

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №4 «Запросы на выборку и модификацию данных. Представления. Работа с индексами»

по дисциплине «Проектирование и реализация баз данных»

Автор: Пузенко А.А.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

Цель работы.....	3
Практическое задание.....	3
Выполнение.....	3
Вывод:.....	21

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.

Выполнение

Создать запросы:

1. Сколько раз заправлял автомобиль каждый из клиентов за заданный период.

Query	Query History
1	SELECT c.full_name, COUNT(p.id_purchase) AS purchase_count
2	FROM clients c
3	LEFT JOIN cards ca ON c.id_client = ca.id_client
4	LEFT JOIN purchases p ON ca.id_card = p.id_card AND p.purchase_date BETWEEN '2024-01-01' AND '2024-01-31'
5	GROUP BY c.id_client, c.full_name;

Data Output			Messages	Notifications
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>				
	full_name character varying (100) 🔒	purchase_count bigint 🔒		
1	Dmitry Savin	2		
2	Andrey Bobrov	1		
3	Petr Popov	6		
4	Alexandra Soboleva	9		
5	Elena Morozova	5		
6	Gleb Krylov	4		
7	Maxim Yefimov	1		
8	Ira Zhukova	3		
9	Andrey Puzenko	3		
10	Ivan Testovich	1		
11	Varvara Frolova	1		
12	Sergey Chernov	2		
13	Andrey Puzenko	11		
14	Ivan Ivanov	8		
15	Polina Korneva	2		
16	Kristina Petrova	0		

2. Кто из клиентов не приобретал топливо в январе текущего года?

Query Query History

```

1 SELECT c.full_name
2 FROM clients c
3 LEFT JOIN cards ca ON c.id_client = ca.id_client
4 LEFT JOIN purchases p ON ca.id_card = p.id_card
5     AND EXTRACT(YEAR FROM p.purchase_date) = EXTRACT(YEAR FROM CURRENT_DATE)
6     AND EXTRACT(MONTH FROM p.purchase_date) = 1
7 WHERE p.id_card IS NULL
8 OR p.purchase_date IS NULL;
```

Data Output	Messages	Notificatio
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> </div>		
	client_name character varying (100) 🔒	
1	Kristina Petrova	

3.1. Найти клиента, купившего наибольший объем топлива по всей сети. (в общем)

Query	Query History
1	SELECT c.full_name, SUM (p.amount_fuel) AS total_fuel_amount
2	FROM purchases p
3	JOIN cards ca ON p.id_card = ca.id_card
4	JOIN clients c ON ca.id_client = c.id_client
5	GROUP BY c.id_client
6	HAVING SUM (p.amount_fuel) = (
7	SELECT SUM (amount_fuel)
8	FROM purchases
9	GROUP BY id_card
10	ORDER BY SUM (amount_fuel) DESC
11	LIMIT 1
12);

Data Output	Messages	Notifications
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>		
	full_name character varying (100) 🔒	total_fuel_amount double precision 🔒
1	Ivan Testovich	326
2	Andrey Puzenko	326

3.2. Найти клиента, купившего наибольший объем топлива по всей сети. (единоразово)



```
1 SELECT c.full_name, MAX(p.amount_fuel) AS max_fuel_amount_in_single_purchase
2 FROM purchases p
3 JOIN cards ca ON p.id_card = ca.id_card
4 JOIN clients c ON ca.id_client = c.id_client
5 GROUP BY c.id_client
6 HAVING MAX(p.amount_fuel) = (
7     SELECT MAX(amount_fuel)
8     FROM purchases
9 );
```

Data Output Messages Notifications

	full_name character varying (100)	max_fuel_amount_in_single_purchase double precision
1	Ivan Testovich	326

4. Вывести данные клиента, купившего топлива на наибольшую сумму в заданный день.

Query	Query History
1	SELECT
2	c.full_name ,
3	SUM(p.amount_fuel * fs.price) AS total_spent
4	FROM
5	purchases p
6	JOIN
7	cards ca ON p.id_card = ca.id_card
8	JOIN
9	clients c ON ca.id_client = c.id_client
10	JOIN
11	fuels_sold fs ON p.id_fuel_sold = fs.id_fuel_sold
12	WHERE
13	p.purchase_date = '2024-01-08'
14	GROUP BY
15	c.id_client
16	HAVING
17	SUM(p.amount_fuel * fs.price) = (
18	SELECT MAX(total_spent)
19	FROM (
20	SELECT SUM(p.amount_fuel * fs.price) AS total_spent
21	FROM purchases p
22	JOIN fuels_sold fs ON p.id_fuel_sold = fs.id_fuel_sold
23	WHERE p.purchase_date = '2024-01-08'
24	GROUP BY p.id_card
25) AS max_spent
26);
27	

	full_name character varying (100) 	total_spent double precision 
1	Alexandra Soboleva	2114.20000000000003

5. Какое топливо пользуется наибольшим спросом в прошедшем году на АЗС конкретного поставщика?

```

1  SELECT
2      f.mark,
3      SUM(p.amount_fuel) AS total_sold_amount
4  FROM
5      purchases p
6  JOIN
7      fuels_sold fs ON p.id_fuel_sold = fs.id_fuel_sold
8  JOIN
9      fuels f ON fs.id_fuel = f.id_fuel
10 JOIN
11     gas_stations gs ON fs.id_station = gs.id_station
12 JOIN
13     supplier_firms sf ON gs.id_firm = sf.id_firm
14     AND EXTRACT(YEAR FROM p.purchase_date) = EXTRACT(YEAR FROM CURRENT_DATE) - 1
15 WHERE
16     sf.name = 'ЛУКОЙЛ'
17 GROUP BY
18     f.mark
19
20 HAVING
21     SUM(p.amount_fuel) = (
22         SELECT MAX(total_sold_amount)
23         FROM (
24             SELECT SUM(p.amount_fuel) AS total_sold_amount
25             FROM purchases p
26             JOIN fuels_sold fs ON p.id_fuel_sold = fs.id_fuel_sold
27             JOIN gas_stations gs ON fs.id_station = gs.id_station
28             JOIN supplier_firms sf ON gs.id_firm = sf.id_firm
29             WHERE sf.name = 'ЛУКОЙЛ'
30             AND EXTRACT(YEAR FROM p.purchase_date) = EXTRACT(YEAR FROM CURRENT_DATE) - 1
31             GROUP BY fs.id_fuel
32         ) AS max_sold
33     );

```

Data Output Messages Notifications

	mark							
	character varying (40)							
	total_sold_amount							
	double precision							

6. Сколько топлива каждого вида было продано за прошедший месяц по каждому поставщику на каждой АЗС.

Query

Query History

```

1  SELECT
2      gs.name AS gas_station_name,
3      sf.name AS supplier_name,
4      f.type,
5      SUM(p.amount_fuel) AS total_amount_sold
6  FROM
7      purchases p
8  JOIN
9      fuels_sold fs ON p.id_fuel_sold = fs.id_fuel_sold
10 JOIN
11     gas_stations gs ON fs.id_station = gs.id_station
12 JOIN
13     fuels f ON fs.id_fuel = f.id_fuel
14 JOIN
15     supplier_firms sf ON gs.id_firm = sf.id_firm
16 WHERE
17     EXTRACT(MONTH FROM p.purchase_date) = EXTRACT(MONTH FROM CURRENT_DATE) - 1
18     AND EXTRACT(YEAR FROM p.purchase_date) = EXTRACT(YEAR FROM CURRENT_DATE)
19 GROUP BY
20     gs.name, sf.name, f.type
21 ORDER BY
22     gs.name, sf.name, f.type;

```

Data Output

Messages

Notifications

	gas_station_name character varying (100)	supplier_name character varying (100)	type character varying (40)	total_amount_sold double precision
1	Заправка 1	РОСНЕФТЬ	бензин	62
2	Заправка 1	РОСНЕФТЬ	дизель	96
3	Заправка 2	ЛУКОЙЛ	газ	101
4	Заправка 3	Газпром нефть	бензин	301
5	Заправка 4	РОСНЕФТЬ	бензин	87
6	Заправка 5	ЛУКОЙЛ	газ	211

7. Какая из заправок продала топлива на наибольшую сумму по всем автозаправкам за последний год?

Query
Query History

```

1 WITH total_sales_per_gas_station AS (
2     SELECT
3         fs.id_station,
4         SUM(p.amount_fuel * fs.price) AS total_sales
5     FROM
6         purchases p
7     JOIN
8         fuels_sold fs ON p.id_fuel_sold = fs.id_fuel_sold
9     JOIN
10        fuels f ON fs.id_fuel = f.id_fuel
11    WHERE
12        EXTRACT(YEAR FROM p.purchase_date) = EXTRACT(YEAR FROM CURRENT_DATE) - 1
13    GROUP BY
14        fs.id_station
15 )
16 SELECT
17     gs.name AS gas_station_name,
18     t.total_sales AS total_sales_amount
19 FROM
20     total_sales_per_gas_station t
21 JOIN
22     gas_stations gs ON t.id_station = gs.id_station
23 GROUP BY
24     gs.name, t.total_sales
25 HAVING
26     t.total_sales = (
27         SELECT MAX(total_sales)
28         FROM total_sales_per_gas_station
29     );

```

Data Output
Messages
Notifications

	gas_station_name		total_sales_amount					
	character varying (100)		double precision					

Создать представления:

- 1. Содержащее сведения обо всех АЗС и всех видах топлива, которые они продают;

```
1 CREATE VIEW gas_station_fuels AS
2 SELECT DISTINCT
3     gs.name AS gas_station_name,
4     f.mark,
5     f.type
6 FROM
7     gas_stations gs
8 JOIN
9     fuels_sold fs ON gs.id_station = fs.id_station
10 JOIN
11     fuels f ON fs.id_fuel = f.id_fuel;
12
```

```
1 SELECT * FROM public.gas_station_fuels
```

```
2
```

	gas_station_name character varying (100) 🔒	mark character varying (40) 🔒	type character varying (40) 🔒
1	Заправка 3	Пропат-бутан	газ
2	Заправка 3	АИ-98	бензин
3	Заправка 4	АИ-92	бензин
4	Заправка 1	АИ-92	бензин
5	Заправка 1	АИ-95	бензин
6	Заправка 4	АИ-95	бензин
7	Заправка 2	Пропат-бутан	газ
8	Заправка 1	ДТЗ	дизель
9	Заправка 5	Пропат-бутан	газ
10	Заправка 3	АИ-92	бензин
11	Заправка 1	АИ-98	бензин
12	Заправка 1	ДТЛ	дизель
13	Заправка 2	Метан	газ
14	Заправка 5	Метан	газ
15	Заправка 4	АИ-98	бензин
16	Заправка 2	М-100	мазут
17	Заправка 3	АИ-95	бензин

2. Самая прибыльная АЗС за истекший месяц для каждого производителя.

Query Query History

```

1 CREATE VIEW most_profitable_gas_stations AS
2 SELECT DISTINCT ON (sf.name)
3     sf.name AS supplier_name,
4     gs.name AS gas_station_name,
5     SUM(p.amount_fuel * fs.price) AS total_profit
6 FROM
7     supplier_firms sf
8 JOIN
9     gas_stations gs ON sf.id_firm = gs.id_firm
10 JOIN
11     fuels_sold fs ON gs.id_station = fs.id_station
12 JOIN
13     fuels f ON fs.id_fuel = f.id_fuel
14 JOIN
15     purchases p ON fs.id_fuel_sold = p.id_fuel_sold
16 WHERE
17     DATE_TRUNC('month', p.purchase_date) = DATE_TRUNC('month', CURRENT_DATE) - INTERVAL '1 month'
18 GROUP BY
19     sf.name,
20     gs.name
21 ORDER BY
22     sf.name,
23     SUM(p.amount_fuel * fs.price) DESC;

```

Query Query History

1 SELECT * FROM public.most_profitable_gas_stations

2 |

Data Output Messages Notifications

	supplier_name character varying (100)	gas_station_name character varying (100)	total_profit double precision
1	Газпром нефть	Заправка 3	18284.6
2	ЛУКОЙЛ	Заправка 5	5795.0999999999999
3	РОСНЕФТЬ	Заправка 1	9715.3

Создать запросы на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов:

INSERT

Добавляет сотрудника в таблицу employees на должность Cashier на Заправку 2

Query

Query History

1

```
INSERT INTO employees (full_name, phone, passport_details, id_position, id_station, passport_number)
VALUES ('Ivan Terekhov', '89027350597', 'МВД России Москва',
(SELECT id_position FROM positions WHERE title = 'Cashier'),
(SELECT id_station FROM gas_stations WHERE name = 'Заправка 2'),
'6020001100');
```

Data Output

Messages

Notifications

INSERT 0 1

Query returned successfully in 91 msec.

18	33	Ivan Terekhov	89027350597	МВД России Москва	2	6	6020001100
----	----	---------------	-------------	-------------------	---	---	------------

UPDATE

Увеличивает баланс на карте, которой владеет клиент с именем Andrey Puzenko

Query

Query History

1

```
UPDATE cards
SET balance = balance + 100
WHERE id_client = (
SELECT id_client
FROM clients
WHERE full_name = 'Andrey Puzenko'
);
```

Data Output

Messages

Notifications

UPDATE 1

Query returned successfully in 101 msec.

До

	id_card [PK] bigint	id_client bigint	balance double precision
1	9	4	998

После

	id_card [PK] bigint	id_client bigint	balance double precision
1	9	4	998

DELETE

Удаляет карту клиента, если не было совершено ни одной покупки.

Query
Query History

```

1 DELETE FROM cards
2 WHERE id_card NOT IN (
3     SELECT id_card
4     FROM purchases
5 );

```

Data Output
Messages
Notifications

DELETE 1

Query returned successfully in 56 msec.

До

	id_card [PK] bigint	id_client bigint	balance double precision
1	9	4	1098
2	10	5	1200
3	11	6	10
4	12	7	120.09

После

Query
Query History

1
EXPLAIN ANALYZE SELECT * FROM clients WHERE phone = '8950777777';

Data Output
Messages
Notifications

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	QUERY PLAN	
	text	🔒
1	Seq Scan on clients (cost=0.00..1.19 rows=1 width=44) (actual time=0.018..0.020 rows=1 loops...	
2	Filter: ((phone)::text = '8950777777'::text)	
3	Rows Removed by Filter: 14	
4	Planning Time: 1.896 ms	
5	Execution Time: 0.172 ms	

Graphical
Analysis
Statistics

🔍

🔄

🔍

⬇️

clients

Второй индекс

Query
Query History

1
CREATE INDEX idx_purchases_fuels_sold ON purchases (id_fuel_sold, id_card, purchase_date);

Без индекса

```
1  EXPLAIN ANALYZE
2  SELECT
3      gs.name AS gas_station_name,
4      sf.name AS supplier_name,
5      f.type,
6      SUM(p.amount_fuel) AS total_amount_sold
7  FROM
8      purchases p
9  JOIN
10     fuels_sold fs ON p.id_fuel_sold = fs.id_fuel_sold
11  JOIN
12     fuels f ON fs.id_fuel = f.id_fuel
13  JOIN
14     gas_stations gs ON fs.id_station = gs.id_station
15  JOIN
16     supplier_firms sf ON gs.id_firm = sf.id_firm
17  WHERE
18     p.purchase_date BETWEEN '2024-02-01' AND '2024-02-29'
19  GROUP BY
20     gs.name, sf.name, f.type
21  ORDER BY
22     gs.name, sf.name, f.type;
```

1	GroupAggregate (cost=19.36..20.17 rows=36 width=542) (actual time=4.115..4.128 rows=6 loops=1)
2	Group Key: gs.name, sf.name, f.type
3	-> Sort (cost=19.36..19.45 rows=36 width=542) (actual time=3.919..3.922 rows=36 loops=1)
4	Sort Key: gs.name, sf.name, f.type
5	Sort Method: quicksort Memory: 27kB
6	-> Hash Join (cost=4.91..18.43 rows=36 width=542) (actual time=2.074..2.125 rows=36 loops=1)
7	Hash Cond: (gs.id_firm = sf.id_firm)
8	-> Hash Join (cost=3.84..17.19 rows=36 width=332) (actual time=1.145..1.190 rows=36 loops=1)
9	Hash Cond: (fs.id_station = gs.id_station)
10	-> Nested Loop (cost=2.73..15.91 rows=36 width=114) (actual time=0.895..0.932 rows=36 loops=1)
11	-> Hash Join (cost=2.58..5.09 rows=36 width=24) (actual time=0.705..0.722 rows=36 loops=1)
12	Hash Cond: (p.id_fuel_sold = fs.id_fuel_sold)
13	-> Seq Scan on purchases p (cost=0.00..2.41 rows=36 width=16) (actual time=0.591..0.600 rows=36 loops=1)
14	Filter: ((purchase_date >= '2024-02-01'::date) AND (purchase_date <= '2024-02-29'::date))
15	Rows Removed by Filter: 59
16	-> Hash (cost=1.70..1.70 rows=70 width=24) (actual time=0.054..0.054 rows=70 loops=1)
17	Buckets: 1024 Batches: 1 Memory Usage: 12kB
18	-> Seq Scan on fuels_sold fs (cost=0.00..1.70 rows=70 width=24) (actual time=0.027..0.035 rows=70 loops=1)
19	-> Memoize (cost=0.15..0.86 rows=1 width=106) (actual time=0.006..0.006 rows=1 loops=36)
20	Cache Key: fs.id_fuel
21	Cache Mode: logical
22	Hits: 29 Misses: 7 Evictions: 0 Overflows: 0 Memory Usage: 1kB
20	Cache Key: fs.id_fuel
21	Cache Mode: logical
22	Hits: 29 Misses: 7 Evictions: 0 Overflows: 0 Memory Usage: 1kB
23	-> Index Scan using "Fuels_pkey" on fuels f (cost=0.14..0.85 rows=1 width=106) (actual time=0.026..0.026 rows=1 loop...)
24	Index Cond: (id_fuel = fs.id_fuel)
25	-> Hash (cost=1.05..1.05 rows=5 width=234) (actual time=0.158..0.158 rows=5 loops=1)
26	Buckets: 1024 Batches: 1 Memory Usage: 9kB
27	-> Seq Scan on gas_stations gs (cost=0.00..1.05 rows=5 width=234) (actual time=0.148..0.150 rows=5 loops=1)
28	-> Hash (cost=1.03..1.03 rows=3 width=226) (actual time=0.898..0.898 rows=3 loops=1)
29	Buckets: 1024 Batches: 1 Memory Usage: 9kB
30	-> Seq Scan on supplier_firms sf (cost=0.00..1.03 rows=3 width=226) (actual time=0.635..0.641 rows=3 loops=1)
31	Planning Time: 7.787 ms
32	Execution Time: 6.125 ms

С индексом

```
1 EXPLAIN ANALYZE
2 SELECT
3     gs.name AS gas_station_name,
4     sf.name AS supplier_name,
5     f.type,
6     SUM(p.amount_fuel) AS total_amount_sold
7 FROM
8     purchases p
9 JOIN
10    fuels_sold fs ON p.id_fuel_sold = fs.id_fuel_sold
11 JOIN
12    fuels f ON fs.id_fuel = f.id_fuel
13 JOIN
14    gas_stations gs ON fs.id_station = gs.id_station
15 JOIN
16    supplier_firms sf ON gs.id_firm = sf.id_firm
17 WHERE
18     p.purchase_date BETWEEN '2024-02-01' AND '2024-02-29'
19 GROUP BY
20     gs.name, sf.name, f.type
21 ORDER BY
22     gs.name, sf.name, f.type;
```

1	GroupAggregate (cost=19.45..20.28 rows=37 width=542) (actual time=0.349..0.364 rows=6 loops=1)
2	Group Key: gs.name, sf.name, f.type
3	-> Sort (cost=19.45..19.54 rows=37 width=542) (actual time=0.338..0.341 rows=36 loops=1)
4	Sort Key: gs.name, sf.name, f.type
5	Sort Method: quicksort Memory: 27kB
6	-> Hash Join (cost=4.91..18.48 rows=37 width=542) (actual time=0.212..0.260 rows=36 loops=1)
7	Hash Cond: (gs.id_firm = sf.id_firm)
8	-> Hash Join (cost=3.84..17.24 rows=37 width=332) (actual time=0.134..0.176 rows=36 loops=1)
9	Hash Cond: (fs.id_station = gs.id_station)
10	-> Nested Loop (cost=2.73..15.95 rows=37 width=114) (actual time=0.096..0.131 rows=36 loops=1)
11	-> Hash Join (cost=2.58..5.11 rows=37 width=24) (actual time=0.054..0.069 rows=36 loops=1)
12	Hash Cond: (p.id_fuel_sold = fs.id_fuel_sold)
13	-> Seq Scan on purchases p (cost=0.00..2.42 rows=37 width=16) (actual time=0.011..0.020 rows=36 loops=1)
14	Filter: ((purchase_date >= '2024-02-01'::date) AND (purchase_date <= '2024-02-29'::date))
15	Rows Removed by Filter: 59
16	-> Hash (cost=1.70..1.70 rows=70 width=24) (actual time=0.029..0.029 rows=70 loops=1)
17	Buckets: 1024 Batches: 1 Memory Usage: 12kB
18	-> Seq Scan on fuels_sold fs (cost=0.00..1.70 rows=70 width=24) (actual time=0.011..0.017 rows=70 loops=1)
19	-> Memoize (cost=0.15..0.86 rows=1 width=106) (actual time=0.001..0.001 rows=1 loops=36)
20	Cache Key: fs.id_fuel
21	Cache Mode: logical
22	Hits: 29 Misses: 7 Evictions: 0 Overflows: 0 Memory Usage: 1kB
23	-> Index Scan using "Fuels_pkey" on fuels f (cost=0.14..0.85 rows=1 width=106) (actual time=0.006..0.006 rows=1 loop...)
24	Index Cond: (id_fuel = fs.id_fuel)
25	-> Hash (cost=1.05..1.05 rows=5 width=234) (actual time=0.029..0.030 rows=5 loops=1)
26	Buckets: 1024 Batches: 1 Memory Usage: 9kB
27	-> Seq Scan on gas_stations gs (cost=0.00..1.05 rows=5 width=234) (actual time=0.024..0.025 rows=5 loops=1)
28	-> Hash (cost=1.03..1.03 rows=3 width=226) (actual time=0.069..0.069 rows=3 loops=1)
29	Buckets: 1024 Batches: 1 Memory Usage: 9kB
30	-> Seq Scan on supplier_firms sf (cost=0.00..1.03 rows=3 width=226) (actual time=0.060..0.062 rows=3 loops=1)
31	Planning Time: 10.932 ms
32	Execution Time: 0.472 ms

Вывод:

В данной лабораторной работе я освоил запросы на выборку данных в базе данных PostgreSQL. А также были созданы представления, запросы на удаление, обновление и добавление данных. Были созданы простые и составные индексы и сравнено время выполнения запросов с индексами и без.