

Национальный исследовательский ядерный университет «МИФИ»

(Московский Инженерно-Физический Институт)

Кафедра №42 «Криптология и кибербезопасность»

## **Лабораторная работа №2-1**

### **«Пользователи. Роли. Привилегии»**

Тимин Александр Б21-515 (2024г.)

1. Определить, в какой схеме находятся таблицы вашей базы данных. Следует ли изменить схему? Следует ли создать несколько отдельных схем для выбранной предметной области? Почему?

В PostgreSQL, если явно не указана схема, все таблицы создаются в схеме `public` (это стандартное поведение). Так как при создании таблиц схема не была явно указана, все таблицы были созданы в `public`.

The screenshot shows the pgAdmin 4 interface. On the left, the 'Object Explorer' pane displays the 'public' schema, which contains several tables. The 'Tables (6)' folder is expanded, showing tables: actors, cast, episodes, reviews, seasons, and shows. The 'SQL' tab is active in the main pane, displaying a query that uses a CTE named 'big\_shows' and another CTE named 'linear\_regression'. The query is as follows:

```
1 WITH "big_shows" AS (  
2     SELECT  
3         "show_id"  
4     FROM  
5         "seasons"  
6     GROUP BY  
7         "show_id"  
8     HAVING  
9         COUNT("show_id") >= 2  
10 ),  
11 "linear_regression" AS (  
12     SELECT  
13         "seasons"."show_id",  
14         "season" * 1.0 AS "x", -- Прив  
15         date_part('epoch', "date") AS "  
16     FROM  
17         "seasons"  
18     JOIN  
19         "big_shows" ON "seasons"."show_  
20 )  
21
```

Below the query, the 'Data Output' tab shows the results of the query. The output is a table with 5 rows and 4 columns: title, next\_season, predicted\_date, and an unnamed column. The data is as follows:

	title	next_season	predicted_date
1	Stranger Things	4	2020-12-02 16:00:00
2	Game of Thron...	4	2014-03-20 16:00:00
3	Breaking Bad	4	2011-04-26 08:00:00
4	The Mandalorian	2	[null]
5	Friends	4	1997-09-18 00:00:00

The status bar at the bottom indicates 'Total rows: 5' and 'Query complete 00:00:00.033'.

Здесь и далее будут приводиться скриншоты из pgAdmin 4.

В общем то можно создать несколько дополнительных схем (например, `series`, `people`, `relationships`):

- **series** для информации о сериалах.
- **people** для информации о людях (актеры, режиссеры и т.д.).
- **relationships** для связей между людьми и сериалами (актеры и их роли).

Потенциально это облегчит администрирование, разграничение прав доступа и масштабирование проекта.

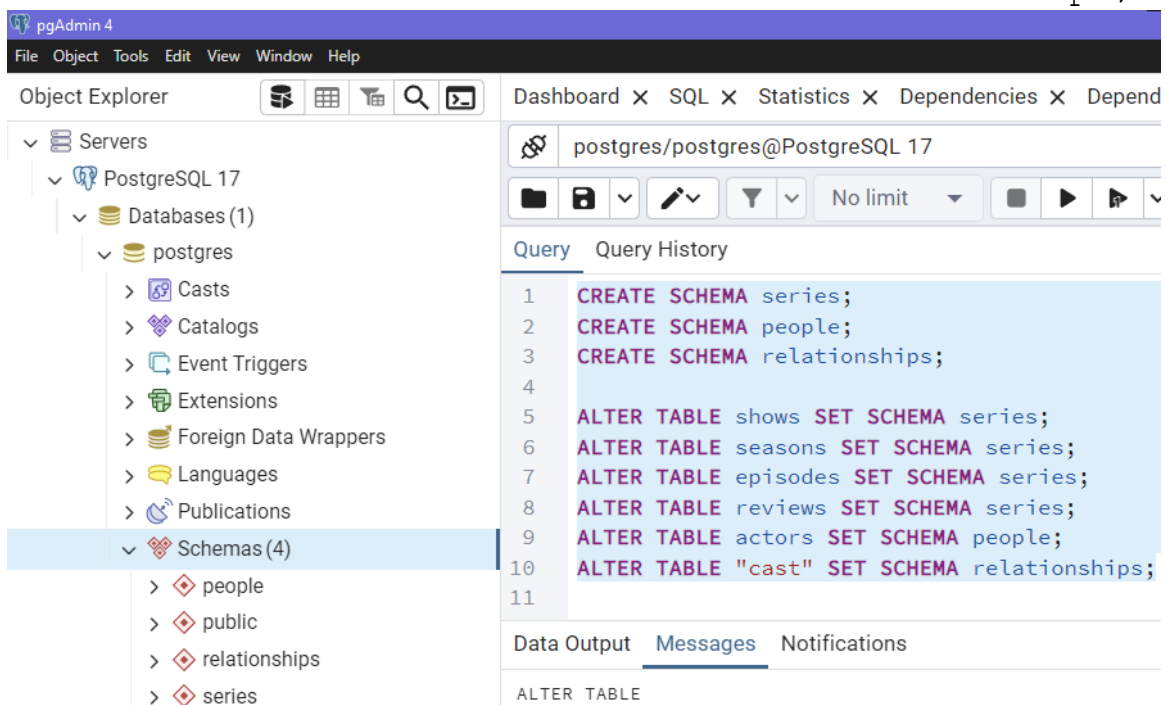
Действия для переноса данных в схемы:

### 1. Создать новые схемы:

```
CREATE SCHEMA series;
CREATE SCHEMA people;
CREATE SCHEMA relationships;
```

### 2. Переместить таблицы в соответствующие схемы:

```
ALTER TABLE shows SET SCHEMA series;
ALTER TABLE seasons SET SCHEMA series;
ALTER TABLE episodes SET SCHEMA series;
ALTER TABLE reviews SET SCHEMA series;
ALTER TABLE actors SET SCHEMA people;
ALTER TABLE "cast" SET SCHEMA relationships;
```



2. Определить, какие роли нужны для нормального функционирования вашей базы данных. Какие системные и объектные привилегии потребуются каждой роли? Понадобятся ли вложенные роли?

Разделение пользователей на роли помогает реализовать принцип наименьших привилегий:

- **Администратор (admin):** Полный контроль над базой данных. Эта роль нужна для настройки структуры базы, управления пользователями и привилегиями.
  - Системные привилегии: CREATEDB, CREATEROLE для управления БД и ролями.
  - Объектные привилегии: ALL PRIVILEGES на все таблицы и схемы.
- **Разработчик (developer):** Разработчики должны иметь возможность изменять структуру таблиц, добавлять новые данные, но не управлять пользователями и базами данных.
  - Системные привилегии: только подключение (LOGIN).
  - Объектные привилегии: создание, чтение, обновление, но без удаления схем.
- **Читатель (reader):** Для аналитиков или других сотрудников, которым нужен только доступ к данным.
  - Системные привилегии: только подключение (LOGIN).
  - Объектные привилегии: только чтение (SELECT) данных из таблиц.

Вложенные роли упрощают управление доступом. Например, если роль reader включена в роль developer, все разработчики автоматически получают права на чтение.

*3. Создать роли и выдать им необходимые объектные и системные привилегии.*

SQL-код для создания ролей и назначения привилегий:

**1. Создание ролей:**

```
CREATE ROLE admin WITH LOGIN PASSWORD 'admin'  
CREATEDB CREATEROLE;  
CREATE ROLE developer WITH LOGIN PASSWORD 'dev';  
CREATE ROLE reader WITH LOGIN PASSWORD 'reader';
```

**2. Назначение вложенных ролей:**

```
GRANT reader TO developer;
```

**3. Назначение привилегий для схем:**

```
GRANT ALL PRIVILEGES ON SCHEMA series TO admin;  
GRANT ALL PRIVILEGES ON SCHEMA people TO admin;  
GRANT ALL PRIVILEGES ON SCHEMA relationships TO  
admin;
```

```
GRANT CREATE, USAGE ON SCHEMA series TO  
developer;  
GRANT CREATE, USAGE ON SCHEMA people TO  
developer;  
GRANT CREATE, USAGE ON SCHEMA relationships TO  
developer;
```

```
GRANT USAGE ON SCHEMA series TO reader;  
GRANT USAGE ON SCHEMA people TO reader;  
GRANT USAGE ON SCHEMA relationships TO reader;
```

**4. Назначение привилегий для таблиц:**

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA  
series TO admin;  
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA  
people TO admin;
```

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA
relationships TO admin;
```

```
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN
SCHEMA series TO developer;
```

```
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN
SCHEMA people TO developer;
```

```
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN
SCHEMA relationships TO developer;
```

```
GRANT SELECT ON ALL TABLES IN SCHEMA series TO
reader;
```

```
GRANT SELECT ON ALL TABLES IN SCHEMA people TO
reader;
```

```
GRANT SELECT ON ALL TABLES IN SCHEMA
relationships TO reader;
```

#### **5. Настройка привилегий для будущих объектов:**

```
ALTER DEFAULT PRIVILEGES IN SCHEMA series GRANT
ALL PRIVILEGES ON TABLES TO admin;
```

```
ALTER DEFAULT PRIVILEGES IN SCHEMA people GRANT
ALL PRIVILEGES ON TABLES TO admin;
```

```
ALTER DEFAULT PRIVILEGES IN SCHEMA relationships
GRANT ALL PRIVILEGES ON TABLES TO admin;
```

```
ALTER DEFAULT PRIVILEGES IN SCHEMA series GRANT
SELECT, INSERT, UPDATE ON TABLES TO developer;
```

```
ALTER DEFAULT PRIVILEGES IN SCHEMA people GRANT
SELECT, INSERT, UPDATE ON TABLES TO developer;
```

```
ALTER DEFAULT PRIVILEGES IN SCHEMA relationships
GRANT SELECT, INSERT, UPDATE ON TABLES TO
developer;
```

```
ALTER DEFAULT PRIVILEGES IN SCHEMA series GRANT
SELECT ON TABLES TO reader;
```

```
ALTER DEFAULT PRIVILEGES IN SCHEMA people GRANT
SELECT ON TABLES TO reader;
```

```
ALTER DEFAULT PRIVILEGES IN SCHEMA relationships
GRANT SELECT ON TABLES TO reader;
```

postgres/postgres@PostgreSQL 17

No limit

Query Query History

```
20 GRANT USAGE ON SCHEMA relationships TO reader;
21
22 -- 4. Назначение привилегий для таблиц:
23 GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA series TO admin;
24 GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA people TO admin;
25 GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA relationships TO admin;
26
27 GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA series TO developer;
28 GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA people TO developer;
29 GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA relationships TO developer;
30
31 GRANT SELECT ON ALL TABLES IN SCHEMA series TO reader;
32 GRANT SELECT ON ALL TABLES IN SCHEMA people TO reader;
33 GRANT SELECT ON ALL TABLES IN SCHEMA relationships TO reader;
34
35 -- 5. Настройка привилегий для будущих объектов:
36 ALTER DEFAULT PRIVILEGES IN SCHEMA series GRANT ALL PRIVILEGES ON TABLES TO admin;
37 ALTER DEFAULT PRIVILEGES IN SCHEMA people GRANT ALL PRIVILEGES ON TABLES TO admin;
38 ALTER DEFAULT PRIVILEGES IN SCHEMA relationships GRANT ALL PRIVILEGES ON TABLES TO admin;
39
40 ALTER DEFAULT PRIVILEGES IN SCHEMA series GRANT SELECT, INSERT, UPDATE ON TABLES TO developer;
41 ALTER DEFAULT PRIVILEGES IN SCHEMA people GRANT SELECT, INSERT, UPDATE ON TABLES TO developer;
42 ALTER DEFAULT PRIVILEGES IN SCHEMA relationships GRANT SELECT, INSERT, UPDATE ON TABLES TO developer;
43
44 ALTER DEFAULT PRIVILEGES IN SCHEMA series GRANT SELECT ON TABLES TO reader;
45 ALTER DEFAULT PRIVILEGES IN SCHEMA people GRANT SELECT ON TABLES TO reader;
46 ALTER DEFAULT PRIVILEGES IN SCHEMA relationships GRANT SELECT ON TABLES TO reader;
47
```

Data Output Messages Notifications

ALTER DEFAULT PRIVILEGES

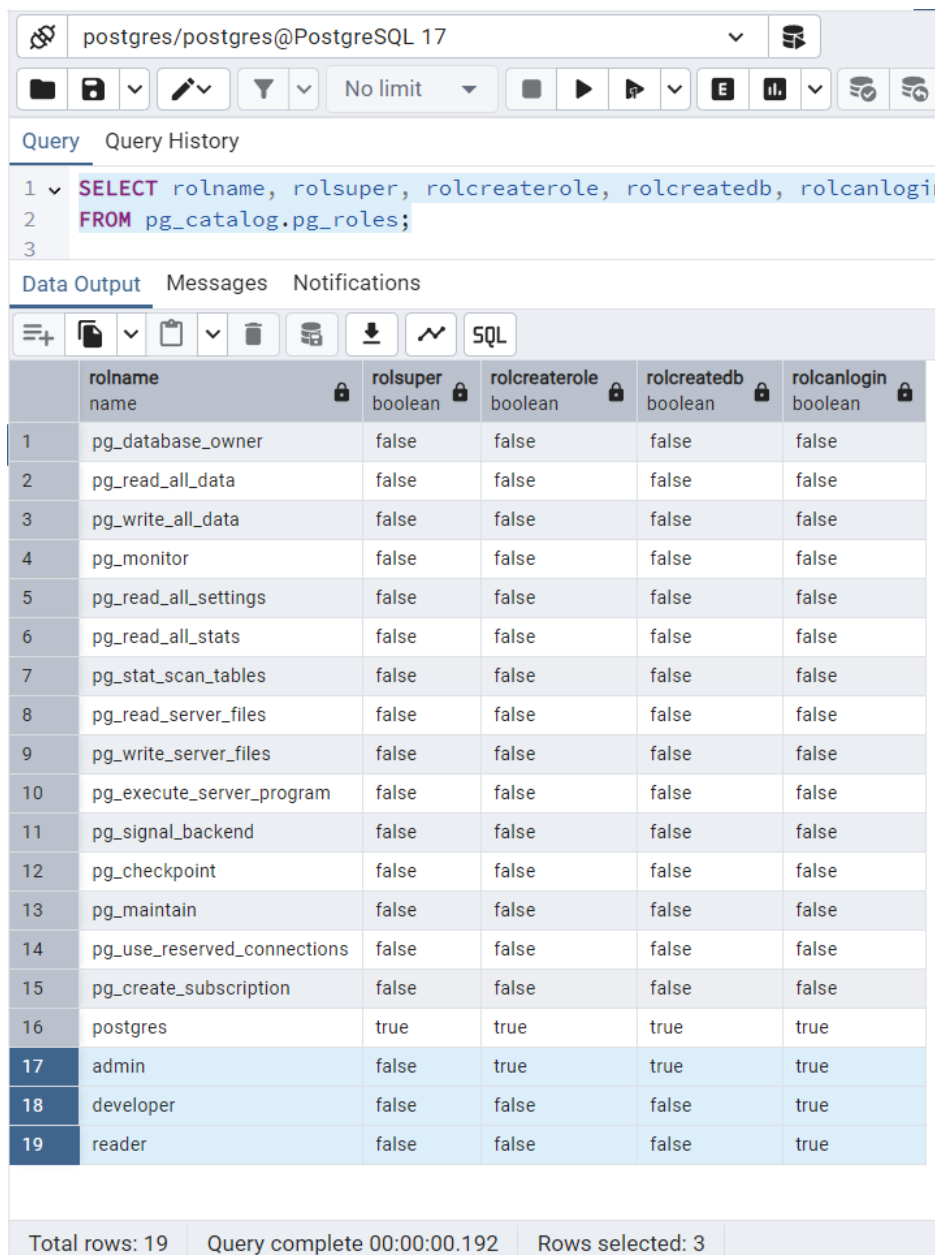
Query returned successfully in 32 msec.

Total rows: Query complete 00:00:00.032

4. Проверить по представлению системного каталога `pg_catalog.pg_roles`, что все нужные роли были созданы и обладают корректным набором привилегий.

Для проверки ролей используем SQL-запрос:

```
SELECT rolname, rolsuper, rolcreatorole, rolcreatedb,
rolcanlogin
FROM pg_catalog.pg_roles;
```



	rolname name	rolsuper boolean	rolcreatorole boolean	rolcreatedb boolean	rolcanlogin boolean
1	pg_database_owner	false	false	false	false
2	pg_read_all_data	false	false	false	false
3	pg_write_all_data	false	false	false	false
4	pg_monitor	false	false	false	false
5	pg_read_all_settings	false	false	false	false
6	pg_read_all_stats	false	false	false	false
7	pg_stat_scan_tables	false	false	false	false
8	pg_read_server_files	false	false	false	false
9	pg_write_server_files	false	false	false	false
10	pg_execute_server_program	false	false	false	false
11	pg_signal_backend	false	false	false	false
12	pg_checkpoint	false	false	false	false
13	pg_maintain	false	false	false	false
14	pg_use_reserved_connections	false	false	false	false
15	pg_create_subscription	false	false	false	false
16	postgres	true	true	true	true
17	admin	false	true	true	true
18	developer	false	false	false	true
19	reader	false	false	false	true
Total rows: 19    Query complete 00:00:00.192    Rows selected: 3					



Запрос отобразил список ролей и их привилегии. Видно наличие созданных ролей: admin, developer и reader, а также, что флаг rolcanlogin установлен для всех ролей.

Проверка привилегий на схемы:

```
\dn+ series
```

```
postgres=# \dn+ series
```

List of schemas			
Name	Owner	Access privileges	Description
series	postgres	postgres=UC/postgres +  admin=UC/postgres +  developer=UC/postgres+  reader=U/postgres	

(1 row)

```
postgres=# \dn+ people
```

List of schemas			
Name	Owner	Access privileges	Description
people	postgres	postgres=UC/postgres +  admin=UC/postgres +  developer=UC/postgres+  reader=U/postgres	

(1 row)

```
postgres=# \dn+ relationships
```

List of schemas			
Name	Owner	Access privileges	Description
relationships	postgres	postgres=UC/postgres +  admin=UC/postgres +  developer=UC/postgres+  reader=U/postgres	

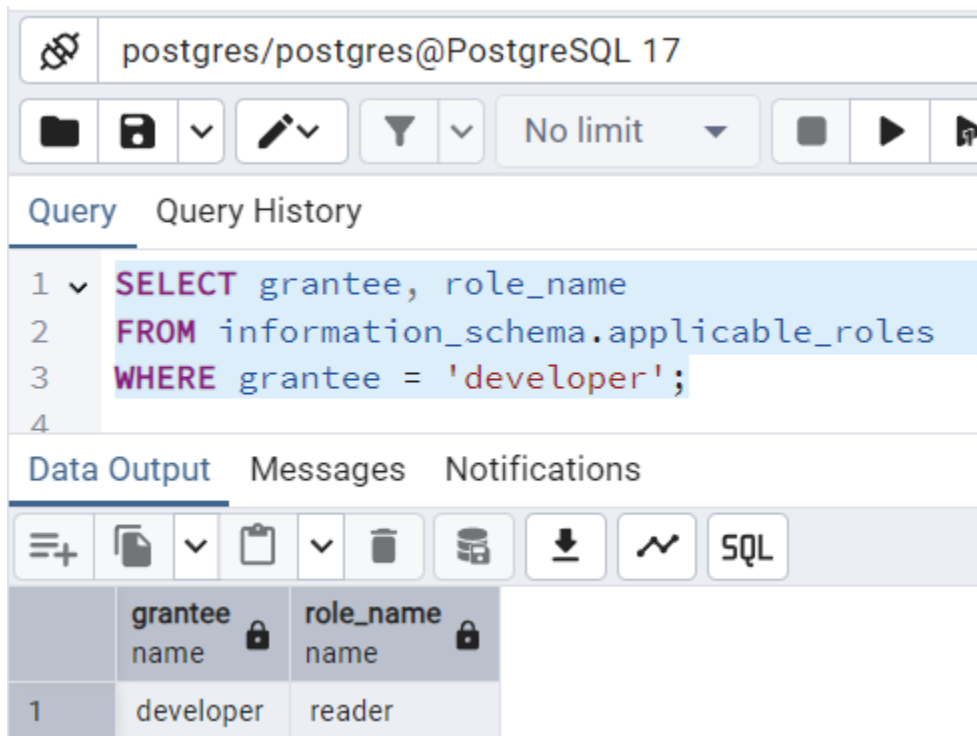
(1 row)

```
postgres=#
```

Из вывода команды видно, что ко всем схемам все роли имеют привилегию USAGE (доступ к объектам, не разрешает создание объектов), а также все кроме reader имеют привилегию CREATE (создание новых объектов).

Проверим наследование привилегий:

```
SELECT grantee, role_name
FROM information_schema.applicable_roles
WHERE grantee = 'developer';
```



The screenshot shows a PostgreSQL client interface. At the top, the connection is labeled 'postgres/postgres@PostgreSQL 17'. Below this is a toolbar with icons for file operations, a dropdown menu, a filter icon, a 'No limit' button, and execution controls. The main area is divided into 'Query' and 'Query History' tabs. The 'Query' tab is active, showing a SQL query with line numbers 1 through 4. Below the query is another toolbar with icons for data output, messages, notifications, and a 'SQL' button. The 'Data Output' tab is active, displaying a table with two columns: 'grantee' and 'role\_name'. The table has one row with the values 'developer' and 'reader'.

	grantee name	role_name name
1	developer	reader

Проверим привилегии по каждому таблицам для роли администратора:

```
SELECT grantee, privilege_type, table_schema,  
table_name  
FROM information_schema.role_table_grants  
WHERE grantee = 'admin';
```

Проверим привилегии по каждому таблицам для роли разработчика:

```
SELECT grantee, privilege_type, table_schema,  
table_name  
FROM information_schema.role_table_grants  
WHERE grantee = 'developer';
```

Проверим привилегии по каждому таблицам для роли reader:

```
SELECT grantee, privilege_type, table_schema,  
table_name  
FROM information_schema.role_table_grants  
WHERE grantee = 'reader';
```

postgres/postgres@PostgreSQL 17

No limit

Query

Query History

```

1 SELECT grantee, privilege_type, table_schema, table_name
2 FROM information_schema.role_table_grants
3 WHERE grantee = 'admin';
4

```

Data Output

Messages

Notifications

SQL

	grantee name	privilege_type character varying	table_schema name	table_name name
23	admin	SELECT	series	reviews
24	admin	UPDATE	series	reviews
25	admin	DELETE	series	reviews
26	admin	TRUNCATE	series	reviews
27	admin	REFERENCES	series	reviews
28	admin	TRIGGER	series	reviews
29	admin	INSERT	people	actors
30	admin	SELECT	people	actors
31	admin	UPDATE	people	actors
32	admin	DELETE	people	actors
33	admin	TRUNCATE	people	actors
34	admin	REFERENCES	people	actors
35	admin	TRIGGER	people	actors
36	admin	INSERT	relationships	cast
37	admin	SELECT	relationships	cast
38	admin	UPDATE	relationships	cast
39	admin	DELETE	relationships	cast
40	admin	TRUNCATE	relationships	cast
41	admin	REFERENCES	relationships	cast
42	admin	TRIGGER	relationships	cast

Total rows: 42

Query complete 00:00:00.102

postgres/postgres@PostgreSQL 17

📁
💾
✎
🔍
No limit
📄
▶
📌
E
📊

Query
Query History

```

1 SELECT grantee, privilege_type, table_schema, table_name
2 FROM information_schema.role_table_grants
3 WHERE grantee = 'developer';
4

```

Data Output
Messages
Notifications

≡
📄
📋
🗑
🗑
📄
⬇
📈
SQL

	grantee name	privilege_type character varying	table_schema name	table_name name
1	developer	INSERT	series	seasons
2	developer	SELECT	series	seasons
3	developer	UPDATE	series	seasons
4	developer	INSERT	series	shows
5	developer	SELECT	series	shows
6	developer	UPDATE	series	shows
7	developer	INSERT	series	episodes
8	developer	SELECT	series	episodes
9	developer	UPDATE	series	episodes
10	developer	INSERT	series	reviews
11	developer	SELECT	series	reviews
12	developer	UPDATE	series	reviews
13	developer	INSERT	people	actors
14	developer	SELECT	people	actors
15	developer	UPDATE	people	actors
16	developer	INSERT	relationships	cast
17	developer	SELECT	relationships	cast
18	developer	UPDATE	relationships	cast



5. Попробовать подключиться от лица каждой роли и проверить доступ.

Запросы:

1. От лица admin:

```
SELECT * FROM series.shows;  
INSERT INTO series.shows (title) VALUES ('New  
Show');
```

2. От лица developer:

```
SELECT * FROM series.shows;  
INSERT INTO series.shows (title) VALUES ('Dev  
Show');
```

3. От лица reader:

```
SELECT * FROM series.shows;  
-- Проверка: ожидается ошибка.  
INSERT INTO series.shows (title) VALUES ('Test  
Show');
```

При попытке выполнить запрос вставки от лица администратора и разработчика высветилась ошибка, суть которой – отсутствие доступа к последовательности, с помощью которой работает автоинкремент индекса. Проблема решилась наделением соответствующих ролей необходимыми привилегиями:

```
GRANT USAGE, SELECT, UPDATE ON SEQUENCE  
series.shows_id_seq TO admin;
```

```
GRANT USAGE, SELECT, UPDATE ON SEQUENCE  
series.shows_id_seq TO developer;
```

PS C:\Users\Katehok> psql -U admin -d postgres

Пароль пользователя admin:

psql (17.2)

ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной страницы windows (1251).  
8-битовые (русские) символы могут отображаться некорректно.  
Подробнее об этом смотрите документацию psql, раздел "Notes for windows users".

Введите "help", чтобы получить справку.

postgres=> SELECT \* FROM series.shows;

id	title	year	budget	synopsis
1	Game of Thrones			
2	Breaking Bad			
3	Stranger Things			
4	Friends			
5	The Mandalorian			

(5 стр.)

postgres=> INSERT INTO series.shows (title) VALUES ('New Show');

ОШИБКА: нет доступа к последовательности shows\_id\_seq

postgres=> GRANT USAGE, SELECT, UPDATE ON SEQUENCE series.shows\_id\_seq TO admin;

ОШИБКА: нет доступа к последовательности shows\_id\_seq

postgres=> exit

PS C:\Users\Katehok> psql -U postgres -d postgres

Пароль пользователя postgres:

psql (17.2)

ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной страницы windows (1251).  
8-битовые (русские) символы могут отображаться некорректно.  
Подробнее об этом смотрите документацию psql, раздел "Notes for windows users".

Введите "help", чтобы получить справку.

postgres=# GRANT USAGE, SELECT, UPDATE ON SEQUENCE series.shows\_id\_seq TO admin;  
GRANT

postgres=# exit

PS C:\Users\Katehok> psql -U admin -d postgres

Пароль пользователя admin:

psql (17.2)

ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной страницы windows (1251).  
8-битовые (русские) символы могут отображаться некорректно.  
Подробнее об этом смотрите документацию psql, раздел "Notes for windows users".

Введите "help", чтобы получить справку.

postgres=> INSERT INTO series.shows (title) VALUES ('New Show');

INSERT 0 1

postgres=> SELECT \* FROM series.shows;

id	title	year	budget	synopsis
1	Game of Thrones			
2	Breaking Bad			
3	Stranger Things			
4	Friends			
5	The Mandalorian			
6	New Show			

(6 стр.)



```
PS C:\Users\Katehok> psql -U postgres -d postgres
```

```
Пароль пользователя postgres:
```

```
psql (17.2)
```

```
ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной
страницы Windows (1251).
8-битовые (русские) символы могут отображаться некорректно.
Подробнее об этом смотрите документацию psql, раздел
"Notes for Windows users".
```

```
Введите "help", чтобы получить справку.
```

```
postgres=# GRANT USAGE, SELECT, UPDATE ON SEQUENCE series.shows_id_seq TO developer;
GRANT
```

```
postgres=# exit
```

```
PS C:\Users\Katehok>
```

```
PS C:\Users\Katehok> psql -U developer -d postgres
```

```
Пароль пользователя developer:
```

```
psql (17.2)
```

```
ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной
страницы Windows (1251).
8-битовые (русские) символы могут отображаться некорректно.
Подробнее об этом смотрите документацию psql, раздел
"Notes for Windows users".
```

```
Введите "help", чтобы получить справку.
```

```
postgres=>
```

```
postgres=> SELECT * FROM series.shows;
```

id	title	year	budget	synopsis
1	Game of Thrones			
2	Breaking Bad			
3	Stranger Things			
4	Friends			
5	The Mandalorian			
6	New Show			

```
(6 строк)
```

```
postgres=> INSERT INTO series.shows (title) VALUES ('Dev Show');
```

```
INSERT 0 1
```

```
postgres=>
```

```
postgres=> SELECT * FROM series.shows;
```

id	title	year	budget	synopsis
1	Game of Thrones			
2	Breaking Bad			
3	Stranger Things			
4	Friends			
5	The Mandalorian			
6	New Show			
7	Dev Show			

```
(7 строк)
```

```
postgres=> _
```

```
PS C:\Users\Katehok> psql -U reader -d postgres
```

Пароль пользователя reader:

psql (17.2)

**ПРЕДУПРЕЖДЕНИЕ:** Кодовая страница консоли (866) отличается от основной страницы Windows (1251).  
8-битовые (русские) символы могут отображаться некорректно.  
Подробнее об этом смотрите документацию `psql`, раздел "Notes for Windows users".

Введите "help", чтобы получить справку.

```
postgres=>
```

```
postgres=> SELECT * FROM series.shows;
```

id	title	year	budget	synopsis
1	Game of Thrones			
2	Breaking Bad			
3	Stranger Things			
4	Friends			
5	The Mandalorian			
6	New Show			
7	Dev Show			

(7 ёёЁюъ)

```
postgres=>
```

```
postgres=> -- Проверка: ожидается ошибка.
```

```
postgres=>
```

```
postgres=> INSERT INTO series.shows (title) VALUES ('Test Show');
```

ОШИБКА: нет доступа к таблице shows

```
postgres=>
```

```
postgres=> SELECT * FROM series.shows;
```

id	title	year	budget	synopsis
1	Game of Thrones			
2	Breaking Bad			
3	Stranger Things			
4	Friends			
5	The Mandalorian			
6	New Show			
7	Dev Show			

(7 ёёЁёююь)

```
postgres=> _
```

## Заключение

В ходе работы в соответствии с предметной областью была создана и заполнена база данных сериалов, к ней было выполнено несколько составных запросов. Также было проведено сравнение PostgreSQL и SQLite.

## Приложение

- директория последней лабы из репозитория предыдущего семестра;
- ERD структуры таблиц;
- отчет (docx);
- отчет (pdf).