

Национальный исследовательский ядерный университет «МИФИ»

(Московский Инженерно-Физический Институт)

Кафедра №42 «Криптология и кибербезопасность»

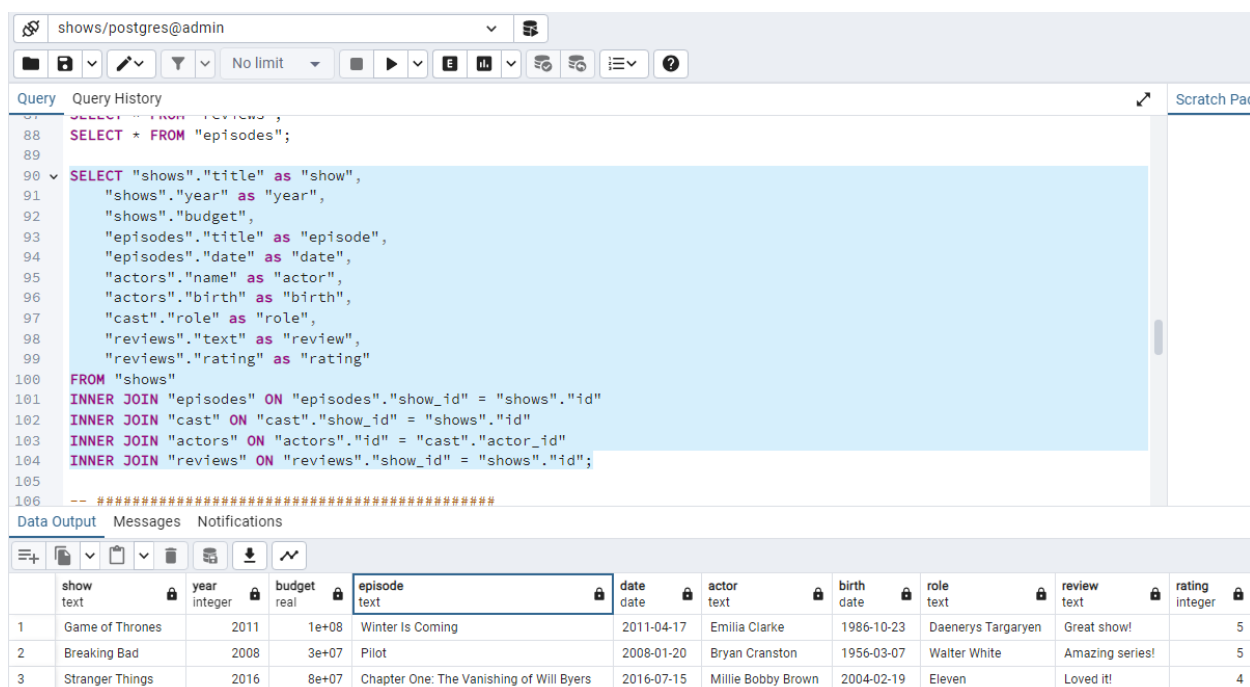
Лабораторная работа №4
«Переход на PostgreSQL»

Тимин Александр Б21-515 (2024г.)

1 Заполнение

Учитывая специфику предметной области и возможности PostgreSQL создадим базу данных и необходимый набор таблиц (листинг запросов приведен в разделе [«Приложение»](#)).

Также заполним таблицы информацией о нескольких известных сериалах (листинг запросов приведен в разделе [«Приложение»](#)). На рисунке 1.1 приведена часть данных, которые в дальнейшем будут использоваться в запросах.



The screenshot shows the pgAdmin 4 interface. The top toolbar includes icons for file operations, query execution, and settings. The 'Query' tab is active, displaying a SQL query that joins several tables: 'shows', 'episodes', 'cast', 'actors', and 'reviews'. The query selects various fields and uses INNER JOINs to link the data. Below the query editor, the 'Data Output' tab shows the results of the query in a table format. The table has 11 columns: show, year, budget, episode, date, actor, birth, role, review, and rating. Three rows of data are visible, representing different TV shows and their episodes.

	show	year	budget	episode	date	actor	birth	role	review	rating
	text	integer	real	text	date	text	date	text	text	integer
1	Game of Thrones	2011	1e+08	Winter Is Coming	2011-04-17	Emilia Clarke	1986-10-23	Daenerys Targaryen	Great show!	5
2	Breaking Bad	2008	3e+07	Pilot	2008-01-20	Bryan Cranston	1956-03-07	Walter White	Amazing series!	5
3	Stranger Things	2016	8e+07	Chapter One: The Vanishing of Will Byers	2016-07-15	Millie Bobby Brown	2004-02-19	Eleven	Loved it!	4

Рисунок 1.1 - данные.

Здесь и далее при демонстрации базы данных используется GUI pgAdmin 4.

2 Запросы

Предположим, что сотрудники компании, которая предоставляет информацию о сериалах, хотят знать ответы на следующие вопросы:

1. Какие актеры снялись в сериале "Game of Thrones"?
2. Какие сериалы были выпущены после 2010 года и их бюджеты?
3. Какие отзывы получили сериалы с рейтингом выше 4?
4. Какова общая сумма бюджета всех сериалов?
5. Какие актеры родились после 1980 года?
6. Какие сериалы имели эпизоды, вышедшие после 2015 года?
7. Какие актеры сыграли роли в сериале с наивысшим бюджетом?
8. Сколько эпизодов было в каждом сериале?

На рисунках 1-8 представлены соответствующие запросы к базе данных и ответы на них.

Ссылка на соответствующий листинг запросов приведен в разделе [«Приложение»](#).

shows/postgres@admin

Query Query History

```
88 SELECT * FROM "episodes";
89
90 -- #####
91
92 -- 1. Какие актеры снялись в сериале "Game of Thrones"?
93 SELECT "actors"."name"
94 FROM "actors"
95 INNER JOIN "cast" ON "actors"."id" = "cast"."actor_id"
96 INNER JOIN "shows" ON "cast"."show_id" = "shows"."id"
97 WHERE "shows"."title" = 'Game of Thrones';
```

Data Output Messages Notifications

	name text
1	Emilia Clarke

Рисунок 2.1 - получение списка актеров, снявшихся в сериале "Game of Thrones".

shows/postgres@admin

No limit

Query Query History

93

SELECT actors.name

94

FROM "actors"

95

INNER JOIN "cast" ON "actors"."id" = "cast"."actor_id"

96

INNER JOIN "shows" ON "cast"."show_id" = "shows"."id"

97

WHERE "shows"."title" = 'Game of Thrones';

98

99

-- 2. Какие сериалы были выпущены после 2010 года и их бюджеты?

100

SELECT "title", "budget"

101

FROM "shows"

102

WHERE "year" > 2010;

103

Data Output Messages Notifications

	title text	budget real
1	Game of Thron...	1e+08
2	Stranger Things	8e+07

Рисунок 2.2 - получение списка сериалов, выпущенных после 2010 года и их бюджеты.

shows/postgres@admin

Query Query History

```

102 WHERE "year" > 2010;
103
104 -- 3. Какие отзывы получили сериалы с рейтингом выше 4?
105 SELECT "shows"."title", "reviews"."reviewer", "reviews"."text", "reviews"."rating"
106 FROM "reviews"
107 INNER JOIN "shows" ON "reviews"."show_id" = "shows"."id"
108 WHERE "reviews"."rating" > 4;
109
110 -- 4. Какова общая сумма бюджета всех сериалов?
111 SELECT SUM("budget") AS "totalbudget"

```

Data Output Messages Notifications

	title text	reviewer text	text text	rating integer
1	Game of Thrones	John Doe	Great show!	5
2	Breaking Bad	Jane Smith	Amazing series!	5

Рисунок 2.3 - получение отзывов, полученных сериалами с рейтингом выше 4.

shows/postgres@admin

Query Query History

```

107 INNER JOIN "shows" ON "reviews"."show_id" = "shows"."id"
108 WHERE "reviews"."rating" > 4;
109
110 -- 4. Какова общая сумма бюджета всех сериалов?
111 SELECT SUM("budget") AS "totalbudget"
112 FROM "shows";
113
114 -- 5. Какие актеры родились после 1980 года?
115 SELECT "name"
116 FROM "actors"

```

Data Output Messages Notifications

	totalbudget real
1	2.1e+08

Рисунок 2.4 - получение суммы бюджета всех сериалов.

shows/postgres@admin

No limit

E

ll.

Query Query History

112 FROM "shows";

113

114 -- 5. Какие актеры родились после 1980 года?

115 SELECT "name"

116 FROM "actors"

117 WHERE "birth" > '1980-01-01';

118

119 -- 6. Какие сериалы имели эпизоды, вышедшие после 2015 года?

120 SELECT DISTINCT "shows"."title"

121 FROM "shows"

122

Data Output Messages Notifications

	name	
	text	
1	Emilia Clarke	
2	Millie Bobby Brown	

Рисунок 2.5 - получение списка актеров, рожденных после 1980 года.

shows/postgres@admin

Query Query History

```

117 WHERE "date" > '2015-01-01';
118
119 -- 6. Какие сериалы имели эпизоды, вышедшие после 2015 года?
120 SELECT DISTINCT "shows"."title"
121 FROM "shows"
122 INNER JOIN "episodes" ON "shows"."id" = "episodes"."show_id"
123 WHERE "episodes"."date" > '2015-01-01';
124
125 -- 7. Какие актеры сыграли роли в сериале с наивысшим бюджетом?
126 SELECT "actors"."name", "cast"."role"
127 FROM "actors"

```

Data Output Messages Notifications

	title
1	Stranger Things

Рисунок 2.6 - получение списка сериалов, имеющих эпизоды, вышедшие после 2015 года.

shows/postgres@admin

Query Query History

```

123 WHERE "episodes"."date" > '2015-01-01';
124
125 -- 7. Какие актеры сыграли роли в сериале с наивысшим бюджетом?
126 SELECT "actors"."name", "cast"."role"
127 FROM "actors"
128 INNER JOIN "cast" ON "actors"."id" = "cast"."actor_id"
129 INNER JOIN "shows" ON "cast"."show_id" = "shows"."id"
130 WHERE "shows"."budget" = (SELECT MAX("budget") FROM "shows");
131
132 -- 8. Сколько эпизодов было в каждом сериале?

```

Data Output Messages Notifications

	name	role
1	Emilia Clarke	Daenerys Targaryen

Рисунок 2.7 – получение списка актеров, сыгравших роли в сериале с наивысшим бюджетом.

shows/postgres@admin

📁 💾 ✎ 🔍 No limit ⏏ ▶ E 📊 🔄 🔄 ☰ ?

Query Query History

```

128 INNER JOIN "cast" ON "actors"."id" = "cast"."actor_id"
129 INNER JOIN "shows" ON "cast"."show_id" = "shows"."id"
130 WHERE "shows"."budget" = (SELECT MAX("budget") FROM "shows");
131
132 -- 8. Сколько эпизодов было в каждом сериале?
133 SELECT "shows"."title", COUNT("episodes"."id") AS "episodecount"
134 FROM "shows"
135 INNER JOIN "episodes" ON "shows"."id" = "episodes"."show_id"
136 GROUP BY "shows"."title";
137

```

Data Output Messages Notifications

☰ 📄 📋 🗑 📊 ⬇ 📈

	title text	episodecount bigint
1	Stranger Things	1
2	Breaking Bad	1
3	Game of Thrones	1

Рисунок 2.8 – получение количества эпизодов в каждом сериале.

3 Сравнение PostgreSQL и SQLite

1. *Жесткая типизация данных.* PostgreSQL не позволяет хранить в одном столбце данные разных типов. SQLite позволяет хранить данные разных типов в одном столбце.
2. *Поддержка сложных запросов.* PostgreSQL поддерживает более сложные запросы и функции, такие как оконные функции и рекурсивные CTE.
3. *Совместный доступ к базе данных.* PostgreSQL поддерживает многопользовательский доступ и управление ролями, в то время как SQLite больше ориентирована на встраиваемые приложения и поддерживает только сериализованный доступ.
4. *COLLATION.* PostgreSQL поддерживает колляции и позволяет задавать порядок сортировки строк в соответствии с выбранной локалью, что позволяет корректно сортировать строки на различных языках. SQLite ограничивается побайтовой сортировкой, что не всегда корректно для всех языков.
5. *Транзакции.* PostgreSQL поддерживает сложные транзакции с возможностью отката (ROLLBACK) и фиксации (COMMIT) в рамках транзакционных блоков, что позволяет обеспечивать целостность данных на высоком уровне. SQLite также поддерживает транзакции, но имеет меньше возможностей для их настройки и управления.
6. *Агрегатные функции и оконные функции.* PostgreSQL поддерживает широкий набор агрегатных и оконных функций, которые могут быть использованы для анализа данных, таких как вычисление скользящего среднего, ранжирование и т.д. SQLite поддерживает ограниченный набор таких функций.
7. *Работа с датами и временем.* PostgreSQL имеет мощные встроенные функции для работы с датами и временем, включая поддержку часовых поясов, интервалов и сложных операций с датами. SQLite имеет ограниченные возможности для работы с датами и временем.

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.

На защиту

На рисунке 1 – скриншот сообщения с заданиями на защиту.

Re: БСБД Лаба 4

- Михаил Куприяшин Вчера, 2:41
Кому: вам



На защиту: для каждого сериала предскажите дату выхода следующего сезона. Если у сериала есть минимум два сезона, предполагайте, что они выходят "линейно". Если у сериала один сезон, выведите NULL

Рисунок 1 - задания на защиту.

Для начала необходимо обновить структуру таблиц. На рисунке 2 – ERD новой структуры.

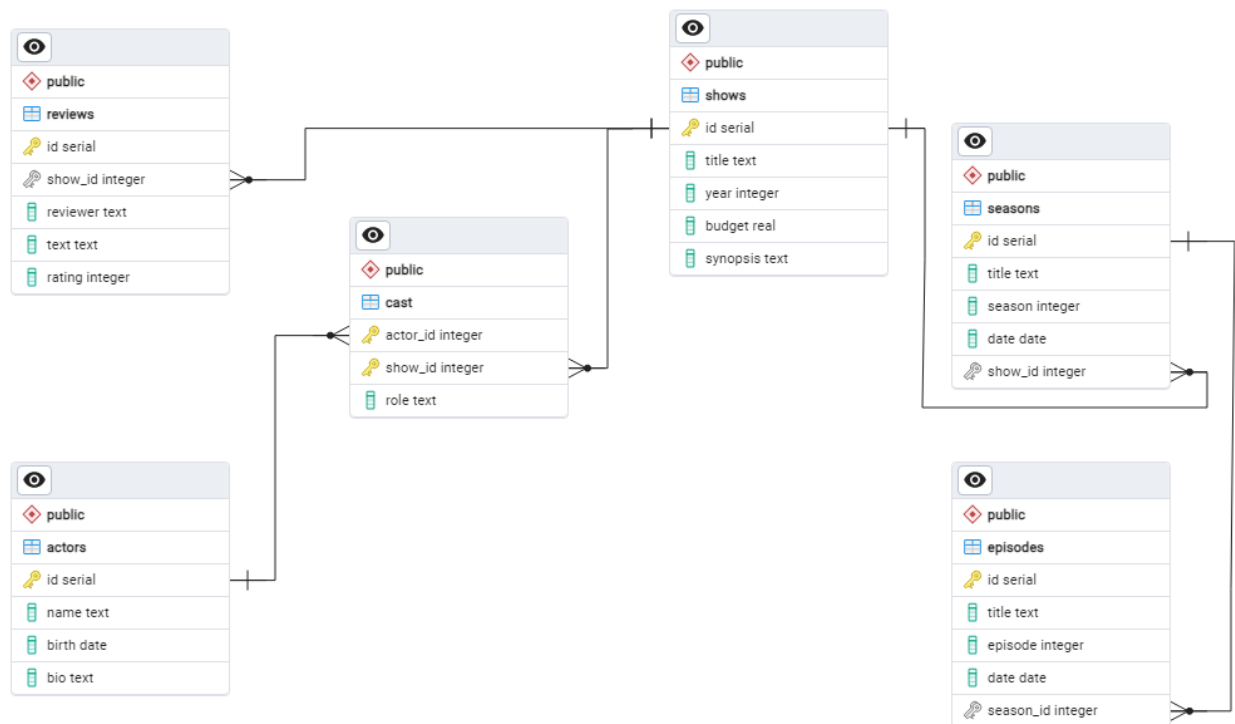
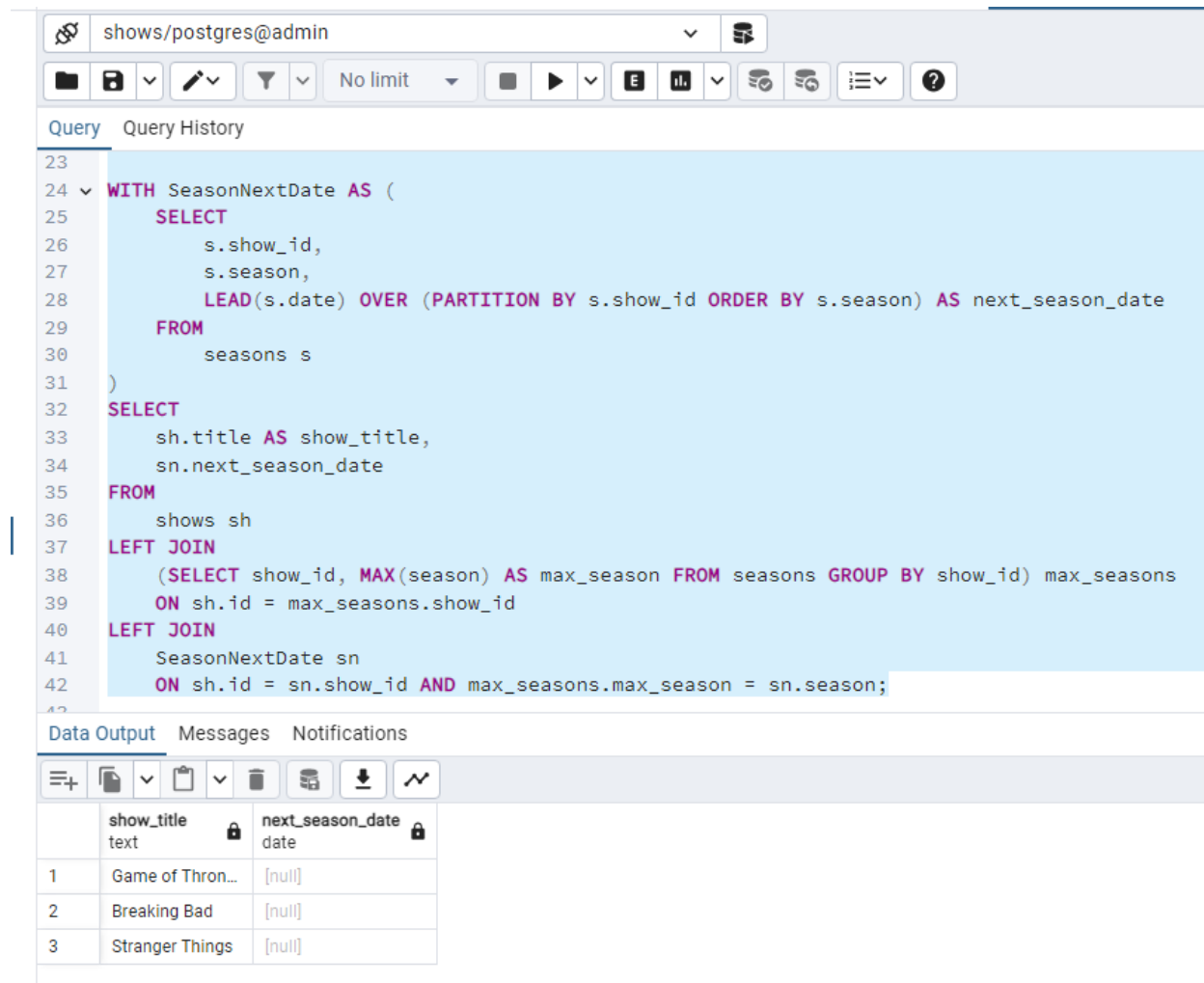


Рисунок 2 - ERD обновленной структуры таблиц.

Основная идея запроса – использование функции LEAD. Оконная функция LEAD позволяет нам получить значение следующей строки в упорядоченном наборе данных. На рисунке 3 – данный запрос и ответ на него.

Ссылка на соответствующие листинг запросов приведен в разделе [«Приложение»](#).



The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
23
24 WITH SeasonNextDate AS (
25     SELECT
26         s.show_id,
27         s.season,
28         LEAD(s.date) OVER (PARTITION BY s.show_id ORDER BY s.season) AS next_season_date
29     FROM
30         seasons s
31 )
32 SELECT
33     sh.title AS show_title,
34     sn.next_season_date
35 FROM
36     shows sh
37 LEFT JOIN
38     (SELECT show_id, MAX(season) AS max_season FROM seasons GROUP BY show_id) max_seasons
39 ON sh.id = max_seasons.show_id
40 LEFT JOIN
41     SeasonNextDate sn
42 ON sh.id = sn.show_id AND max_seasons.max_season = sn.season;
```

The results are displayed in a table with columns 'show_title' and 'next_season_date'.

	show_title	next_season_date
1	Game of Thron...	[null]
2	Breaking Bad	[null]
3	Stranger Things	[null]

Рисунок 3 - определение даты выхода следующего сезона.

На защиту

На рисунке 4 – скриншот сообщения с заданиями на защиту.

Re: Fwd: Re: БСБД Лаба 4

•

M

Михаил Куприяшин

Сегодня, 17:36

Кому: вам

🔖

🔗

📅

🖨

...

LEAD позволяет получить следующую строку в наборе данных, если она *есть*. Если её нет, СУБД её за Вас не придумает xD

Тут нужна линейная аппроксимация.

Рисунок 4 - задание на защиту.

Не без помощи всем известной нейросети вспомнил, как делать линейную аппроксимацию.

Изменил данные в таблицах (так как в порыве научных изысканий, я там что-то сломал).

Касаемо самого запроса данных: воспользовавшись каскадом временных таблиц и объединением двух запросов сконструировал квазимодо, результат которого представлен на рисунке 5.

Data Output				Messages	Notifications
	title	next_season	predicted_date		
	text	integer	text		
1	Stranger Things	4	2020-12-02 16:00:00		
2	Game of Thrones	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		

Рисунок 5 - результат запроса.

Приложение

- [SQL-скрипт создания таблиц;](#)
- [SQL-скрипт заполнения таблиц;](#)
- [SQL-скрипт обновления структуры таблицы;](#)
- [SQL-скрипт получения данных из таблиц;](#)
- [ERD обновленной структуры таблиц;](#)
- [обновленный отчет \(docx\);](#)
- [обновленный отчет \(pdf\).](#)