

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет инфокоммуникационных технологий

Лабораторная работа №2
«Запросы на выборку и модификацию
данных, представления и индексы в
PostgreSQL»
по дисциплине:
«Проектирование и реализация баз
данных»

Выполнила:
студентка II курса ИКТ
группы К3242
Скокова Алина Викторовна

Проверила:
Говорова Марина Михайловна

Санкт-Петербург
2022

Цель работы: овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

Практическое задание.

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию, часть 2 и 3).
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
3. Изучить графическое представление запросов и просмотреть историю запросов
4. Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

Схема базы данных.

Наименование БД – «telephone_exchange_db».

Схема логической модели базы данных, сгенерированная в Generate ERD, представлена на рисунке 1.

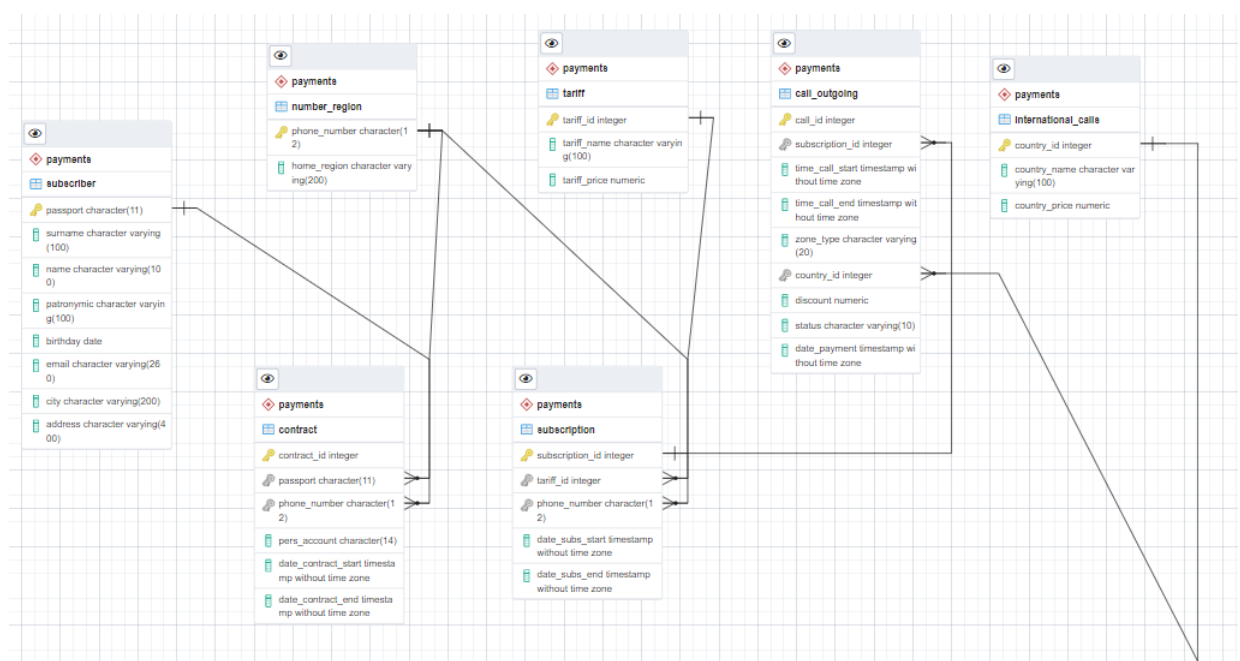


Рисунок 1 – Схема логической модели БД

Выполнение.

Запросы к базе данных:

1. Вывести суммарное время переговоров каждого абонента за заданный период (рис. 2).

```
select passport, sum(time_call_end - time_call_start) as duration_sum
from (payments.call_outgoing natural join payments.subscription) natural join
payments.contract
where date_part('month', time_call_start) in (8, 9)
group by passport
union
select distinct passport, interval '00:00:00'
from payments.contract
where passport not in (select passport from (payments.call_outgoing natural join
payments.subscription) natural join payments.contract where date_part('month',
time_call_start) in (8, 9));
```

	passport character (11)	duration_sum interval
1	1111 111111	00:20:00
2	2232 227422	00:00:00
3	6666 648381	00:00:00
4	8543 123456	00:00:00
5	4900 836477	00:03:00
6	7271 000100	00:28:00
7	9000 111112	00:25:00
8	3668 208482	00:00:00
9	1010 674583	00:00:00
10	5116 343435	00:20:00

Рисунок 2 – Результат выполнения запроса 1

2. Найти среднюю продолжительность разговора абонента АТС (рис. 3).

```
select avg(time_call_end - time_call_start) as duration_avg
from payments.call_outgoing;
```

	duration_avg interval
1	00:21:44

Рисунок 3– Результат выполнения запроса 2

3. Вывести количество междугородных переговоров каждого абонента (рис. 4).

```
select passport, count(*) as intercity_num
from (payments.call_outgoing natural join payments.subscription) natural join
payments.contract
where zone_type = 'страна'
group by passport
union
select distinct passport, 0
from payments.contract
where passport not in (select passport from (payments.call_outgoing natural join
payments.subscription) natural join payments.contract where zone_type = 'страна')
order by intercity_num desc;
```

	passport character (11)	intercity_num bigint
1	1111 111111	3
2	3668 208482	2
3	4900 836477	1
4	8543 123456	0
5	6666 648381	0
6	2232 227422	0
7	9000 111112	0
8	1010 674583	0
9	7271 000100	0
10	5116 343435	0

Рисунок 4 – Результат выполнения запроса 3

4. Вывести список абонентов, не внёсших оплату за прошедший месяц (рис. 5).

```
select distinct passport
from (payments.call_outgoing natural join payments.subscription) natural join
payments.contract
where status = 'ожидание' and date_part('year', time_call_start) = date_part('year',
current_timestamp) and date_part('month', time_call_start) = date_part('month',
current_timestamp);
```

	passport character (11)
1	3668 208482
2	5116 343435

Рисунок 5 – Результат выполнения запроса 4

5. Сколько звонков было сделано в каждый из следующих городов: в Москву, Лондон, Париж (рис. 6).

Вместо городов были выбраны страны: Армения, Казахстан и Финляндия. Замена была сделана для соответствия существующей структуре таблицы базы данных.

```
select country_name, count(call_id) as calls_num
from payments.call_outgoing natural right join payments.international_calls
group by country_name
having country_name in ('Армения', 'Казахстан', 'Финляндия');
```

	country_name character varying (100)	calls_num bigint
1	Армения	3
2	Казахстан	1
3	Финляндия	0

Рисунок 6 – Результат выполнения запроса 5

6. Вывести список абонентов, звонивших только в ночное время (рис. 7).

```
select distinct passport
from (payments.call_outgoing natural join payments.subscription) natural join
payments.contract
where (date_part('hour', time_call_start) > 19 or date_part('hour', time_call_start) <
5) and passport not in (select passport from (payments.call_outgoing natural join
```

payments.subscription) natural join payments.contract where date_part('hour', time_call_start) between 5 and 19);

	passport character (11)
1	5116 343435

Рисунок 7 – Результат выполнения запроса 6

7. Вывести список абонентов, время разговоров которых превышает среднее для этой же зоны (рис. 8).

```
select distinct passport
from ((payments.call_outgoing natural join payments.subscription) natural join
payments.contract) natural join (select zone_type, avg(time_call_end -
time_call_start) as duration_mean from payments.call_outgoing group by
zone_type) as t
where (time_call_end - time_call_start) > t.duration_mean
```

	passport character (11)
1	9000 111112
2	5116 343435
3	1111 111111
4	7271 000100
5	3668 208482
6	1010 674583

Рисунок 8 – Результат выполнения запроса 7

Представления:

1. Содержащее сведения обо всех абонентах и их переговорах за прошедший месяц (рис. 9, 10).

```
create view payments.prev_month_view as
select passport, surname, name, patronymic, birthday, email, city, address,
pers_account, date_contract_start, date_contract_end, phone_number,
home_region, tariff_name, tariff_price, date_subs_start, date_subs_end,
time_call_start, time_call_end, discount, zone_type, country_name,
country_price, status, date_payment
```

```

from (((((payments.subscriber natural join payments.contract)
      natural join payments.number_region) natural join payments.subscription)
      natural join payments.tariff) natural left join payments.call_outgoing)
      natural left join payments.international_calls as t
where date_part('year', time_call_start) = date_part('year', current_timestamp)
and date_part('month', time_call_start) = date_part('month',
current_timestamp);

```

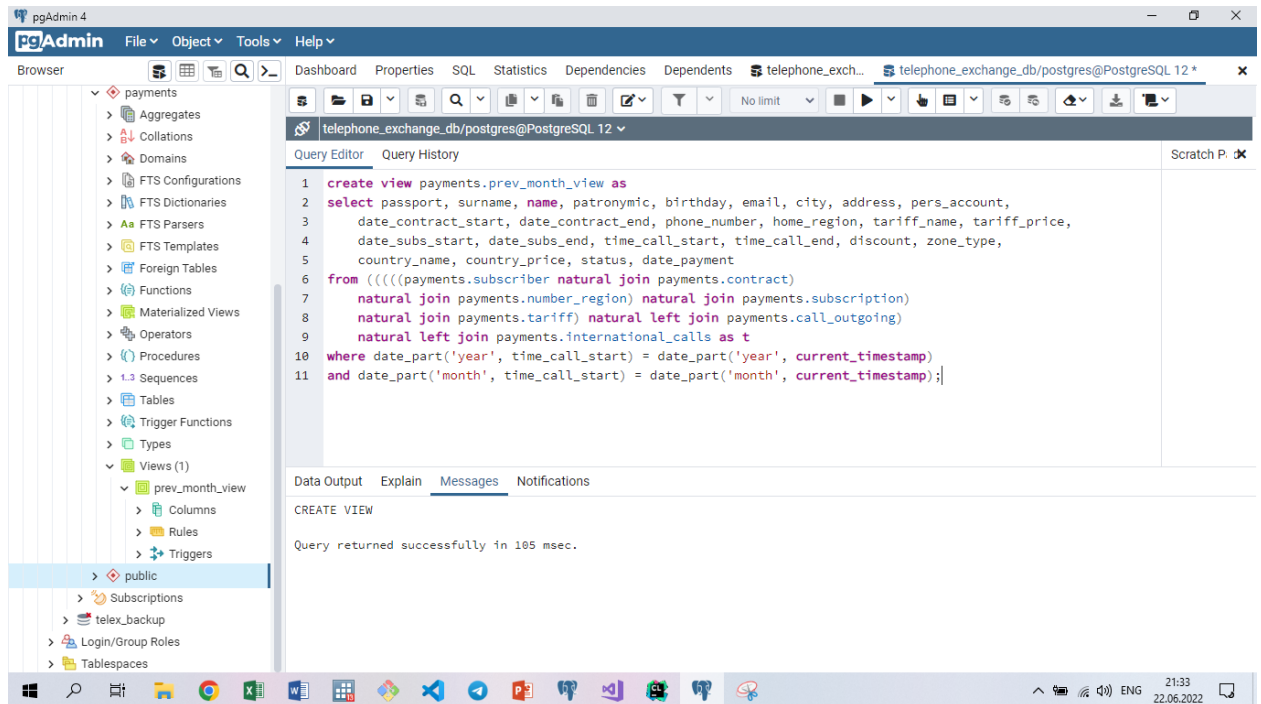


Рисунок 9 – Скриншот выполнения запроса 1

Data Output										
passport	surname	name	patronymic	birthday	email	city	address	pers_account	date_cc	time
character (11)	character varying (100)	character varying (100)	character varying (100)	date	character varying (260)	character varying (200)	character varying (400)	character (14)	timestamp	
1 5116 343435	Аксенов	Дмитрий	Максимович	1993-01-26	dmitriymak@gmail.com	Санкт-Петербург	наб. Обводного канала, 83	5555 5555 5555	2021-01	
2 5116 343435	Аксенов	Дмитрий	Максимович	1993-01-26	dmitriymak@gmail.com	Санкт-Петербург	наб. Обводного канала, 83	5555 5555 5555	2021-01	
3 9000 111112	Самойлов	Нестор	Алексеевич	2001-04-17	nestor01@gmail.com	Наро-Фоминск	пер. Ладыгина, 43	9999 9999 9999	2021-01	
4 1010 674583	Мясникова	Дарья	Олеговна	1989-12-22	mas22dar@gmail.com	Лутовицы	шоссе Космонавтов, 61	1010 1010 1010	2022-01	
5 1111 111111	Иванов	Иван	Иванович	1976-01-01	ivanov@mail.ru	Санкт-Петербург	ул. Большая, 4	1111 1111 1111	2001-01	
6 3668 208482	Мишина	Ксения	Львовна	1988-02-10	ksumi@gmail.com	Ростов	пл. Славы, 02	3333 3333 3333	2020-01	
7 1111 111111	Иванов	Иван	Иванович	1976-01-01	ivanov@mail.ru	Санкт-Петербург	ул. Большая, 4	1111 1111 1111	2001-01	
8 3668 208482	Мишина	Ксения	Львовна	1988-02-10	ksumi@gmail.com	Ростов	пл. Славы, 02	3333 3333 3333	2020-01	

Рисунок 10 – Просмотр содержимого представления 1

2. Найти самую популярную зону звонков за истекший год (рис. 11, 12).

```
create view payments.popular_zone_prev_year_view as
select t.zone_type, t.call_count
from (select zone_type, count(*) as call_count
      from payments.call_outgoing
      where date_part('year', time_call_start) = date_part('year',
current_timestamp)
      group by zone_type) as t
where t.call_count >= all (select count(*)
                           from payments.call_outgoing
                           where date_part('year', time_call_start) =
date_part('year', current_timestamp)
                           group by zone_type);
```

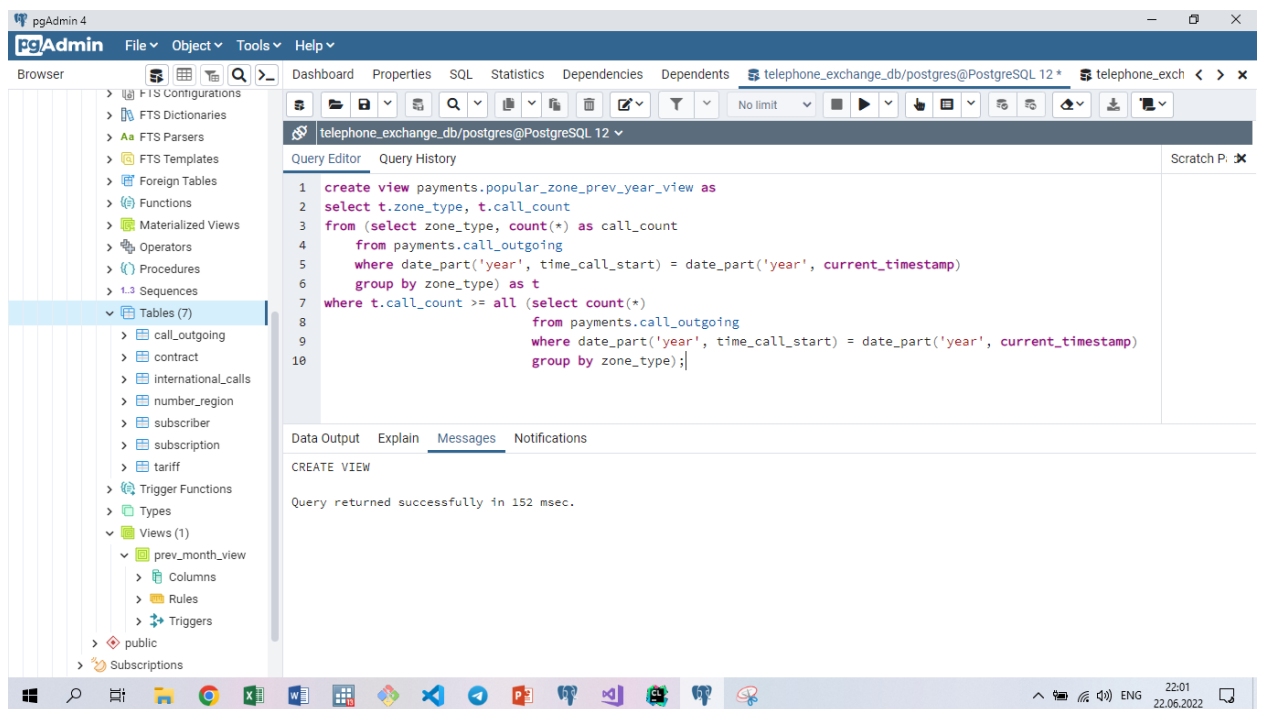


Рисунок 11 – Скриншот выполнения запроса 1

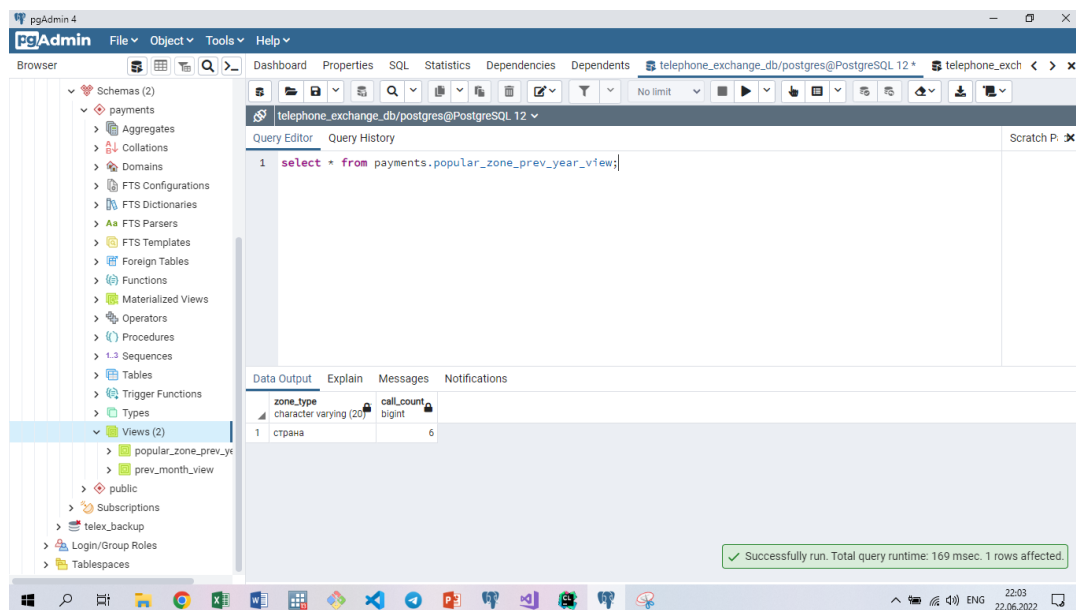


Рисунок 12 – Просмотр содержимого представления 1

Запросы на модификацию данных:

1. Абонентам, которые не совершали звонки за последний год, изменить тариф на стандартный, то есть «Средненький» (рис. 13, 14).
`insert into payments.subscription (tariff_id, phone_number, date_subs_start)`
`select (select tariff_id from payments.tariff where tariff_name = 'Средненький'),`
`phone_number, current_timestamp`
`from payments.subscription`
`where subscription_id not in (select subscription_id from payments.call_outgoing`
`where date_part('year', time_call_start) = date_part('year',`
`current_timestamp));`

subscription_id [PK] integer	tariff_id integer	phone_number character (12)	date_subs_start timestamp without time zone	date_subs_end timestamp without time zone
1	1	+79316538912	2001-03-06 02:00:00	[null]
2	3	+74951473804	2001-01-23 00:00:00	[null]
3	4	+79220711918	2001-01-23 15:15:00	[null]
4	5	+79229001231	2020-08-17 00:00:00	2020-12-28 00:00:00
5	6	+79223214545	2021-01-20 00:00:00	[null]
6	7	+79319319331	2021-01-18 00:00:00	[null]
7	8	+79315680002	2021-03-04 00:00:00	[null]
8	10	+79812342324	2021-03-04 00:00:00	2021-10-01 00:00:00
9	11	+79812342324	2021-10-01 00:00:00	2022-03-04 00:00:00
10	12	+79117775456	2021-06-28 00:00:00	[null]
11	13	+79317776688	2021-06-17 17:02:00	[null]
12	14	+79588652495	2022-02-23 11:15:00	[null]

Рисунок 13 – Скриншот до выполнения запроса 1

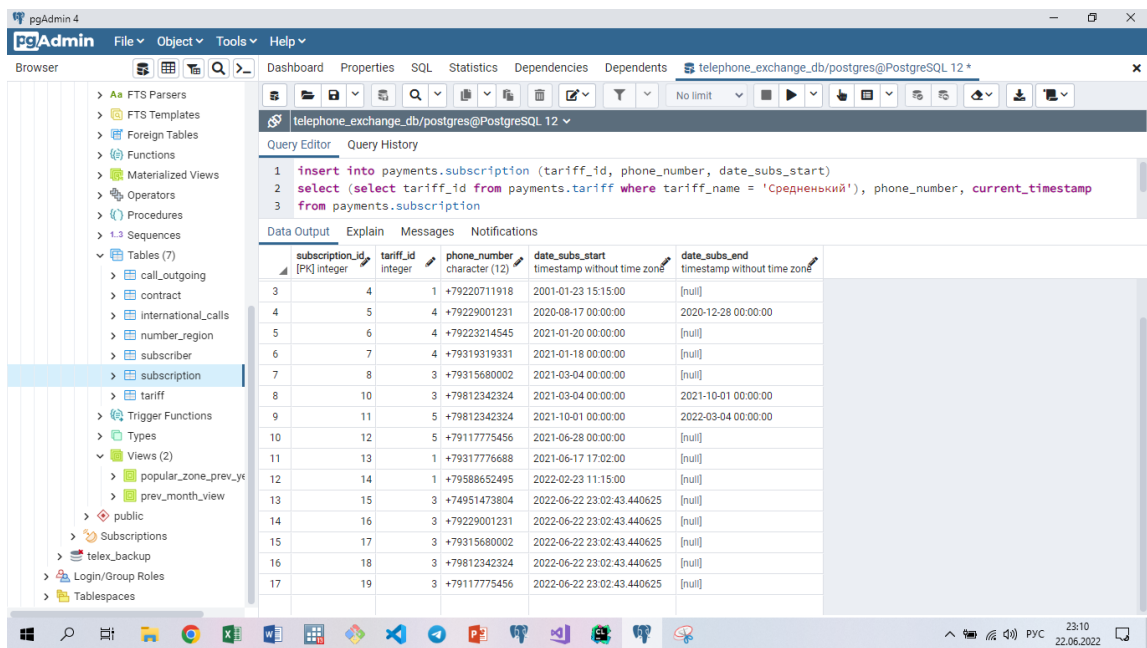


Рисунок 14 - Скриншот после выполнения запроса 1

2. Закрыть долг для абонентов, у которых день рождения в январе (рис. 15, 16).

update payments.call_outgoing

set status = 'оплачено', date_payment = current_timestamp

where call_id in (select call_id from ((payments.call_outgoing natural join

payments.subscription) natural join payments.contract) natural join

payments.subscriber where date_part('month', birthday) = 1 and status =

'ожидание');

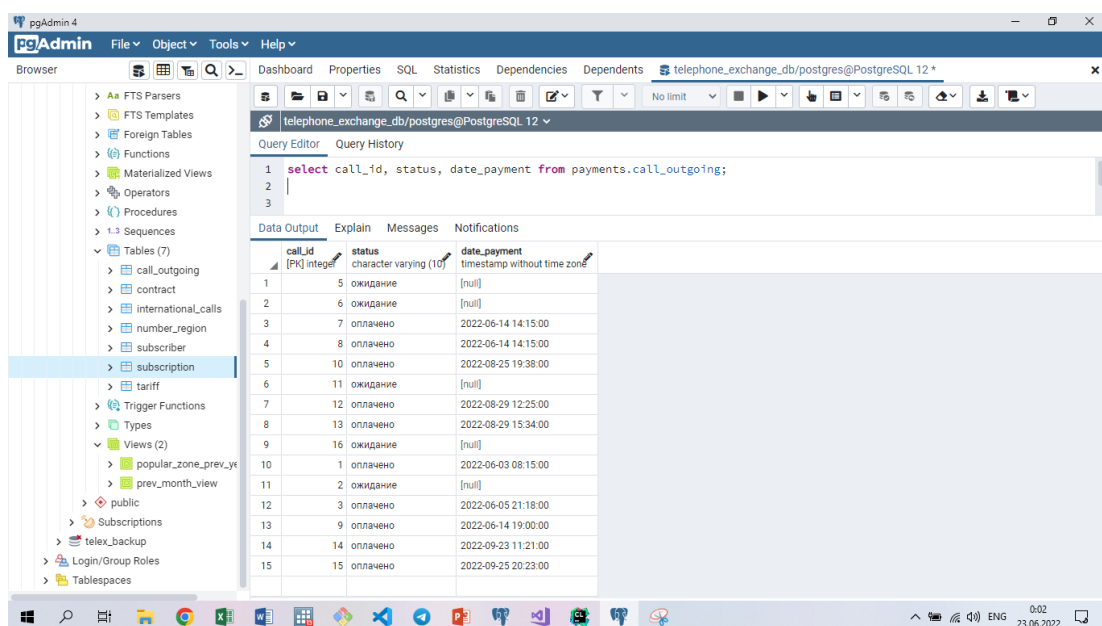


Рисунок 15 – Скриншот до выполнения запроса 2

	call_id [PK] integer	status character varying (10)	date_payment timestamp without time zone
1	7	оплачено	2022-06-14 14:15:00
2	8	оплачено	2022-06-14 14:15:00
3	10	оплачено	2022-08-25 19:38:00
4	11	ожидание	[null]
5	12	оплачено	2022-08-29 12:25:00
6	13	оплачено	2022-08-29 15:34:00
7	1	оплачено	2022-06-03 08:15:00
8	2	ожидание	[null]
9	3	оплачено	2022-06-05 21:18:00
10	9	оплачено	2022-06-14 19:00:00
11	14	оплачено	2022-09-23 11:21:00
12	15	оплачено	2022-09-25 20:23:00
13	16	оплачено	2022-06-23 00:07:45.572892
14	6	оплачено	2022-06-23 00:07:45.572892
15	5	оплачено	2022-06-23 00:07:45.572892

Рисунок 16 – Скриншот после выполнения запроса 2

- Удалить информацию об оплаченных звонках для абонентов, которые расторгли контракт с АТС (рис. 17, 18).

`delete from payments.call_outgoing`

`where call_id in (select call_id from (payments.call_outgoing natural join payments.subscription) natural join payments.contract where date_contract_end is not null and status = 'оплачено');`

telephone_exchange_db/postgres@PostgreSQL 12										
Query Editor Query History										
1 select * from payments.call_outgoing;										
2										
3										
	call_id [PK] integer	subscription_id integer	time_call_start timestamp without time zone	time_call_end timestamp without time zone	zone_type character varying (20)	country_id integer	discount numeric	status character varying (10)	date_paym timestamp	
1	7	13	2022-06-14 14:00:00	2022-06-14 14:10:00	дальнее зарубежье	5	0.0	оплачено	2022-06	
2	8	14	2022-06-14 14:00:00	2022-06-14 14:19:00	город	[null]	0.0	оплачено	2022-06	
3	10	10	2022-08-25 19:32:00	2022-08-25 19:38:00	СНГ	2	0.2	оплачено	2022-08	
4	11	10	2022-08-29 02:00:00	2022-08-29 02:05:00	город	[null]	0.0	ожидание	[null]	
5	12	13	2022-08-29 12:00:00	2022-08-29 12:25:00	дальнее зарубежье	5	0.0	оплачено	2022-08	
6	13	10	2022-08-29 15:17:00	2022-08-29 15:34:00	город	[null]	0.0	оплачено	2022-08	
7	1	1	2022-06-03 08:00:00	2022-06-03 08:10:00	страна	[null]	0.0	оплачено	2022-06	
8	2	4	2022-06-03 08:00:00	2022-06-03 08:30:00	страна	[null]	0.0	ожидание	[null]	
9	3	1	2022-06-05 21:14:00	2022-06-05 21:17:00	страна	[null]	0.2	оплачено	2022-06	
10	9	4	2022-06-14 17:54:00	2022-06-14 18:30:00	страна	[null]	0.0	оплачено	2022-06	
11	14	6	2022-09-23 11:17:00	2022-09-23 11:20:00	страна	[null]	0.0	оплачено	2022-09	
12	15	1	2022-09-25 20:00:00	2022-09-25 20:20:00	страна	[null]	0.2	оплачено	2022-09	
13	5	7	2022-06-13 20:00:00	2022-06-13 21:17:00	СНГ	1	0.2	ожидание	[null]	
14	6	7	2022-06-13 22:00:00	2022-06-13 22:45:00	СНГ	1	0.2	ожидание	[null]	
15	16	7	2022-09-25 20:00:00	2022-09-25 20:20:00	СНГ	1	0.2	ожидание	[null]	

Рисунок 17 – Скриншот до выполнения запроса 3

Data Output		Explain	Messages	Notifications						
	call_id [PK] integer	subscription_id integer	time_call_start timestamp without time zone	time_call_end timestamp without time zone	zone_type character varying (20)	country_id integer	discount numeric	status character varying (10)	date_paym timestamp	
1	7	13	2022-06-14 14:00:00	2022-06-14 14:10:00	дальнее зарубежье	5	0.0	оплачено	2022-06-14	
2	8	14	2022-06-14 14:00:00	2022-06-14 14:19:00	город	[null]	0.0	оплачено	2022-06-14	
3	11	10	2022-08-29 02:00:00	2022-08-29 02:05:00	город	[null]	0.0	ожидание	[null]	
4	12	13	2022-08-29 12:00:00	2022-08-29 12:25:00	дальнее зарубежье	5	0.0	оплачено	2022-08-29	
5	1	1	2022-06-03 08:00:00	2022-06-03 08:10:00	страна	[null]	0.0	оплачено	2022-06-03	
6	2	4	2022-06-03 08:00:00	2022-06-03 08:30:00	страна	[null]	0.0	ожидание	[null]	
7	3	1	2022-06-05 21:14:00	2022-06-05 21:17:00	страна	[null]	0.2	оплачено	2022-06-05	
8	9	4	2022-06-14 17:54:00	2022-06-14 18:30:00	страна	[null]	0.0	оплачено	2022-06-14	
9	14	6	2022-09-23 11:17:00	2022-09-23 11:20:00	страна	[null]	0.0	оплачено	2022-09-23	
10	15	1	2022-09-25 20:00:00	2022-09-25 20:20:00	страна	[null]	0.2	оплачено	2022-09-25	
11	5	7	2022-06-13 20:00:00	2022-06-13 21:17:00	СНГ	1	0.2	ожидание	[null]	
12	6	7	2022-06-13 22:00:00	2022-06-13 22:45:00	СНГ	1	0.2	ожидание	[null]	
13	16	7	2022-09-25 20:00:00	2022-09-25 20:20:00	СНГ	1	0.2	ожидание	[null]	

Рисунок 18 – Скриншот после выполнения запроса 3

Создание индексов:

1. Запрос 4 (рис. 19, 20).

Query Editor

Query History

```

1 explain select distinct passport
2 from (payments.call_outgoing natural join payments.subscription) natural join payments.contract
3 where status = 'ожидание' and date_part('year', time_call_start) = date_part('year', current_timestamp)
4 and date_part('month', time_call_start) = date_part('month', current_timestamp);
5
6

```

Data Output

Explain

Messages

Notifications

QUERY PLAN

text

1 Unique (cost=35.02..35.03 rows=2 width=48)

2 [...] -> Sort (cost=35.02..35.03 rows=2 width=48)

3 [...] Sort Key: contract.passport

4 [...] -> Nested Loop (cost=0.30..35.01 rows=2 width=48)

5 [...] -> Nested Loop (cost=0.15..31.84 rows=1 width=52)

6 [...] -> Seq Scan on call_outgoing (cost=0.00..23.65 rows=1 width=4)

7 [...] Filter: (((status)::text = 'ожидание'::text) AND (date_part('year'::text, time_call_start) = date_part('year'::text, CURRENT_TIMESTAMP))) AND (date_part('month'::text, time_call_start) = date_part('month'::text, CURRENT_TIMESTAMP))

8 [...] -> Index Scan using subscription_pkey on subscription (cost=0.15..8.17 rows=1 width=56)

9 [...] Index Cond: (subscription_id = call_outgoing.subscription_id)

10 [...] -> Index Only Scan using u_pass_tel_date on contract (cost=0.15..3.15 rows=2 width=100)

11 [...] Index Cond: (phone_number = subscription.phone_number)

Рисунок 19 – План запроса 4

Total query runtime: 122 msec.

Создание составного индекса:

create index idx_stts_cstart on payments.call_outgoing(status, time_call_start);

1	<code>create index idx_stts_cstart on payments.call_outgoing(status, time_call_start);</code>
2	
3	<code>explain select distinct passport</code>
4	<code>from (payments.call_outgoing natural join payments.subscription) natural join payments.contract</code>
5	<code>where status = 'ожидание' and date_part('year', time_call_start) = date_part('year', current_timestamp)</code>
6	<code>and date_part('month', time_call_start) = date_part('month', current_timestamp);</code>
7	
8	
Data Output Explain Messages Notifications	
QUERY PLAN	
text	
1	Unique (cost=12.79..12.80 rows=2 width=48)
2	[...] -> Sort (cost=12.79..12.80 rows=2 width=48)
3	[...] Sort Key: contract.passport
4	[...] -> Nested Loop (cost=0.30..12.78 rows=2 width=48)
5	[...] -> Nested Loop (cost=0.15..9.61 rows=1 width=52)
6	[...] -> Seq Scan on call_outgoing (cost=0.00..1.42 rows=1 width=4)
7	[...] Filter: (((status)::text = 'ожидание'::text) AND (date_part('year'::text, time_call_start) = date_part('year'::text, CURRENT_TIMESTAMP))) AND (date_part('month'::text, time_call_start) = date_part('month'::text, CURRENT_TIMESTAMP))
8	[...] -> Index Scan using subscription_pkey on subscription (cost=0.15..8.17 rows=1 width=56)
9	[...] Index Cond: (subscription_id = call_outgoing.subscription_id)
10	[...] -> Index Only Scan using u_pass_tel_date on contract (cost=0.15..3.15 rows=2 width=100)
11	[...] Index Cond: (phone_number = subscription.phone_number)

Рисунок 20 – План запроса 4 после создания индекса

Total query runtime: 96 msec.

Время выполнения запроса после создания индекса уменьшилось.

Удаление индекса:

`drop index payments.idx_stts_cstart;`

2. Запрос 5 (рис. 21, 22).

3	<code>explain select country_name, count(call_id) as calls_num</code>
4	<code>from payments.call_outgoing natural right join payments.international_calls</code>
5	<code>group by country_name</code>
6	<code>having country_name in ('Армения', 'Казахстан', 'Финляндия');</code>
Data Output Explain Messages Notifications	
QUERY PLAN	
text	
1	GroupAggregate (cost=15.21..15.28 rows=4 width=226)
2	[...] Group Key: international_calls.country_name
3	[...] -> Sort (cost=15.21..15.22 rows=4 width=222)
4	[...] Sort Key: international_calls.country_name
5	[...] -> Hash Left Join (cost=1.29..15.17 rows=4 width=222)
6	[...] Hash Cond: (international_calls.country_id = call_outgoing.country_id)
7	[...] -> Seq Scan on international_calls (cost=0.00..13.85 rows=4 width=222)
8	[...] Filter: ((country_name)::text = ANY ('(Армения,Казахстан,Финляндия)::text[]))
9	[...] -> Hash (cost=1.13..1.13 rows=13 width=8)
10	[...] -> Seq Scan on call_outgoing (cost=0.00..1.13 rows=13 width=8)

Рисунок 21 – План запроса 5

Total query runtime: 77 msec.

Создание индекса:

`create index idx_country on payments.international_calls(country_name);`


	QUERY PLAN	
	text	
1	HashAggregate (cost=2.32..2.35 rows=3 width=226)	
2	[...] Group Key: international_calls.country_name	
3	[...] -> Hash Right Join (cost=1.11..2.28 rows=8 width=222)	
4	[...] Hash Cond: (call_outgoing.country_id = international_calls.country_id)	
5	[...] -> Seq Scan on call_outgoing (cost=0.00..1.13 rows=13 width=8)	
6	[...] -> Hash (cost=1.07..1.07 rows=3 width=222)	
7	[...] -> Seq Scan on international_calls (cost=0.00..1.07 rows=3 width=222)	
8	[...] Filter: ((country_name)::text = ANY ('{Армения,Казахстан,Финляндия}':text[]))	

Рисунок 22 – План запроса 5 после создания индекса

Total query runtime: 88 msec.

Время выполнения запроса увеличилось, однако уменьшилось количество пунктов плана запроса.

Удаление индекса:

`drop index payments.idx_country;`

Выводы.

Я овладела практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL и использования подзапросов при модификации данных. Я изучила графическое представление запросов и просмотрела историю запросов. Также я практиковалась в создании индексов и проанализировала их влияние на время выполнения и план запросов.