Белорусский Государственный Университет
Информатики и Радиоэлектроники
Кафедра ЭВМ

Отчет по лабораторной работе № 3

Тема: «Реализация SQL-запросов для создания схемы базы данных»

Выполнили:

ст. гр. 950503
Зарубо Д. Ю
Ященко В.П

Проверила:
Куприянова Д.В

Минск 2022

1 ЦЕЛЬ РАБОТЫ

В лабораторной работе выполняется реализация схемы базы данных по ранее построенной реляционной схеме данных.
Требуется сформировать SQL-запросы для создания таблиц базы данных и выполнить их в СУБД. Требуется заполнить таблицы данными с помощью оператора INSERT.

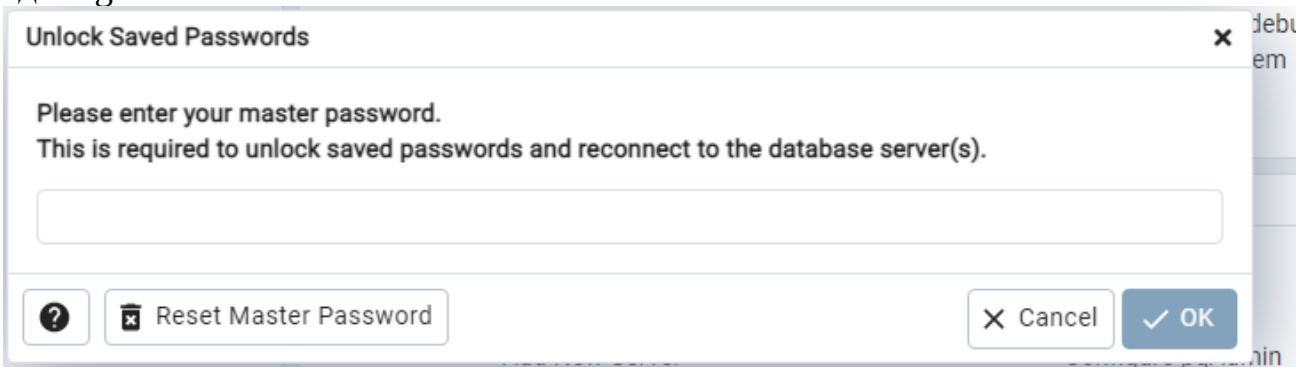
2 ЗАДАНИЕ

1) Создать в СУБД новую схему данных для хранения пользовательских объектов (см. часть 2).
2) В этой новой схеме данных с помощью скрипта с запросами на языке DDL SQL реализовать таблицы, соответствующие реляционным отношениям схемы данных полученной в лабораторной работе No2, с помощью одного (желательно) оператора CREATE TABLE для каждой таблицы в следующем порядке:
- реализовать простую структуру таблиц, включающую только набор столбцов с добавлением описаний первичного ключа;
- дополнить описание таблицы реализацией ограничений для описания внешних ключей; для внешних ключей установить свойства контроля целостности данных (каскадное удаление и обновление), если это возможно в целевой СУБД;
- дополнить описание таблицы реализацией ограничений для описания бизнес-правил;
- дополнить описание таблицы реализацией комментариев для значимых элементов таблицы.
3) Заполнить с помощью SQL-скрипта с использованием оператора INSERT таблицы строками данных для проверки правильного выбора первичных ключей и работоспособности ссылок между таблицами:
- строками данных сначала заполнять мастер-таблицы (или таблицы, которые НЕ ссылаются на другие таблицы);
- в каждую таблицу добавить 5 – 10 строк осмысленных данных;
- если не удается добавить данные в таблицу по причине нарушения уникальности первичного ключа, то следует перепроверить описание этого первичного ключа и его смысл для реального мира;
- если не удается добавить данные в таблицу по причине нарушения ссылочной целостности, то следует убедиться, что целевые данные существуют, иначе перепроверить описание внешнего ключа.
4) Рассмотреть простые действия по изменению структуры таблицы (переименование столбца таблицы, добавление и удаление ограничений на столбец таблицы или всю таблицу) и реализовать их с помощью оператора ALTER TABLE.
5) Создать временную таблицу с помощью оператора CREATE TABLE и

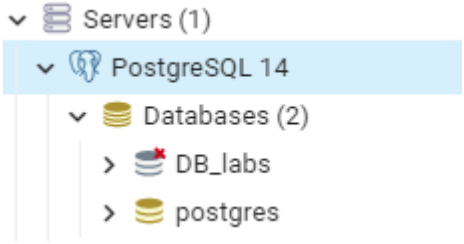
удалить ее с помощью оператора DROP TABLE.
6) Экспортировать результаты работы в SQL-скрипт (см. часть 2), сравнить полученный скрипт со скриптами, созданными на этапах 2 и 3.

3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ

1. Вход в PgAdmin



2. Подключение к базам данных



3. Реализация реляционной модели с помощью команд DDL SQL

- Реализация структуры таблиц в виде набора столбцов с добавлением описаний только первичных ключей;

```sql
CREATE TABLE IF NOT EXISTS video_rental.clients
(
    id integer,
    first_name character varying,
    last_name character varying,
    passport_number character varying,
    payment_id integer,
    rent_id integer,
    CONSTRAINT client_id PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS video_rental.rent
(
    id integer,
    start_date date,
    duration integer,
    payment_id integer,
    department_id integer,
    CONSTRAINT rent_id PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS video_rental.payment
(
    id integer,
    payment_amount integer,
    payment_number character varying,
    payment_date date,
    CONSTRAINT payment_id PRIMARY KEY (id)
);
```

```sql
CREATE TABLE IF NOT EXISTS video_rental.films
(
    id integer,
    publishing_company character varying,
    age_constraints integer,
    title character varying,
    amount integer,
    film_author character,
    CONSTRAINT film_id PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS video_rental.departments
(
    id integer,
    department_address character varying,
    department_capacity integer,
    CONSTRAINT department_id PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS video_rental.genres
(
    id integer,
    genre_titel character varying,
    genre_raiting real,
    CONSTRAINT genre_id PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS video_rental.films_genres
(
    films_id integer,
    genres_id integer
);

CREATE TABLE IF NOT EXISTS video_rental.rent_films
(
    rent_id integer,
    films_id integer
);

CREATE TABLE IF NOT EXISTS video_rental.departments_films
(
    departments_id integer,
    films_id integer
);
```

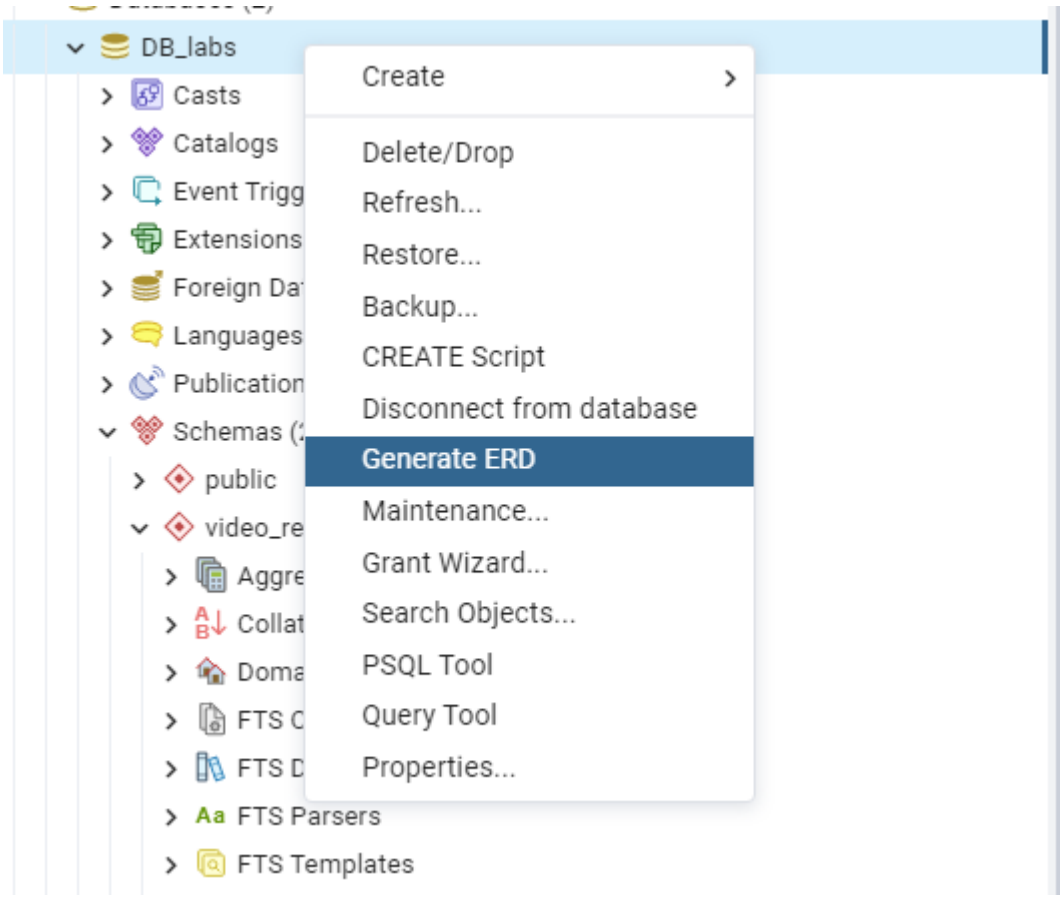
- Реализация ограничений таблиц с добавлением описаний внешних ключей, бизнес-правил

```
ALTER TABLE video_rental.clients
    ADD FOREIGN KEY (payment_id)
    REFERENCES video_rental.payment (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE video_rental.clients
    ADD FOREIGN KEY (rent_id)
    REFERENCES video_rental.rent (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE video_rental.rent
    ADD FOREIGN KEY (payment_id)
    REFERENCES video_rental.payment (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE  video_rental.rent
    ADD FOREIGN KEY (department_id)
    REFERENCES video_rental.departments (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE  video_rental.films_genres
    ADD FOREIGN KEY (films_id)
    REFERENCES video_rental.films (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;
```

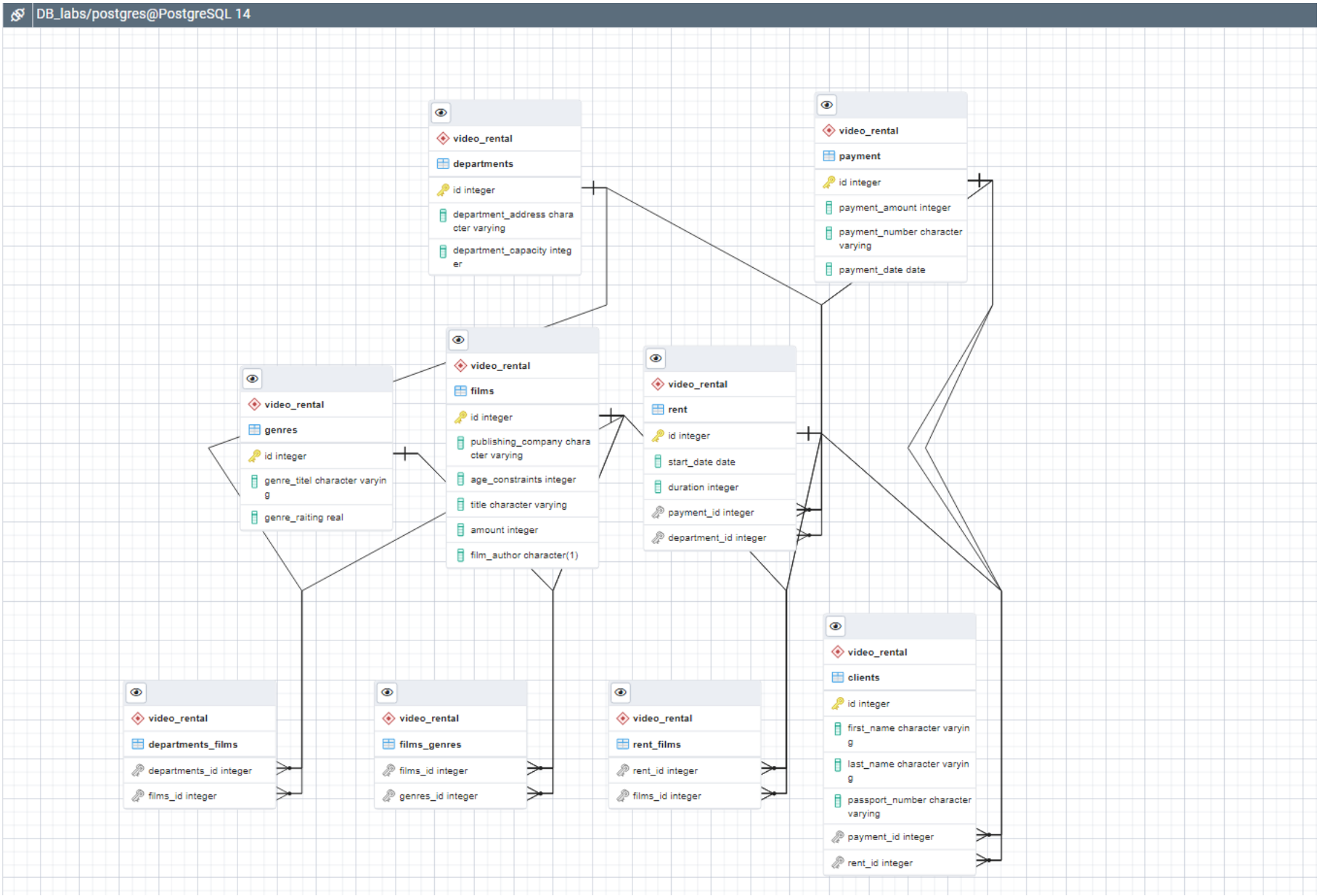
```
ALTER TABLE   video_rental.rent_films
    ADD FOREIGN KEY (rent_id)
    REFERENCES video_rental.rent (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE    video_rental.rent_films
    ADD FOREIGN KEY (films_id)
    REFERENCES video_rental.films (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE    video_rental.departments_films
    ADD FOREIGN KEY (departments_id)
    REFERENCES video_rental.departments (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE    video_rental.departments_films
    ADD FOREIGN KEY (films_id)
    REFERENCES video_rental.films (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;
```

4. Изучение созданной схемы данных

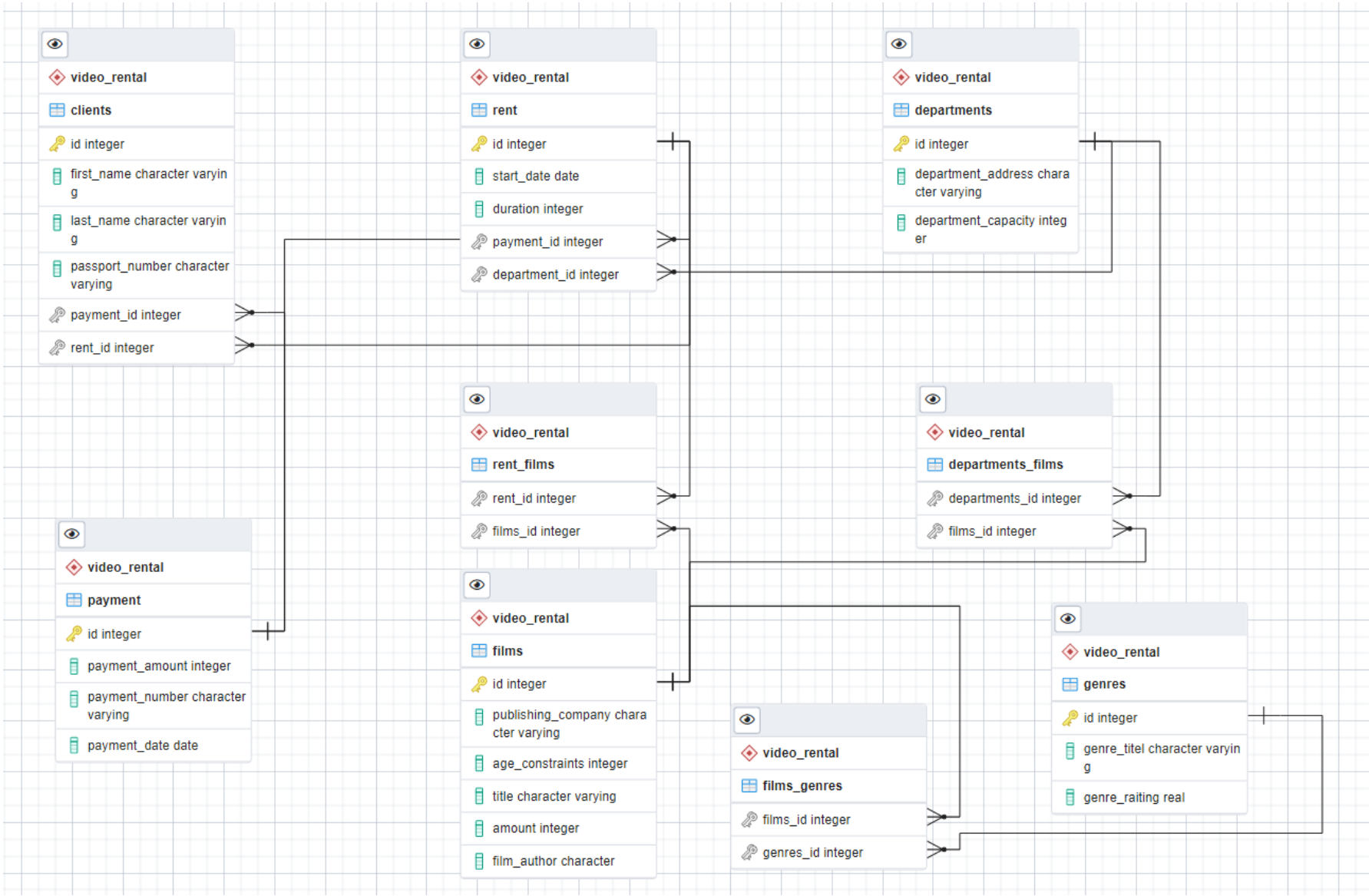
Генерация реляционной диаграммы:


DB_labs
Casts
Catalogs
Event Trigg
Extensions
Foreign Da
Languages
Publication
Schemas (
public
video_re
Aggre
Collat
Doma
FTS C
FTS D
FTS Parsers
FTS Templates
Create
Delete/Drop
Refresh...
Restore...
Backup...
CREATE Script
Disconnect from database
Generate ERD
Maintenance...
Grant Wizard...
Search Objects...
PSQL Tool
Query Tool
Properties...

Сгенерированная диаграмма на основе схемы:



Исходная диаграмма:



5. Заполнение созданных таблиц строками для проверки правильного выбора первичных ключей и работоспособности ссылок между таблицами можно выполнить следующими путями.

   - Заполнение таблицы departments

Query | Query History

```
1  INSERT INTO video_rental.departments (department_address, department_capacity) VALUES
2  ('Minsk, Platonova', 20),
3  ('Minsk, Kolasa', 30),
4  ('Brest, Lenina', 100),
5  ('Grodno, Pobedy', 110),
6  ('Vitebsk, Lenina', 80)
7
8  SELECT * FROM video_rental.departments
```

Data output | Messages | Notifications

	id [PK] integer	department_address character varying	department_capacity integer
1	1	Minsk, Platonova	20
2	2	Minsk, Kolasa	30
3	3	Brest, Lenina	100
4	4	Grodno, Pobedy	110
5	5	Vitebsk, Lenina	80

- Заполнение таблицы genres

Query | Query History

```
1  SELECT * FROM video_rental.genres;
2
3  INSERT INTO video_rental.genres (genre_titel, genre_raiting) VALUES
4  ('Criminal', 8.3),
5  ('Drama', 7.6),
6  ('Action', 9.2),
7  ('Documentary', 6.6),
8  ('Russian', 4.1),
9  ('Foreign', 8.1);
10
11 SELECT * FROM video_rental.genres
```

Data output | Messages | Notifications

	id [PK] integer	genre_titel character varying	genre_raiting real
1	1	Criminal	8.3
2	2	Drama	7.6
3	3	Action	9.2
4	4	Documentary	6.6
5	5	Russian	4.1
6	6	Foreign	8.1

- Заполнение таблицы films

Query | Query History

```sql
SELECT * FROM video_rental.films;

INSERT INTO video_rental.films (title, film_author, publishing_company, age_constraints, amount) VALUES
('The Green Mile', 'Frank Darabont', 'Castle Rock Entertaiment', 16, 20),
('Forrest Gump', 'Robert Zemekis', 'Paramount Pictures', 16, 15),
('1+1(The Intouchables)','Olivier Nakache', 'Gaumont', 16, 10 ),
('Back to the future', 'Robert Zemekis', 'Universal Pictures', 12, 25),
('The Polar Express', 'Robert Zemekis', 'Castle Rock Entertaiment', 6, 12),

('The Misk', 'Frank Darabont', 'Darkwoods Productions', 18, 5),
('Pulp Fiction', 'Quentin Tarantino', 'Miramax', 18, 18),
('Once Upon a Time in Hollywood', 'Quentin Tarantino', 'Columbia Pictures', 18, 5),
('Fear and Loathing in Las Vegas', 'Terry Gilliam', 'Rhino Films', 18, 3),
('12 Monkeys','Terry Gilliam', 'Universal Pictures', 16, 10 );

SELECT * FROM video_rental.films;
```

Data output | Messages | Notifications

	id [PK] integer	publishing_company character varying	age_constraints integer	title character varying	amount integer	film_author character varying
1	1	Castle Rock Entertai...	16	The Green Mile	20	Frank Darabont
2	2	Paramount Pictures	16	Forrest Gump	15	Robert Zemekis
3	3	Gaumont	16	1+1(The Intouch...	10	Olivier Nakache
4	4	Universal Pictures	12	Back to the future	25	Robert Zemekis
5	5	Castle Rock Entertai...	6	The Polar Express	12	Robert Zemekis
6	6	Darkwoods Producti...	18	The Misk	5	Frank Darabont
7	7	Miramax	18	Pulp Fiction	18	Quentin Tarantino
8	8	Columbia Pictures	18	Once Upon a Tim...	5	Quentin Tarantino
9	9	Rhino Films	18	Fear and Loathin...	3	Terry Gilliam
10	10	Universal Pictures	16	12 Monkeys	10	Terry Gilliam

- Заполнение таблицы payment

Query Query History

```
1  SELECT * FROM video_rental.payment;
2
3
4  INSErT INTO video_rental.payment (payment_amount, payment_date, payment_number) VALUES
5  (53, '2022-09-28', 100034),
6  (75, '2022-10-01', 195954),
7  (32, '2022-10-01', 195323),
8  (123, '2022-10-02', 193923),
9  (42, '2022-10-03', 134234),
10 (77, '2022-10-03', 184843);
11
12
13 SELECT * FROM video_rental.payment;
```

Data output Messages Notifications

	id [PK] integer	payment_amount integer	payment_number character varying	payment_date date
1	1	53	100034	2022-09-28
2	2	75	195954	2022-10-01
3	3	32	195323	2022-10-01
4	4	123	193923	2022-10-02
5	5	42	134234	2022-10-03
6	6	77	184843	2022-10-03

- Заполнение таблицы rent

```
INSERT INTO video_rental.rent(department_id, payment_id, start_date, duration) VALUES
((SELECT id FROM video_rental.departments WHERE departments.department_address = 'Minsk, Platonova'),
 (SELECT id FROM video_rental.payment WHERE payment.payment_number = '100034'),
 '2022-09-28', 10
),
((SELECT id FROM video_rental.departments WHERE departments.department_address = 'Minsk, Kolasa'),
 (SELECT id FROM video_rental.payment WHERE payment.payment_number = '195954'),
 '2022-10-01', 5
),
((SELECT id FROM video_rental.departments WHERE departments.department_address = 'Minsk, Platonova'),
 (SELECT id FROM video_rental.payment WHERE payment.payment_number = '195323'),
 '2022-10-01', 5
),
((SELECT id FROM video_rental.departments WHERE departments.department_address = 'Brest, Lenina'),
 (SELECT id FROM video_rental.payment WHERE payment.payment_number = '193923'),
 '2022-10-02', 7
),
((SELECT id FROM video_rental.departments WHERE departments.department_address = 'Grodno, Pobedy'),
 (SELECT id FROM video_rental.payment WHERE payment.payment_number = '134234'),
 '2022-10-03', 10
),

((SELECT id FROM video_rental.departments WHERE departments.department_address = 'Vitebsk, Lenina'),
 (SELECT id FROM video_rental.payment WHERE payment.payment_number = '184843'),
 '2022-10-03', 5
);

```

Data output | Messages | Notifications

	id [PK] integer	start_date date	duration integer	payment_id integer	department_id integer
1	2	2022-09-28	10	1	1
2	3	2022-10-01	5	2	2
3	4	2022-10-01	5	3	1
4	5	2022-10-02	7	4	3
5	6	2022-10-03	10	5	4
6	7	2022-10-03	5	6	5

- Заполнение таблицы clients

Query Query History

```
1  SELECT * FROM video_rental.clients;
2
3  INSERT INTO video_rental.clients (first_name, last_name, passport_number, rent_id, payment_id) VALUES
4  ('Ivan', 'Ivanov', 'P001', 4, (SELECT payment_id from video_rental.rent WHERE id = 4)),
5  ('Petr', 'Petrov', 'P010', 2, (SELECT payment_id from video_rental.rent WHERE id = 2)),
6  ('Sergey', 'Sergeev', 'P021', 3, (SELECT payment_id from video_rental.rent WHERE id = 3)),
7  ('Alexey', 'Lehovich', 'P342', 6, (SELECT payment_id from video_rental.rent WHERE id = 6)),
8  ('Igor', 'Suhoy', 'P232', 5, (SELECT payment_id from video_rental.rent WHERE id = 5)),
9  ('Evegn', 'Evgenov', 'P091', 7, (SELECT payment_id from video_rental.rent WHERE id = 7));
10
11 SELECT * FROM video_rental.clients;
```

Data output Messages Notifications

	id [PK] integer	first_name character varying	last_name character varying	passport_number character varying	payment_id integer	rent_id integer
1	1	Ivan	Ivanov	P001	3	4
2	2	Petr	Petrov	P010	1	2
3	3	Sergey	Sergeev	P021	2	3
4	4	Alexey	Lehovich	P342	5	6
5	5	Igor	Suhoy	P232	4	5
6	6	Evegn	Evgenov	P091	6	7

- Заполнение таблицы films_genres

```
SELECT * FRom video_rental.films_genres;
INSERT INTO video_rental.films_genres (films_id, genres_id) VALUES
(1, 2),
(1, 6),
(2, 2),
(2, 6),
(3, 2),
(3, 6),
(3, 4),
(4, 3),
(4, 6),
(5, 3),
(5, 6),
(6, 2),
(6, 3),
(6, Loading...
(7, 1),
(7, 2),
(7, 3),
(7, 6),
(8, 1),
(8, 3),
(8, 6),
(9, 1),
(9, 3),
(9, 6),
(10, 2),
(10, 3),
(10, 6);
SELECT * FRom video_rental.films_genres;
```

	films_id integer	genres_id integer
1	1	2
2	1	6
3	2	2
4	2	6
5	3	2
6	3	6
7	3	4
8	4	3
9	4	6
10	5	3
11	5	6
12	6	2
13	6	3
14	6	6
15	7	1
16	7	2
17	7	3
18	7	6
19	8	1
20	8	3
21	8	6
22	9	1
23	9	3
24	9	6
25	10	2
26	10	3
27	10	6

- Заполнение таблицы rent_films

Query Query History

```
1  SELECT * FROM video_rental.rent_films;
2
3
4  INSERT INTO video_rental.rent_films (rent_id, films_id) VALUES
5
6  (2, 4),
7  (2, 9),
8  (3, 1),
9  (4, 4),
10 (4, 5),
11 (4, 7),
12 (5, 10),
13 (5, 2),
14 (5, 8),
15 (6, 2),
16 (7, 6);
17
18
19 SELECT * FROM video_rental.rent_films;
```

Data output Messages Notifications

	rent_id integer	films_id integer
1	2	4
2	2	9
3	3	1
4	4	4
5	4	5
6	4	7
7	5	10
8	5	2
9	5	8
10	6	2

Заполнение таблицы department_films

```
INSERT INTO video_rental.departments_films(departments_id, films_id)
(SELECT d.id, f.id FROM video_rental.departments as d
CROSS JOIN video_rental.films as f);
```

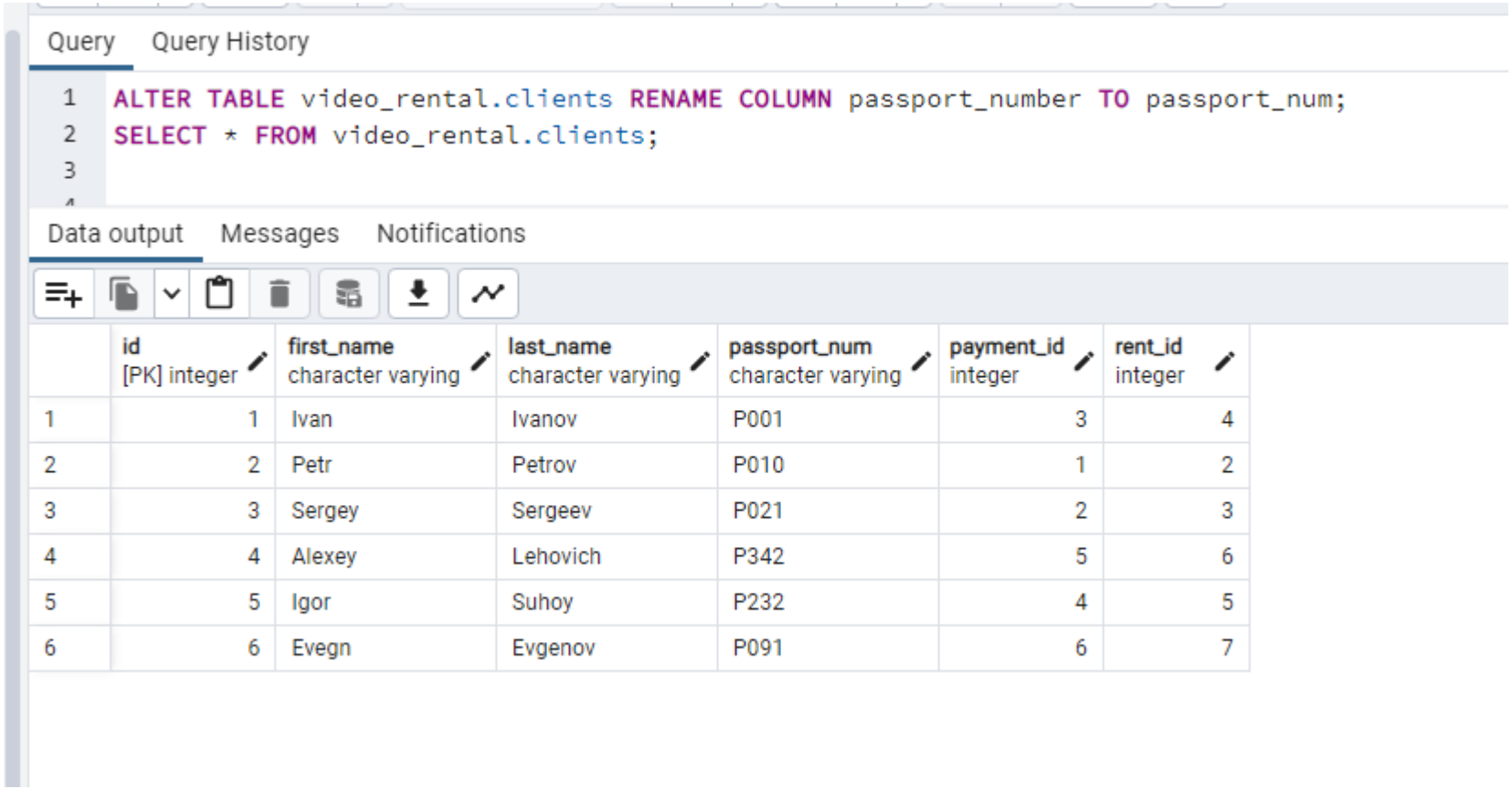
```
2
3  -- INSERT INTO video_rental.departments_films(departments_id, films_id)
4  -- (SELECT d.id, f.id FROM video_rental.departments as d
5  -- CROSS JOIN video_rental.films as f);
6
7  SELECT * FROM video_rental.departments_films;
```

Data output | Messages | Notifications

	departments_id integer	films_id integer
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	1	6
7	1	7
8	1	8
9	1	9
10	1	10
11	2	1
12	2	2
13	2	3
14	2	4
15	2	5
16	2	6
17	2	7
18	2	8
19	2	9
20	2	10
21	3	1
22	3	2
23	3	3
24	3	4
25	3	5
26	3	6
27	3	7
28	3	8
29	3	9
30	3	10
31	4	1
32	4	2
33	4	3
34	4	4
35	4	5
36	4	6
37	4	7
38	4	8
39	4	9
40	4	10
41	5	1
42	5	2
43	5	3
44	5	4
45	5	5
46	5	6
47	5	7
48	5	8
49	5	9
50	5	10

6. Простые действия по изменению структуры таблицы

- Переименование столбца таблицы



	id [PK] integer	first_name character varying	last_name character varying	passport_num character varying	payment_id integer	rent_id integer
1	1	Ivan	Ivanov	P001	3	4
2	2	Petr	Petrov	P010	1	2
3	3	Sergey	Sergeev	P021	2	3
4	4	Alexey	Lehovich	P342	5	6
5	5	Igor	Suhoy	P232	4	5
6	6	Evegn	Evgenov	P091	6	7

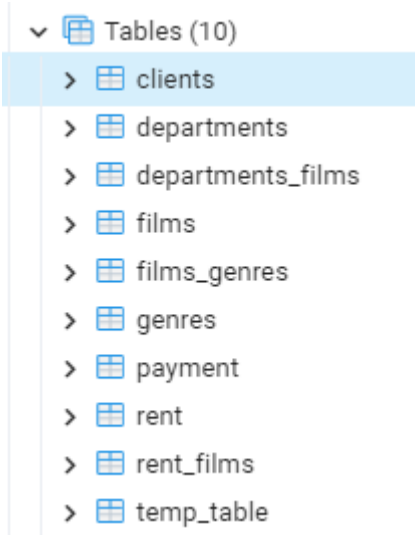
7. Создание временной таблицы и ее удаление

Создание:

```
2  CREATE TABLE video_rental.temp_table(
3    ID   INT              NOT NULL,
4    NAME VARCHAR (20)     NOT NULL,
5    AGE  INT              NOT NULL,
6    PRIMARY KEY (ID)
7  );
```

Data output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 124 msec.
```



Удаление:

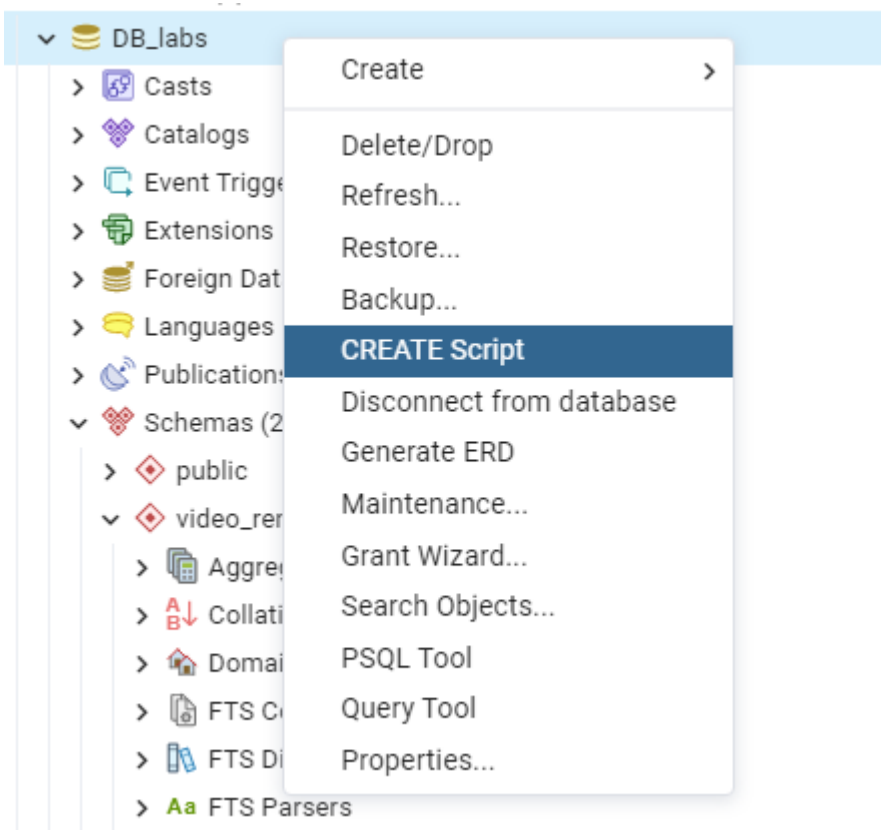
```
1
2  DROP TABLE video_rental.temp_table;
```

Data output | Messages | Notifications

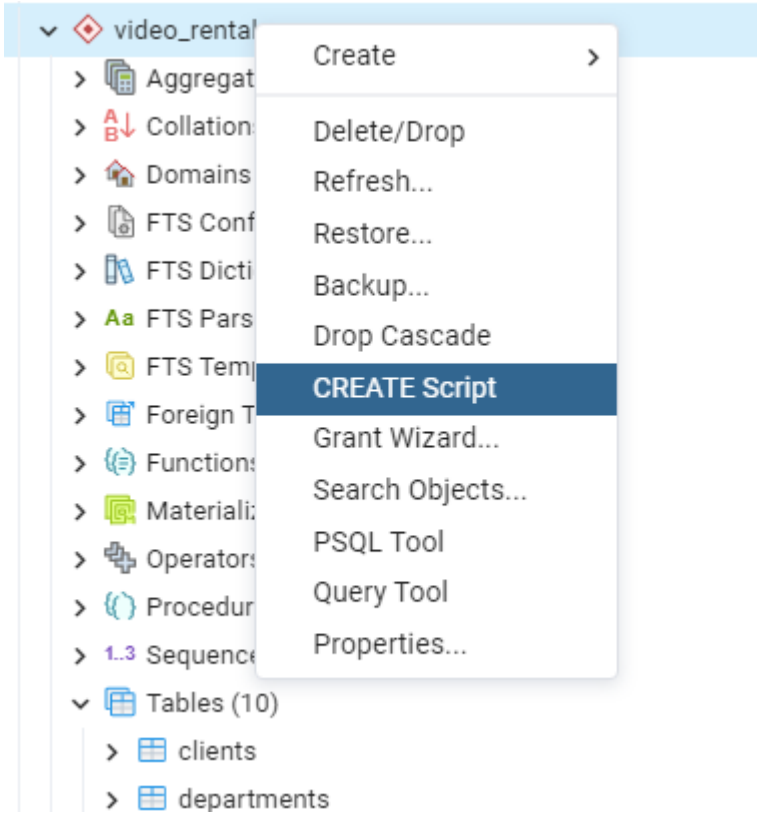
```
DROP TABLE

Query returned successfully in 91 msec.
```

8. Экспорт результатов работы в скрипт:

Базы данных:



Схемы:



Из командной строки:

```
pg_dump --schema-only --no-owner the_database > create_the_tables.sql
```

4 ВЫВОД

В ходе выполнения лабораторной работы были сформированы SQL запросы для работы с базой данных и ее таблицами. Целевые скрипты были выполнены в среде PgAdmin4.