

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №4 «Создание таблиц базы данных PostgreSQL. Заполнение
таблиц рабочими данными»
«Анализ данных. Построение инфологической модели данных»

по дисциплине **«Проектирование и реализация баз данных»**

Автор: Дерещук Татьяна Евгеньевна

Факультет: ИКТ

Группа: К3140

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

Цель работы	3
Практическое задание	3
Выполнение	3
Вывод	12

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.
(добавила данные)

The screenshot shows a PostgreSQL query editor with a query window and a results window. The query window contains the following SQL code:

```
1 SELECT c.customer_id,
2      cu.full_name,
3      COUNT(s.date_sold) AS total
4 FROM schema_1.card c
5      JOIN schema_1.customer cu ON c.customer_id = cu.customer_id
6      JOIN schema_1.service_1 s ON c.card_number = s.card_number
7 WHERE s.date_sold BETWEEN '2023-01-01' AND '2023-01-10'
8 GROUP BY c.customer_id, cu.full_name;
```

The results window shows the output of the query:

	customer_id bigint	full_name character varying (100)	total bigint
1	1	Иван Иванов	1
2	4	Наталья Смирнова	1
3	13	Анна Семенчук	1

- (2) Какое топливо пользуется наибольшим спросом в прошедшем году на АЗС конкретного поставщика?

SELECT

schema_1.fuel.brand,

```

    schema_1.fuel.type,
    SUM(schema_1.service_1.amount_sold_fuel) AS total_amount
FROM
    schema_1.service_1
JOIN
    schema_1.sold_fuel ON schema_1.service_1.sold_fuel_code = schema_1.sold_fuel.sold_fuel_code
JOIN
    schema_1.fuel ON schema_1.sold_fuel.fuel_code = schema_1.fuel.fuel_code
JOIN
    schema_1.gas_station ON schema_1.sold_fuel.gas_station_code =
schema_1.gas_station.gas_station_code
JOIN
    schema_1.fuel_supplier_company ON schema_1.gas_station.fuel_type =
schema_1.fuel_supplier_company.fuel_type
WHERE
    schema_1.fuel_supplier_company.date_at_the_end >= CURRENT_DATE - INTERVAL '1 year'
GROUP BY
    schema_1.fuel.brand, schema_1.fuel.type
ORDER BY
    total_amount DESC

```

	brand character varying (20) 🔒	type character varying (20) 🔒	total_amount numeric 🔒
1	Пропан	Газ	13

- (3) Кто из клиентов не приобрел топливо в июле текущего года?

```

SELECT c.full_name, s.date_sold
FROM schema_1.customer c
LEFT JOIN schema_1.card cr ON c.customer_id = cr.customer_id
LEFT JOIN schema_1.service_1 s ON cr.card_number = s.card_number
WHERE s.date_sold IS NULL OR (EXTRACT(MONTH FROM s.date_sold) != 7 OR EXTRACT(YEAR FROM
s.date_sold) != EXTRACT(YEAR FROM CURRENT_DATE))

```

	full_name character varying (100)	date_sold date
1	Алексей Сидоров	2023-06-29
2	Наталья Смирнова	2023-05-18
3	Елена Морозова	2023-04-26
4	Анна Иванова	2023-06-24

- (4) Найти клиента, купившего наибольший объем топлива по всей сети. (id)

Query
Query History

```

1 SELECT c.customer_id,
2       s.card_number,
3       SUM(s.amount_sold_fuel) AS total_fuel_amount
4 FROM schema_1.customer c
5 JOIN schema_1.card cd ON c.customer_id = cd.customer_id
6 JOIN schema_1.service_1 s ON cd.card_number = s.card_number
7 GROUP BY c.customer_id, s.card_number
8 HAVING SUM(s.amount_sold_fuel) = (SELECT MAX(total_amount)
9                                FROM (SELECT SUM(amount_sold_fuel) AS total_amount
11                                   FROM schema_1.service_1
12                                   GROUP BY card_number) AS max_amount)
12 ORDER BY s.card_number

```

Data Output
Messages
Notifications

	customer_id bigint	card_number bigint	total_fuel_amount numeric
1	9	2819282632040862	5

- (5) Вывести данные клиента, купившего топлива на наибольшую сумму в заданный день.

Query

Query History

1

2

3

4

5

6

7

8

9

10

11

12

13

```
SELECT c.customer_id,
      s.date_sold,
      SUM(s.amount_sold_fuel) AS total_fuel_amount
FROM schema_1.customer c
JOIN schema_1.card cd ON c.customer_id = cd.customer_id
JOIN schema_1.service_1 s ON cd.card_number = s.card_number
WHERE s.date_sold = '2023-07-21'
GROUP BY c.customer_id, s.date_sold
HAVING SUM(s.amount_sold_fuel) = (SELECT MAX(total_amount)
                                  FROM (SELECT SUM(amount_sold_fuel) AS total_amount
                                        FROM schema_1.service_1
                                        WHERE date_sold = '2023-07-21'
                                        GROUP BY card_number) AS max_amount)
```

Data Output

Messages

Notifications

+

📄

▼

📋

▼

🗑️

🔄

⬇️

📈

	customer_id	date_sold	total_fuel_amount
	bigint	date	numeric
1	9	2023-07-21	5

- (6) Какая из заправок продала топлива на наибольшую сумму по всем автозаправкам за последний год?

Query

Query History

```
1 SELECT gas_station_code, total_sales
2 FROM (
3     SELECT gs.gas_station_code, SUM(sf.price) AS total_sales
4     FROM schema_1.gas_station gs
5     INNER JOIN schema_1.sold_fuel sf ON gs.gas_station_code = sf.gas_station_code
6     WHERE sf.price_start_date >= CURRENT_DATE - INTERVAL '1 year'
7     GROUP BY gs.gas_station_code
8 ) AS sales
9 WHERE total_sales = (
10     SELECT MAX(total_sales)
11     FROM (
12         SELECT SUM(sf.price) AS total_sales
13         FROM schema_1.gas_station gs
14         INNER JOIN schema_1.sold_fuel sf ON gs.gas_station_code = sf.gas_station_code
15         WHERE sf.price_start_date >= CURRENT_DATE - INTERVAL '1 year'
16         GROUP BY gs.gas_station_code
17     ) AS sales
18 );
```

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑️

🔄

⬇️

📈

	gas_station_code [PK] bigint	total_sales numeric
1	9	58

- (7) Сколько топлива каждого вида было продано за прошедший месяц по каждому поставщику на каждой АЗС.

Query

Query History

```

1 SELECT
2     c.fuel_type,
3     gs.gas_station_code,
4     MAX(sf.price) AS max_price
5 FROM
6     schema_1.gas_station gs
7 JOIN
8     schema_1.fuel_supplier_company c ON gs.fuel_type = c.fuel_type
9 JOIN
10    schema_1.sold_fuel sf ON gs.gas_station_code = sf.gas_station_code
11 WHERE
12     sf.price_end_date >= CURRENT_DATE - INTERVAL '1 month'
13 GROUP BY
14     c.fuel_type,
15     gs.gas_station_code
16 ORDER BY
17     c.fuel_type,
18     gs.gas_station_code;

```

Data Output

Messages

Notifications

	fuel_type bigint	gas_station_code bigint	max_price bigint
1	1	1	50
2	1	4	52
3	1	7	53
4	1	10	54
5	2	2	45
6	2	5	47
7	2	8	48
8	3	3	55
9	3	6	57
10	3	9	58

Создать представление:

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

CREATE OR REPLACE VIEW schema_1.gas_station_fu

CREATE VIEW

```

SELECT
    gs.gas_station_code,
    gs.company_phone,
    f.fuel_code,
    f.brand AS fuel_brand,
    f.type AS fuel_type
FROM
    schema_1.gas_station gs
JOIN schema_1.fuel_supplier_company fsc ON gs.
JOIN schema_1.fuel f ON fsc.fuel_type = f.fuel

```

Query returned successfully in 80 msec.

```
1 SELECT * FROM schema_1.gas_station_fuel_info;
```

Data Output

Messages

Notifications

	gas_station_code bigint	company_phone bigint	fuel_code bigint	fuel_brand character varying (20)	fuel_type character
1	1	9876543210	1	Бензин	АИ-95
2	2	1234567890	2	Дизель	ДТ-Л
3	3	5678901234	3	Газ	Пропан
4	4	8901234567	1	Бензин	АИ-95
5	5	9012345678	2	Дизель	ДТ-Л
6	6	3456789012	3	Газ	Пропан
7	7	9999999999	1	Бензин	АИ-95
8	8	8888888888	2	Дизель	ДТ-Л
9	9	7777777777	3	Газ	Пропан
10	10	6666666666	1	Бензин	АИ-95

- Самая прибыльная АЗС за истекший месяц для каждого производителя.

```

1 CREATE OR REPLACE VIEW schema_1.most_profitable_gas_stations AS
2 SELECT DISTINCT ON (f.brand, EXTRACT(MONTH FROM sf.price_end_date))
3 f.brand AS manufacturer,
4 gs.gas_station_code AS gas_station,
5 sf.price * s.amount_sold_fuel AS profit
6 FROM schema_1.fuel_supplier_company fsc
7 JOIN schema_1.gas_station gs ON fsc.fuel_type = gs.fuel_type
8 JOIN schema_1.sold_fuel sf ON gs.gas_station_code = sf.gas_station_code
9 JOIN schema_1.service_1 s ON sf.sold_fuel_code = s.sold_fuel_code
10 JOIN schema_1.fuel f ON fsc.fuel_type = f.fuel_code
11 WHERE EXTRACT(MONTH FROM sf.price_end_date) = EXTRACT(MONTH FROM CURRENT_DATE - INTERVAL '1 month')
12 ORDER BY f.brand, EXTRACT(MONTH FROM sf.price_end_date), profit DESC;

```


Составь 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.

- INSERT с использованием подзапроса: Добавляет нового клиента в таблицу schema_1.customer, при этом используя подзапросы для вычисления customer_id и phone.

Query	Query History
1	INSERT INTO schema_1.customer (full_name, customer_address, customer_id, phone)
2	VALUES ('Иван Иванов', 'ул. Ленина, д. 10',
3	(SELECT MAX(customer_id)+1 FROM schema_1.customer),
4	(SELECT MAX(phone)+1 FROM schema_1.customer));

Data Output	Messages	Notifications
INSERT 0 1		
Query returned successfully in 92 msec.		

- Этот запрос обновляет баланс карты клиента с именем "John Smith", увеличивая его на 1000.

Query	Query History
1	UPDATE schema_1.card
2	SET account_balance = account_balance + 1000
3	WHERE customer_id = (SELECT customer_id FROM schema_1.customer WHERE full_name = 'John Smith');

Data Output	Messages	Notifications
UPDATE 0		
Query returned successfully in 122 msec.		

- Этот запрос выбирает даты продажи топлива, где количество проданного топлива меньше 2.

Query	Query History
1	DELETE FROM schema_1.service_1
2	WHERE date_sold IN (SELECT date_sold FROM schema_1.service_1 WHERE amount_sold_fuel < 2);

- Вывести всех клиентов с их адресами и номерами телефонов (без индексов)

Query Query History

```
1 SELECT c.full_name, c.customer_address, c.phone
2 FROM schema_1.customer c;
```

Data Output Messages Notifications

Successfully run. Total query runtime: 112 msec.
10 rows affected.

- Вывести всех клиентов с их адресами и номерами телефонов (с индексами)

Query Query History

```
1 SELECT c.full_name, c.customer_address, c.phone
2 FROM schema_1.customer c;
```

Data Output Messages Notifications

Successfully run. Total query runtime: 89 msec.
10 rows affected.


Простой: CREATE INDEX idx_customer_address ON schema_1.customer(customer_address);

Составной: CREATE INDEX idx_customer_address_phone ON
schema_1.customer(customer_address, phone);

Data Output Messages Explain × Notifications

Graphical Analysis Statistics

🔍 🔄 🔍 ⬇️



customer

- Получить информацию о всех проданных типах топлива, их брендах и количестве проданного топлива (без индексов)

Query Query History

```

1 SELECT f.brand, f.type, sf.price, sf.price_start_date, sf.price_end_date, s1.amount_sold_fuel
2 FROM schema_1.sold_fuel sf
3 JOIN schema_1.fuel f ON sf.fuel_code = f.fuel_code
4 JOIN schema_1.service_1 s1 ON sf.sold_fuel_code = s1.sold_fuel_code;

```

Data Output Messages Notifications

Successfully run. Total query runtime: 110 msec.
10 rows affected.

- Получить информацию о всех проданных типах топлива, их брендах и количестве проданного топлива (с индексами)

Query Query History

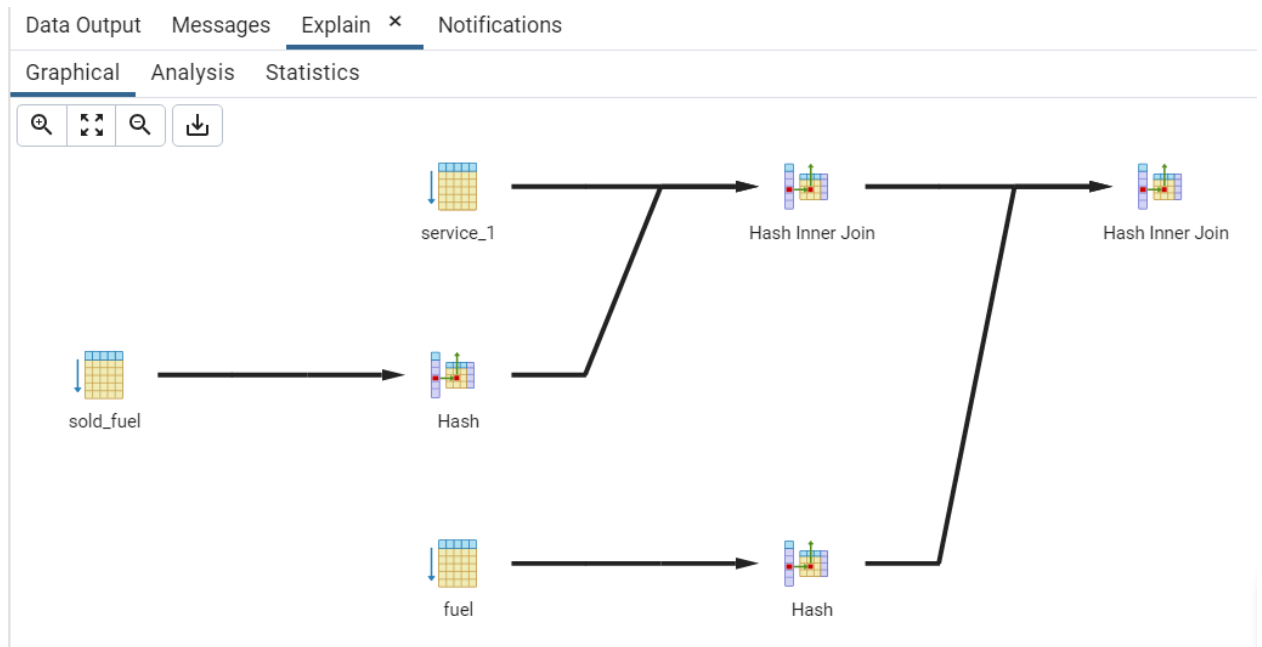
```

1 SELECT f.brand, f.type, s1.amount_sold_fuel
2 FROM schema_1.sold_fuel sf
3 JOIN schema_1.fuel f ON sf.fuel_code = f.fuel_code
4 JOIN schema_1.service_1 s1 ON sf.sold_fuel_code = s1.sold_fuel_code;

```

Data Output Messages Notifications

Successfully run. Total query runtime: 74 msec.
10 rows affected.



Индексы:

Простой: CREATE INDEX idx_fuel_code ON schema_1.sold_fuel (fuel_code);

Составной: CREATE INDEX idx_brand_type ON schema_1.fuel (brand, type);

Вывод

В данной лабораторной работе мы овладели практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов. Были созданы запросы и представления на выборку данных к базе данных PostgreSQL согласно индивидуальным заданиям с использованием подзапросов. Было изучено графическое представление запросов и истории запросов, создана простые и составные индексы для двух произвольных запросов и сравнено время выполнения запросов без индексов и с индексами, используя команду EXPLAIN.