данных.

Ход работы

Для выполнения данной лабораторной работы будет использоваться СУБД PostgreSQL. Для просмотра списка уже существующих баз данных в PostgreSQL необходимо подключиться к консоли PostgreSQL, это можно сделать с помощью команды *psql -U postgres*, и после выполнения команды *\l* будет выведен список существующих баз данных. Пример выполнения данных команд приведен на рисунке 1.



```
vitaly@ubuntu: ~  
vitaly@ubuntu:~$ psql -U postgres  
Password for user postgres:  
psql (12.5 (Ubuntu 12.5-0ubuntu0.20.04.1))  
Type "help" for help.  
  
postgres=# \l  
  
          List of databases  
+-----+-----+-----+-----+-----+-----+  
 Name      | Owner      | Encoding | Collate | Ctype  | Access privileges  
+-----+-----+-----+-----+-----+-----+  
 medical_db | medical_user | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/medical_user  
           |             |          |             |             | medical_user=CTc/medical_user +  
 postgres  | postgres    | UTF8     | en_US.UTF-8 | en_US.UTF-8 |  
 symfony_db | symfony_user | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/symfony_user  
           |             |          |             |             | symfony_user=CTc/symfony_user +  
 template0 | postgres    | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres  
           |             |          |             |             | postgres=CTc/postgres +  
 template1 | postgres    | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres  
           |             |          |             |             | postgres=CTc/postgres +  
 todolist_db | django_graphene | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/django_graphene  
           |             |          |             |             | django_graphene=CTc/django_graphene +  
(6 rows)  
  
postgres=#
```

Рисунок 1 – Подключение к PostgreSQL и вывод списка существующих БД

Следующим шагом необходимо создать базу данных с которой мы будем работать, для этого необходимо выполнить следующую команду: *create database online_shop_db*; в данной команде *online_shop_db* – название создаваемой базы данных. После успешного создания БД подключимся к ней с помощью команды *\c online_shop_db* и посмотрим на таблицы в ней с помощью команды */dt*. Как можно увидеть таблицы в данной базе данных отсутствуют. Пример выполнения данных действий приведено на рисунке 2.

```

vitaly@ubuntu: ~
CREATE DATABASE
postgres=# \l

```

Name	Owner	Encoding	Collate	Ctype	Access privileges
medical_db	medical_user	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/medical_user medical_user=CTc/medical_user
online_shop_db	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
symfony_db	symfony_user	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/symfony_user symfony_user=CTc/symfony_user
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres postgres=CTc/postgres
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres postgres=CTc/postgres
todoist_db	django_graphene	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/django_graphene django_graphene=CTc/django_graphene

```

(7 rows)

postgres=# \c online_shop_db
You are now connected to database "online_shop_db" as user "postgres".
online_shop_db=# \dt
Did not find any relations.
online_shop_db=#

```

Рисунок 2 – Создание новой базы данных и подключение к ней

Теперь создадим таблицы в нашей базе данных. Данная БД будет включать следующие таблицы: customer (клиент), product (товар), product_photo (фотография товара), cart (корзина) и cart_product (товар в корзине).

Таблица клиент будет иметь следующую структуру:

```

create table customer(
customer_id serial primary key,
first_name varchar(255) not null,
last_name varchar(255) not null,
phone varchar(30) not null,
email varchar(255) not null);

```

Таблица товара будет иметь следующую структуру:

```

create table product(
product_id serial primary key,
name varchar(255) not null,
description text,
price money not null);

```

Таблица фотографий товара будет иметь следующую структуру:

```

create table product_photo(

```

product_photo_id serial primary key,
photo_url varchar(255) not null,
product_id integer references product(product_id) not null);

Таблица корзины будет иметь следующую структуру:

```
create table cart(
cart_id serial primary key,
customer_id integer references customer(customer_id) not null);
```

Таблица товара в корзине будет иметь следующую структуру:

```
create table cart_product(
cart_product_id serial primary key,
cart_id integer references cart(cart_id) not null,
product_id integer references product(product_id) not null);
```

Пример создания и просмотра таблиц с информацией о таблицы представлены на рисунке 3 для таблицы клиента, рисунке 4 для таблицы товара, рисунке 5 для таблицы фотографий товара, рисунке 6 для таблицы корзины, рисунке 7 для таблицы товара в корзине.

```
vitaly@ubuntu: ~
online_shop_db=# create table customer(
online_shop_db=# customer_id serial primary key,
online_shop_db=# first_name varchar(255) not null,
online_shop_db=# last_name varchar(255) not null,
online_shop_db=# phone varchar(30) not null,
online_shop_db=# email varchar(255) not null
online_shop_db=# );
CREATE TABLE
online_shop_db=# \d
List of relations
Schema | Name | Type | Owner
-----+-----+-----+-----
public | customer | table | postgres
public | customer_customer_id_seq | sequence | postgres
(2 rows)

online_shop_db=# \d customer
Table "public.customer"
Column | Type | Collation | Nullable | Default
-----+-----+-----+-----+-----
customer_id | integer | | not null | nextval('customer_customer_id_seq'::regclass)
first_name | character varying(255) | | not null | 
last_name | character varying(255) | | not null | 
phone | character varying(30) | | not null | 
email | character varying(255) | | not null | 
Indexes:
    "customer_pkey" PRIMARY KEY, btree (customer_id)
online_shop_db=#
```

Рисунок 4 – Создание таблицы клиента и просмотр её структуры

```

vitaly@ubuntu: ~
online_shop_db=# create table product(
online_shop_db(# product_id serial primary key,
online_shop_db(# name varchar(255) not null,
online_shop_db(# description text,
online_shop_db(# price money not null
online_shop_db(# );
CREATE TABLE
online_shop_db=# \d

List of relations
Schema | Name | Type | Owner
-----+-----+-----+-----
public | customer | table | postgres
public | customer_customer_id_seq | sequence | postgres
public | product | table | postgres
public | product_product_id_seq | sequence | postgres
(4 rows)

online_shop_db=# \d product

Table "public.product"
Column | Type | Collation | Nullable | Default
-----+-----+-----+-----+-----
product_id | integer | | not null | nextval('product_product_id_seq'::regclass)
name | character varying(255) | | not null |
description | text | | |
price | money | | not null |
Indexes:
    "product_pkey" PRIMARY KEY, btree (product_id)
online_shop_db=#

```

Рисунок 5 – Создание таблицы продукта и просмотр её структуры

```

vitaly@ubuntu: ~
online_shop_db=# create table product_photo(
online_shop_db(# product_photo_id serial primary key,
online_shop_db(# photo_url varchar(255) not null,
online_shop_db(# product_id integer references product(product_id) not null
online_shop_db(# );
CREATE TABLE
online_shop_db=# \d

List of relations
Schema | Name | Type | Owner
-----+-----+-----+-----
public | customer | table | postgres
public | customer_customer_id_seq | sequence | postgres
public | product | table | postgres
public | product_photo | table | postgres
public | product_photo_product_photo_id_seq | sequence | postgres
public | product_product_id_seq | sequence | postgres
(6 rows)

online_shop_db=# \d product_photo

Table "public.product_photo"
Column | Type | Collation | Nullable | Default
-----+-----+-----+-----+-----
product_photo_id | integer | | not null | nextval('product_photo_product_photo_id_seq'::regclass)
photo_url | character varying(255) | | not null |
product_id | integer | | not null |
Indexes:
    "product_photo_pkey" PRIMARY KEY, btree (product_photo_id)
Foreign-key constraints:
    "product_photo_product_id_fkey" FOREIGN KEY (product_id) REFERENCES product(product_id)
online_shop_db=#

```

Рисунок 6 – Создание таблицы фотографии товара и просмотр её структуры

```

online_shop_db=# create table cart_product(
online_shop_db=# cart_id integer references cart(cart_id) not null,
online_shop_db=# product_id integer references product(product_id) not null
online_shop_db=# );
CREATE TABLE
online_shop_db=# \d

```

Schema	Name	Type	Owner
public	cart	table	postgres
public	cart_cart_id_seq	sequence	postgres
public	cart_product	table	postgres
public	customer	table	postgres
public	customer_customer_id_seq	sequence	postgres
public	product	table	postgres
public	product_photo	table	postgres
public	product_photo_product_photo_id_seq	sequence	postgres
public	product_product_id_seq	sequence	postgres

```

(9 rows)

online_shop_db=# \d cart_product
Table "public.cart_product"
  Column      | Type   | Collation | Nullable | Default
-----+-----+-----+-----+-----
 cart_id      | integer |           | not null |
 product_id   | integer |           | not null |
Foreign-key constraints:
 "cart_product_cart_id_fkey" FOREIGN KEY (cart_id) REFERENCES cart(cart_id)
 "cart_product_product_id_fkey" FOREIGN KEY (product_id) REFERENCES product(product_id)
online_shop_db=#

```

Рисунок 7 – Создание таблицы товара в корзине и просмотр её структуры

Теперь необходимо заполнить БД информацией. Заполним таблицы клиента, товара и фотографии к товару. Пример заполнения этих таблиц представлен на рисунках 8, 9 и 10 соответственно.

```

product_id | integer |           | not null |
Foreign-key constraints:
 "cart_product_cart_id_fkey" FOREIGN KEY (cart_id) REFERENCES cart(cart_id)
 "cart_product_product_id_fkey" FOREIGN KEY (product_id) REFERENCES product(product_id)

online_shop_db=# \d customer
Table "public.customer"
  Column      | Type           | Collation | Nullable | Default
-----+-----+-----+-----+-----
 customer_id  | integer        |           | not null | nextval('customer_customer_id_seq'::regclass)
 first_name   | character varying(255) |           | not null |
 last_name    | character varying(255) |           | not null |
 phone        | character varying(30)  |           | not null |
 email        | character varying(255) |           | not null |
Indexes:
 "customer_pkey" PRIMARY KEY, btree (customer_id)
Referenced by:
 TABLE "cart" CONSTRAINT "cart_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customer(customer_id)

online_shop_db=# insert into customer(last_name, first_name, phone, email) values ('Иванов', 'Иван', '+78005553535', 'ivanov.i@mail.ru'), ('Петров', 'Петр', '+79046948525', 'petrov.p@mail.ru'), ('Посаднев', 'Виталий', '+79997506544', 'elemabor@gmail.com');
INSERT 0 3
online_shop_db=# select * from customer;
 customer_id | first_name | last_name | phone        | email
-----+-----+-----+-----+-----
 1 | Иван      | Иванов    | +78005553535 | ivanov.i@mail.ru
 2 | Петр      | Петров    | +79046948525 | petrov.p@mail.ru
 3 | Виталий   | Посаднев  | +79997506544 | elemabor@gmail.com
(3 rows)

online_shop_db=#

```

Рисунок 8 – Заполнение таблицы клиентов


```

vitaly@ubuntu:~
1 | Иван      | Иванов | +78005553535 | ivanov.i@mail.ru
2 | Петр      | Петров | +79046948525 | petrov.p@mail.ru
3 | Виталий   | Посаднев | +79997506544 | elemabor@gmail.com
(3 rows)

online_shop_db=# \d product
               Table "public.product"
   Column      |      Type      | Collation | Nullable |      Default
-----+-----+-----+-----+-----
product_id     | integer         |           | not null | nextval('product_product_id_seq'::regclass)
name           | character varying(255) |           | not null |
description     | text            |           |          |
price          | money           |           | not null |
Indexes:
    "product_pkey" PRIMARY KEY, btree (product_id)
Referenced by:
    TABLE "cart_product" CONSTRAINT "cart_product_product_id_fkey" FOREIGN KEY (product_id) REFERENCES product(product_id)
    TABLE "product_photo" CONSTRAINT "product_photo_product_id_fkey" FOREIGN KEY (product_id) REFERENCES product(product_id)

online_shop_db=# insert into product(name, description, price) values ('MacBook Pro 13 2020, M1', 'Тут будет описание, но не сейчас', 140000), ('iPhone 12 Pro', 'Новая модель старого айфона', 109900);
INSERT 0 2
online_shop_db=# select * from product;
 product_id | name | description | price
-----+-----+-----+-----
1 | MacBook Pro 13 2020, M1 | Тут будет описание, но не сейчас | 140 000,00 ₺
2 | iPhone 12 Pro | Новая модель старого айфона | 109 900,00 ₺
(2 rows)

online_shop_db=#

```

Рисунок 9 – Заполнение таблицы товаров

```

vitaly@ubuntu:~
INSERT 0 2
online_shop_db=# select * from product;
 product_id | name | description | price
-----+-----+-----+-----
1 | MacBook Pro 13 2020, M1 | Тут будет описание, но не сейчас | 140 000,00 ₺
2 | iPhone 12 Pro | Новая модель старого айфона | 109 900,00 ₺
(2 rows)

online_shop_db=# \d product_photo
               Table "public.product_photo"
   Column      |      Type      | Collation | Nullable |      Default
-----+-----+-----+-----+-----
product_photo_id | integer         |           | not null | nextval('product_photo_product_photo_id_seq'::regclass)
photo_url        | character varying(255) |           | not null |
product_id       | integer         |           | not null |
Indexes:
    "product_photo_pkey" PRIMARY KEY, btree (product_photo_id)
Foreign-key constraints:
    "product_photo_product_id_fkey" FOREIGN KEY (product_id) REFERENCES product(product_id)

online_shop_db=# insert into product_photo (photo_url, product_id) values ('macbook_pro_1.png', 1), ('macbook_pro_2.png', 2), ('iphone_12.png', 2);
INSERT 0 3
online_shop_db=# select * from product_photo;
 product_photo_id | photo_url | product_id
-----+-----+-----
1 | macbook_pro_1.png | 1
2 | macbook_pro_2.png | 2
3 | iphone_12.png | 2
(3 rows)

online_shop_db=#

```

Рисунок 10 – Заполнение таблицы фотографий к товарам

Теперь необходимо создать резервную копию базы данных, для этого воспользуемся следующей командой: `pg_dump -U postgres online_shop_db > online_shop_db.bak`. После ввода пароля пользователя в текущей папке будет создан файл `online_shop_db.bak` с полной информацией о БД.

Зайдем снова в консоль PostgreSQL и удалим нашу базу данных с помощью команды `drop database online_shop_db; .` Пример успешного выполнения данной команды изображен на рисунке 11.

```

vitaly@ubuntu: ~/lab8
postgres=# \l

```

Name	Owner	Encoding	Collate	Ctype	Access privileges
medical_db	medical_user	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/medical_user medical_user=CTc/medical_user +
online_shop_db	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
symfony_db	symfony_user	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/symfony_user symfony_user=CTc/symfony_user +
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres postgres=CTc/postgres +
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres +
todolist_db	django_graphene	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/django_graphene django_graphene=CTc/django_graphene +

```

(7 rows)

postgres=# drop database online_shop_db;
DROP DATABASE
postgres=# \l

```

Name	Owner	Encoding	Collate	Ctype	Access privileges
medical_db	medical_user	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/medical_user medical_user=CTc/medical_user +
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
symfony_db	symfony_user	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/symfony_user symfony_user=CTc/symfony_user +
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres postgres=CTc/postgres +
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres +
todolist_db	django_graphene	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/django_graphene django_graphene=CTc/django_graphene +

```

(6 rows)

postgres=#

```

Рисунок 11 – Успешное удаление текущей базы данных

Также создадим новую базу данных с названием `online_shop_restored`. Для этого необходимо воспользоваться командой `create database online_shop_restored;` . Пример успешного создания новой базы данных представлен на рисунке 12.

```

vitaly@ubuntu: ~/lab8
postgres=# \l

```

Name	Owner	Encoding	Collate	Ctype	Access privileges
medical_db	medical_user	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/medical_user medical_user=CTc/medical_user +
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
symfony_db	symfony_user	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/symfony_user symfony_user=CTc/symfony_user +
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres postgres=CTc/postgres +
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres +
todolist_db	django_graphene	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/django_graphene django_graphene=CTc/django_graphene +

```

(6 rows)

postgres=# create database online_shop_restored;
CREATE DATABASE
postgres=# \l

```

Name	Owner	Encoding	Collate	Ctype	Access privileges
medical_db	medical_user	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/medical_user medical_user=CTc/medical_user +
online_shop_restored	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
symfony_db	symfony_user	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/symfony_user symfony_user=CTc/symfony_user +
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres postgres=CTc/postgres +
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres +
todolist_db	django_graphene	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/django_graphene django_graphene=CTc/django_graphene +

```

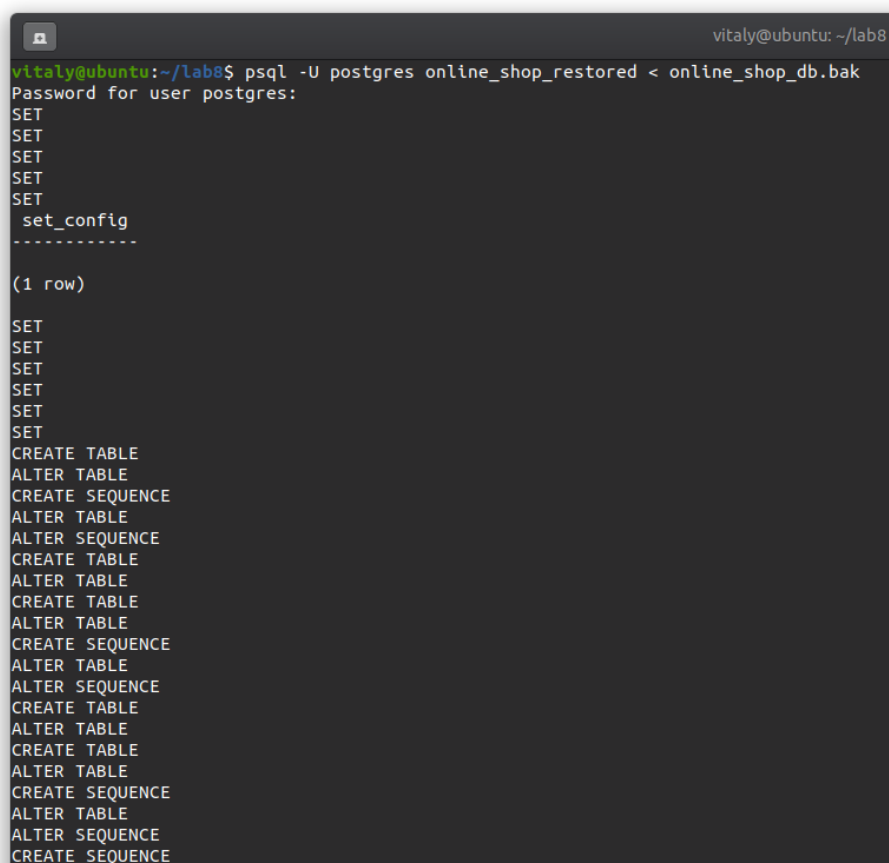
(7 rows)

postgres=#

```

Рисунок 12 – Создание новой базы данных

Последним шагом необходимо восстановить базу данных `online_shop_restored` из резервной копии `online_shop_db.bak`. Для этого необходимо выполнить следующую команду `psql -U postgres online_shop_restored < online_shop_db.bak`. Пример выполнения данной команды изображен на рисунке 13.



```
vitaly@ubuntu: ~/lab8
vitaly@ubuntu:~/lab8$ psql -U postgres online_shop_restored < online_shop_db.bak
Password for user postgres:
SET
SET
SET
SET
SET
set_config
-----
(1 row)

SET
SET
SET
SET
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER TABLE
ALTER SEQUENCE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER TABLE
ALTER SEQUENCE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER TABLE
ALTER SEQUENCE
CREATE SEQUENCE
```

Рисунок 13 – Восстановление БД из резервной копии

Теперь зайдём в консоль PostgreSQL, подключимся к нашей БД и посмотрим на её содержимое. Результат подключения к БД и просмотра списка таблиц представлен на рисунке 14. Просмотр содержимого таблицы клиентов изображен на рисунке 15.

```
vitaly@ubuntu: ~/lab8
Type "help" for help.
postgres=# \l
                                List of databases
  Name          | Owner          | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----
 medical_db     | medical_user   | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/medical_user
               |               |          |             |             | medical_user=CTc/medical_user +
 online_shop_restored | postgres      | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres       | postgres      | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 symfony_db     | symfony_user   | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/symfony_user
               |               |          |             |             | symfony_user=CTc/symfony_user +
 template0      | postgres      | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
               |               |          |             |             | postgres=CTc/postgres +
 template1      | postgres      | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
               |               |          |             |             | postgres=CTc/postgres +
 todolist_db    | django_graphene | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/django_graphene
               |               |          |             |             | django_graphene=CTc/django_graphene +
(7 rows)

postgres=# \c online_shop_restored
You are now connected to database "online_shop_restored" as user "postgres".
online_shop_restored=# \d
                                List of relations
 Schema | Name          | Type  | Owner
-----+-----+-----+-----
 public | cart          | table | postgres
 public | cart_cart_id_seq | sequence | postgres
 public | cart_product  | table | postgres
 public | customer      | table | postgres
 public | customer_customer_id_seq | sequence | postgres
 public | product       | table | postgres
 public | product_photo | table | postgres
 public | product_photo_product_photo_id_seq | sequence | postgres
 public | product_product_id_seq | sequence | postgres
(9 rows)

online_shop_restored=#
```

Рисунок 14 – Подключение к консоли PostgreSQL и просмотр содержимого БД

```
vitaly@ubuntu: ~/lab8
online_shop_restored | postgres | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
 postgres           | postgres | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
 symfony_db         | symfony_user | UTF8 | en_US.UTF-8 | en_US.UTF-8 | =Tc/symfony_user
                   |           |      |             |             | symfony_user=CTc/symfony_user +
 template0          | postgres | UTF8 | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
                   |           |      |             |             | postgres=CTc/postgres +
 template1          | postgres | UTF8 | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
                   |           |      |             |             | postgres=CTc/postgres +
 todolist_db        | django_graphene | UTF8 | en_US.UTF-8 | en_US.UTF-8 | =Tc/django_graphene
                   |           |      |             |             | django_graphene=CTc/django_graphene +
(7 rows)

postgres=# \c online_shop_restored
You are now connected to database "online_shop_restored" as user "postgres".
online_shop_restored=# \d
                                List of relations
 Schema | Name          | Type  | Owner
-----+-----+-----+-----
 public | cart          | table | postgres
 public | cart_cart_id_seq | sequence | postgres
 public | cart_product  | table | postgres
 public | customer      | table | postgres
 public | customer_customer_id_seq | sequence | postgres
 public | product       | table | postgres
 public | product_photo | table | postgres
 public | product_photo_product_photo_id_seq | sequence | postgres
 public | product_product_id_seq | sequence | postgres
(9 rows)

online_shop_restored=# select * from customer;
 customer_id | first_name | last_name | phone | email
-----+-----+-----+-----+-----
          1 | Иван      | Иванов    | +78005553535 | ivanov.i@mail.ru
          2 | Петр      | Петров    | +79046948525 | petrov.p@mail.ru
          3 | Виталий   | Посаднев  | +79997506544 | elemabor@gmail.com
(3 rows)

online_shop_restored=#
```

Рисунок 15 – Просмотр содержимого таблицы клиента

Вывод

При выполнении данной лабораторной работы была изучена работа со средствами создания резервных копий баз данных на примере утилиты *pg_dump* для СУБД PostgreSQL.