

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

Отчет

по лабораторной работе «ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ, ПРЕДСТАВЛЕНИЯ И
ИНДЕКСЫ В PostgreSQL»

по дисциплине «**Базы данных**»

Автор: Казарян Т.Г.

Факультет: ИКТ

Группа: K3241

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL 1X, pgAdmin 4.

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.

Ход работы:

1. Выполнение запросов согласно индивидуальному заданию, часть 2.

- Вывести данные официанта, принявшего заказы на максимальную сумму за истекший месяц.

The screenshot shows the pgAdmin 4 interface. The 'Query' tab is active, displaying a SQL query that selects waiter information based on the highest order sum for the current month. The 'Data Output' tab shows the results of the query in a table format.

```
1 SELECT e .*
2 FROM restauran.employee e
3 JOIN restauran.orders o
4   ON e. id_employee = o.id_waiter
5 WHERE EXTRACT (MONTH
6 FROM o.order_date) = EXTRACT (MONTH FROM CURRENT_DATE)
7 GROUP BY e.id_employee
8 HAVING SUM(o.cost) =
9   (SELECT MAX(order_sum)
10  FROM
11   (SELECT SUM(cost) AS order_sum
12    FROM restauran.orders
13    WHERE EXTRACT (MONTH
14 FROM order_date) = EXTRACT (MONTH
15 FROM CURRENT_DATE)
16    GROUP BY id_waiter ) AS x );
```

	id_employee [PK] integer	passport_data character varying (100)	fio character varying (50)	id_post integer
1	3	3804 638490	Петров Максим Максимович	3

- Рассчитать премию каждого официанта за последние 10 дней (5% от стоимости каждого заказа).

Query Query History

```

1 SELECT e.id_employee,
2       e.fio,
3       SUM(o.cost * 0.05) AS Премия
4 FROM restaurant.employee e
5 JOIN restaurant.orders o
6   ON e.id_employee = o.id_waiter
7 WHERE o.order_date >= CURRENT_DATE - INTERVAL '10 days'
8 GROUP BY e.id_employee, e.fio;

```

Data Output Messages Notifications

	id_employee [PK] integer	fio character varying (50)	Премия numeric
1	3	Петров Максим Максимович	63.15
2	1	Иванов Иван Иванович	25.3475

- Подсчитать, сколько ингредиентов содержит каждое блюдо.

Query Query History

```

1 SELECT d.name,
2       COUNT(*) AS sum_of_ing
3 FROM restaurant.dish AS d
4 JOIN restaurant.dish_composition c
5   ON d.id_dish = c.id_dish
6 GROUP BY d.name
7 ORDER BY sum_of_ing DESC;

```

Data Output Messages Notifications

	name character varying (30)	sum_of_ing bigint
1	Мясо по-французски	2
2	Цезарь	1

- Вывести название блюда, содержащее максимальное число ингредиентов.

Query	Query History
1	SELECT di.name
2	FROM restaurant.dish AS di
3	JOIN restaurant.dish_composition AS d
4	ON di.id_dish = d.id_dish
5	GROUP BY di.id_dish
6	HAVING COUNT(*) =
7	(SELECT MAX(count)
8	FROM
9	(SELECT COUNT(*) AS count
10	FROM restaurant.dish_composition
11	GROUP BY id_dish) AS x);

Data Output	Messages	Notifications
<div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>📦</div> <div>⬇️</div> <div>📈</div> </div>		
	name	
	character varying (30)	🔒
1	Мясо по-французски	

- Какой повар может приготовить максимальное число видов блюд?

Query	Query History
1	SELECT e.fio
2	FROM restaurant.employee AS e
3	JOIN restaurant.skill AS s
4	ON e.id_employee = s.id_employee
5	GROUP BY e.id_employee
6	HAVING COUNT(*) =
7	(SELECT MAX(count)
8	FROM
9	(SELECT COUNT(*) AS count
10	FROM restaurant.skill
11	GROUP BY id_employee) AS x);

Data Output	Messages	Notifications
<div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>📦</div> <div>⬇️</div> <div>📈</div> </div>		
	fio	
	character varying (50)	🔒
1	Лысенко Виктория Юрьевна	

- Сколько закреплено столов за каждым из официантов?

Query	Query History
1	SELECT e.id_employee, e.fio,
2	COUNT (DISTINCT o.id_table) AS Количество_столов
3	FROM restaurant.employee e
4	JOIN restaurant.orders o
5	ON e.id_employee = o.id_waiter
6	GROUP BY e.id_employee, e.fio;

Data Output	Messages	Notifications
<div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>📦</div> <div>⬇️</div> <div>📈</div> </div>		
	id_employee	fio
	[PK] integer	character varying (50)
		Количество_столов
		bigint
1	1	Иванов Иван Иванович
2	3	Петров Максим Максимович

- Какой из ингредиентов используется в максимальном количестве блюд?

```

1 SELECT i.name
2 FROM restaurant.ingredient AS i
3 JOIN restaurant.dish_composition AS d
4   ON i.id_ingredient = d.id_ingredient
5 GROUP BY i.id_ingredient
6 HAVING COUNT(*) =
7   (SELECT MAX(count)
8    FROM
9      (SELECT COUNT(*) AS count
10       FROM restaurant.dish_composition
11        GROUP BY id_ingredient) AS x);

```

Data Output	Messages	Notifications
<div> <div>name</div> <div>character varying (40)</div> </div>		
1	Говядина	

2. Выполнение запросов на создание представлений согласно индивидуальному заданию, часть 3.

- для расчета стоимости ингредиентов для заданного блюда;

```

SELECT d.name AS dish_name,

       sum(co.ingredient_volume * pu.ingredient_price) AS cost_of_ingredients

FROM restaurant.dish d

JOIN restaurant.dish_composition co ON d.id_dish = co.id_dish

JOIN restaurant.purchase pu ON co.id_ingredient = pu.id_ingredient

GROUP BY d.name;

```

Data Output	Messages	Notifications
CREATE VIEW		
Query returned successfully in 222 msec.		

```

Query  Query History
1 SELECT cost_of_ingredients
2 FROM restaurant.ingredient_dish_cost
3 WHERE dish_name = 'Мясо по-французски';

```

Data Output	Messages	Notifications
<div> <div>cost_of_ingredients</div> <div>bigint</div> </div>		
1	100	

1 **SELECT** * **from** restaurant.ingredient_dish_cost

Data OutputMessagesNotifications

☰+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	dish_name character varying (30) 🔒	cost_of_ingredients bigint 🔒
1	Мясо по-французски	100
2	Цезарь	80

- для всех поваров количество приготовленных блюд по каждому блюду за определенную дату.

```
CREATE OR REPLACE VIEW chef_dish_counts AS
SELECT
```

```
    e.id_employee AS chef_id,
    d.name AS dish_name,
    o.order_date::date AS preparation_date,
    COUNT(*) AS dish_count
```

```
FROM
```

```
    restaurant.employee e
  JOIN restaurant.skill s ON e.id_employee = s.id_employee
  JOIN restaurant.dish d ON s.id_dish = d.id_dish
  JOIN restaurant.order_composition oc ON d.id_dish = oc.id_dish
  JOIN restaurant.orders o ON oc.id_order = o.id_order
```

```
GROUP BY
```

```
    e.id_employee, d.name, o.order_date::date;
```

	chef_id integer 🔒	dish_name character varying (30) 🔒	preparation_date date 🔒	dish_count bigint 🔒
1	2	Торт "Наполеон"	2023-12-15	1
2	2	Мясо по-французски	2023-12-15	1
3	4	Мясо по-французски	2023-12-15	1

3. Выполнение запросов на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.

INSERT

```
INSERT into restaurant.skill (id_employee, id_dish) values (  
(SELECT id_employee  
FROM restaurant.employee  
WHERE fio = 'Меньшин Александр Юрьевич'), 2);
```

До:

	id_employee integer	id_dish integer	skill_id [PK] integer
1	2	2	1
2	2	1	2
3	2	3	3
4	4	1	4

После:

	id_employee integer	id_dish integer	skill_id [PK] integer
1	2	2	1
2	2	1	2
3	2	3	3
4	4	1	4
5	4	2	5

UPDATE

```
UPDATE restaurant.ingredient SET quantity_in_stock = quantity_in_stock - 1  
WHERE id_ingredient IN (SELECT id_ingredient  
FROM restaurant.dish_composition  
WHERE ingredient_volume > 1);
```

До:

	id_ingredient [PK] integer	name character varying (40)	product_type character varying (20)	required_stock double precision	quantity_in_stock double precision	caloric_content integer
1	1	Лук	Овощи	5	3.5	40
2	2	Говядина	Мясо	8	7	187
3	3	Яйцо куриное	Яйца	10	9	143

После:

	id_ingredient [PK] integer	name character varying (40)	product_type character varying (20)	required_stock double precision	quantity_in_stock double precision	caloric_content integer
1	1	Лук	Овощи	5	3.5	40
2	2	Говядина	Мясо	8	7	187
3	3	Яйцо куриное	Яйца	10	8	143

DELETE

```
DELETE FROM restauran.employee
WHERE id_employee NOT IN
(SELECT id_employee
FROM restauran.skill)
AND id_post = 2;
```

До:

	id_employee [PK] integer	passport_data character varying (100)	fio character varying (50)	id_post integer
1	1	4018 998445	Иванов Иван Иванович	3
2	2	5334 584036	Лысенко Виктория Юрьевна	1
3	3	3804 638490	Петров Максим Максимович	3
4	4	4567 377988	Меньшин Александр Юрьевич	1
5	5	6546 646416	Козловский Марк Григорьевич	2
6	6	4882 457235	Кулаков Никита Сергеевич	2
7	7	1623 698877	Исаев Матвей Семёнович	2

После:

	id_employee [PK] integer	passport_data character varying (100)	fio character varying (50)	id_post integer
1	1	4018 998445	Иванов Иван Иванович	3
2	2	5334 584036	Лысенко Виктория Юрьевна	1
3	3	3804 638490	Петров Максим Максимович	3
4	4	4567 377988	Меньшин Александр Юрьевич	1

Query

Query History

1

explain analyze select name from restaurant.ingredient;

Data Output

Messages

Notifications

+

▼

▼

QUERY PLAN

text

1

Seq Scan on ingredient (cost=0.00..1.03 rows=3 width=98) (actual time=0.013..0.014 rows=3 loops...

2

Planning Time: 0.076 ms

3

Execution Time: 0.034 ms

2 Пример:

Без индекса:

Query

Query History

1

```
EXPLAIN ANALYZE
SELECT fio
FROM restaurant.employee
WHERE id_employee = ANY (
  SELECT id_employee
  FROM restaurant.skill
  WHERE id_dish = 1
);
```

9

10

Data Output

Messages

Notifications

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

QUERY PLAN

text

🔒

1

Hash Semi Join (cost=35.62..48.39 rows=10 width=118) (actual time=0.058..0.059 rows=2 loops=1)

2

Hash Cond: (employee.id_employee = skill.id_employee)

3

-> Seq Scan on employee (cost=0.00..12.10 rows=210 width=122) (actual time=0.018..0.018 rows=4...

4

-> Hash (cost=35.50..35.50 rows=10 width=4) (actual time=0.017..0.017 rows=2 loops=1)

5

Buckets: 1024 Batches: 1 Memory Usage: 9kB

6

-> Seq Scan on skill (cost=0.00..35.50 rows=10 width=4) (actual time=0.008..0.009 rows=2 loops...

7

Filter: (id_dish = 1)

8

Rows Removed by Filter: 3

9

Planning Time: 0.183 ms

10

Execution Time: 0.091 ms

Создание индекса:

```
1 create index FIOIndex on restaurant.employee(fio);
```

С индексом:

Query

Query History

1

```
explain analyze select fio from restaurant.employee;
```

Data Output

Messages

Notifications

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

QUERY PLAN

text

🔒

1

Seq Scan on employee (cost=0.00..1.04 rows=4 width=118) (actual time=0.011..0.012 rows=4 loops...

2

Planning Time: 0.312 ms




3

Execution Time: 0.023 ms

3 Пример:

Без индекса:

Query	Query History
1	EXPLAIN ANALYZE
2	SELECT *
3	FROM restaurant.orders
4	WHERE order_date BETWEEN '2023-12-01' AND '2023-12-31'
5	AND id_table = 1;

Data Output	Messages	Notifications
       		
QUERY PLAN		
text		
1	Seq Scan on orders (cost=0.00..26.10 rows=1 width=60) (actual time=0.220..0.222 rows=1 loops=1)	
2	Filter: ((order_date >= '2023-12-01 00:00:00':timestamp without time zone) AND (order_date <= '2023-12-31 00:00:00':timestamp without time zone) AND (id_table ...	
3	Rows Removed by Filter: 1	
4	Planning Time: 13.336 ms	
5	Execution Time: 0.676 ms	

Создание индекса:

1	CREATE INDEX idx_orders_order_date_id_table ON restaurant.orders (order_date, id_table);
---	--

Data Output	Messages	Notifications
CREATE INDEX		
Query returned successfully in 68 msec.		

С индексом:

1	EXPLAIN ANALYZE
2	SELECT *
3	FROM restaurant.orders
4	WHERE order_date BETWEEN '2023-12-01' AND '2023-12-31'
5	AND id_table = 1;
6	

Data Output	Messages	Notifications
       		
QUERY PLAN		
text		
1	Seq Scan on orders (cost=0.00..1.03 rows=1 width=60) (actual time=0.013..0.014 rows=1 loops=1)	
2	Filter: ((order_date >= '2023-12-01 00:00:00':timestamp without time zone) AND (order_date <= '2023-12-31 00:00:00':timestamp without time zone) AND (id_table ...	
3	Rows Removed by Filter: 1	
4	Planning Time: 0.416 ms	
5	Execution Time: 0.029 ms	

После создания индексов запросы выполняются быстрее.

- Индексы помогают ускорить выполнение запросов, особенно тех, которые включают условия фильтрации или сортировки по индексированным столбцам. Однако, создание индексов также может повлечь проблемы при вставке, обновлении или удалении данных, так как индексы должны быть поддержаны и обновлены.

Вывод

В ходе лабораторной работы я научился создавать представления и запросы на выборку данных к базе данных PostgreSQL, используя подзапросы при модификации данных и индексов. По моему мнению, эти знания полезны и часто применяются на практике.