

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИТМО»**

**Факультет инфокоммуникационных технологий**

**Дисциплина:**

**«Проектирование и реализация баз данных»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2  
«ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ,  
ПРЕДСТАВЛЕНИЯ И ИНДЕКСЫ В POSTGRESQL»**

**Выполнил:**

студент группы К32392

Тишалович Леонид

Михайлович

---

(подпись)

**Проверил(а):**

Говорова Марина Михайловна

---

(отметка о выполнении)

---

(подпись)

Санкт-Петербург  
2023 г.

**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 посмотреть 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.

**Индивидуальное практическое задание:**

База данных “Оптовая база”

Запросы на выборку:

1. Вывести список поставщиков, которые поставляют все товары.
2. Определить поставщика, который поставляет каждый из товаров по самой низкой цене.
3. Вывести названия товаров, цены на которые у всех поставщиков одинаковы.
4. Чему равен общий суточный доход оптового склада за прошедший день?
5. Вычислить общую стоимость каждого вида товара, находящегося на базе.
6. В какой день было вывезено минимальное количество товара?
7. Сколько различных видов товара имеется на базе?

Представления:

1. количество заказов фирм-покупателей за прошедший год;
2. доход базы за конкретный период.

## Выполнение

### Запросы на выборку:

1. Вывести список поставщиков, которые поставляют все товары.

```
SELECT p.name_of_company
FROM provider p
JOIN shipment s ON p.provider_id = s.provider_id
JOIN shipment_content sc ON s.shipment_id = sc.shipment_id
JOIN product pr ON sc.product_id = pr.product_id
GROUP BY p.provider_id, p.name_of_company
HAVING COUNT(DISTINCT pr.product_id) = (SELECT COUNT(DISTINCT
pr1.product_id) FROM product pr1);
```

Рис. 1 - SELECT №1


	name_of_company character varying (70) 
1	Company A
2	Company B
3	Company C

Рис. 2 - Результат SELECT №1

2. Определить поставщика, который поставляет каждый из товаров по самой низкой цене.

```
SELECT
    p.name_of_company,
    pr.product_name,
    sc.price AS lowest_price
FROM
    provider p
JOIN shipment s ON p.provider_id = s.provider_id
JOIN shipment_content sc ON s.shipment_id = sc.shipment_id
JOIN product pr ON sc.product_id = pr.product_id
WHERE
    NOT EXISTS (
        SELECT 1
        FROM
            shipment_content sc2
            JOIN shipment s2 ON sc2.shipment_id = s2.shipment_id
        WHERE
            s2.provider_id = p.provider_id
            AND sc2.product_id = sc.product_id
            AND sc2.price < sc.price
    )
GROUP BY
    p.name_of_company, pr.product_name, sc.price;
```

Рис. 3 - SELECT №2

	<b>name_of_company</b> character varying (70) 🔒	<b>product_name</b> character varying (70) 🔒	<b>lowest_price</b> integer 🔒
1	Company A	Chair	100
2	Company A	Laptop	500
3	Company A	Milk	2
4	Company B	Chair	100
5	Company B	Laptop	480
6	Company B	Milk	2
7	Company C	Chair	100
8	Company C	Laptop	450
9	Company C	Milk	2

Рис. 4 - Результат SELECT №2

3. Вывести названия товаров, цены на которые у всех поставщиков одинаковы.

```
SELECT pr.product_name
FROM product pr
WHERE NOT EXISTS (
    SELECT sc1.price
    FROM shipment_content sc1
    JOIN shipment sh1 ON sc1.shipment_id = sh1.shipment_id
    WHERE sc1.product_id = pr.product_id AND EXISTS (
        SELECT sc2.price
        FROM shipment_content sc2
        JOIN shipment sh2 ON sc2.shipment_id = sh2.shipment_id
        WHERE sc2.product_id = pr.product_id AND sc1.price != sc2.price
    )
);
```

Рис. 5 - SELECT №3

	<b>product_name</b> character varying (70) 🔒	<b>price</b> integer 🔒
1	Chair	100
2	Milk	2

Рис. 6 - Результат SELECT №3

4. Чему равен общий суточный доход оптового склада за прошедший день?

```
SELECT
    SUM(pc.amount_of_goods * sc.price) AS daily_income
FROM
    purchase_content pc
    JOIN purchase_invoice pi ON pc.purchase_id = pi.purchase_id
    JOIN purchase p ON pc.purchase_id = p.purchase_id
    JOIN shipment_content sc ON pc.product_id = sc.product_id
WHERE
    pi.invoice_date = '2023-05-04'
    AND p.status = 'completed';
```

Рис. 7 - SELECT №4

5. Вычислить общую стоимость каждого вида товара, находящегося на базе.

```
SELECT
    p.product_name,
    SUM(sc.remains * sc.price) AS total_cost
FROM
    product p
    JOIN shipment_content sc ON p.product_id = sc.product_id
GROUP BY
    p.product_name;
```

Рис. 8 - SELECT №5



	product_name character varying (70) 	total_cost bigint 
1	Chair	6000
2	Laptop	14300
3	Milk	300

Рис. 9 - Результат SELECT №5

6. В какой день было вывезено минимальное количество товара?

```
SELECT
    sc.expiration_date,
    SUM(sc.amount_of_goods) AS total_goods
FROM
    shipment_content sc
    JOIN shipment s ON sc.shipment_id = s.shipment_id
```

```

WHERE
    s.invoice_fulfillment_date = (
        SELECT MIN(invoice_fulfillment_date)
        FROM shipment
    )
GROUP BY
    sc.expiration_date
ORDER BY
    total_goods ASC
LIMIT 1;

```

Рис. 10 - SELECT №6

	expiration_date date	total_goods bigint
1	2023-06-30	10

Рис. 11 - Результат SELECT №6

7. Сколько различных видов товара имеется на базе?

```

SELECT COUNT(DISTINCT pr.product_id) as total_types
FROM product pr;

```

Рис. 12 - SELECT №7

	num_products bigint
1	3

Рис. 13 - Результат SELECT №7

### Представления:

1. Количество заказов фирм-покупателей за прошедший год;

```

CREATE VIEW num_orders_last_year AS
SELECT
    c.organization_name,
    COUNT(*) AS num_orders
FROM
    purchase p
    JOIN client c ON p.client_id = c.client_id
WHERE
    p.invoice_fulfillment_date >= DATE_TRUNC('year', CURRENT_DATE -
INTERVAL '1 year')

```

```

AND p.invoice_fulfillment_date < DATE_TRUNC('year', CURRENT_DATE)
GROUP BY
c.organization_name;

```

Рис. 14 - VIEW №1

	organization_name character varying (70) 🔒	num_orders bigint 🔒
1	Org A	1
2	Org B	1

Рис. 15 - Пример VIEW №1

2. Доход базы за конкретный период.

```

CREATE VIEW base_income AS
SELECT
    SUM(sc.amount_of_goods * sc.price) AS income
FROM
    shipment_content sc
    JOIN shipment s ON sc.shipment_id = s.shipment_id
WHERE
    s.invoice_fulfillment_date >= '2023-01-01'
    AND s.invoice_fulfillment_date <= '2023-04-30';

```

Рис. 16 - VIEW №2

	income bigint 🔒
1	27375

Рис. 17 - Пример VIEW №2

## INSERT:

Этот запрос добавит нового поставщика с именем 'Company D', если такой поставщик еще не существует в таблице. **NOT EXISTS** здесь используется для предотвращения дублирования данных.

```

INSERT INTO provider (name_of_company, full_name, address)
SELECT 'Company D', 'David Brown', '1010 D Street'
WHERE NOT EXISTS (
    SELECT 1 FROM provider
    WHERE name_of_company = 'Company D' AND full_name = 'David Brown'
    AND address = '1010 D Street'
);

```

Рис. 16 - INSERT

	provider_id [PK] bigint	name_of_company character varying (70)	full_name character varying (70)	address character varying (70)
1	1	Company A	John Smith	123 A Street
2	2	Company B	Jane Doe	456 B Street
3	3	Company C	Alex Johnson	789 C Street

Рис. 17 - до INSERT

	provider_id [PK] bigint	name_of_company character varying (70)	full_name character varying (70)	address character varying (70)
1	1	Company A	John Smith	123 A Street
2	2	Company B	Jane Doe	456 B Street
3	3	Company C	Alex Johnson	789 C Street
4	4	Company D	David Brown	1010 D Street

Рис. 18 - после INSERT

**UPDATE:**

Этот запрос обновит адрес клиента 'Client A' с 'Organization A' на '111 A Avenue'. Подзапрос используется для определения **client\_id** по имени и названию организации.

```
UPDATE client
SET address = '111 A Avenue'
WHERE client_id = (
    SELECT client_id FROM client
    WHERE full_name = 'Client A' AND organization_name = 'Organization
A'
);
```

Рис. 19 - UPDATE

	client_id [PK] bigint	full_name character varying (70)	organization_name character varying (70)	address character varying (70)
1	2	Client B	Organization B	456 B Avenue
2	3	Client C	Organization C	789 C Avenue
3	1	Client A	Organization A	111 A Avenue

Рис. 20 - до UPDATE

	client_id [PK] bigint	full_name character varying (70)	organization_name character varying (70)	address character varying (70)
1	2	Client B	Organization B	456 B Avenue
2	3	Client C	Organization C	789 C Avenue
3	1	Client A	Organization A	111 A Avenue



Рис. 21 - после UPDATE

### DELETE:

Этот запрос удалит продукт 'Apple' типа 'Fruit' из таблицы **product**. Подзапрос используется для определения **product\_id** по имени и типу продукта.

```
DELETE FROM product
WHERE product_id = (
    SELECT product_id FROM product
    WHERE product_name = 'Apple' AND product_type = 'Fruit'
);
```

Рис. 22 - DELETE

	product_id [PK] bigint	product_type character varying (70)	product_name character varying (70)	unit_of_measurement_id bigint
1	1	Fruit	Apple	1
2	2	Fruit	Orange	1
3	3	Drink	Milk	2
4	4	Furniture	Table	3

Рис. 23 - до DELETE

	product_id [PK] bigint	product_type character varying (70)	product_name character varying (70)	unit_of_measurement_id bigint
1	2	Fruit	Orange	1
2	3	Drink	Milk	2
3	4	Furniture	Table	3

Рис. 24 - после DELETE

### Простой индекс:

```
SELECT p.name_of_company
FROM provider p
JOIN shipment s ON p.provider_id = s.provider_id
JOIN shipment_content sc ON s.shipment_id = sc.shipment_id
JOIN product pr ON sc.product_id = pr.product_id
GROUP BY p.provider_id, p.name_of_company
HAVING COUNT(DISTINCT pr.product_id) = (SELECT COUNT(DISTINCT
pr1.product_id) FROM product pr1);
CREATE INDEX idx_provider_name_of_company
ON provider (name_of_company);
```

Рис. 25 - простой INDEX

	QUERY PLAN	
	text	🔒
1	GroupAggregate (cost=38.40..38.43 rows=1 width=166)	
2	Group Key: p.provider_id	
3	Filter: (count(DISTINCT pr.product_id) = \$0)	
4	InitPlan 1 (returns \$0)	
5	-> Aggregate (cost=12.75..12.76 rows=1 width=8)	
6	-> Seq Scan on product pr1 (cost=0.00..12.20 rows=220 width=8)	
7	-> Sort (cost=25.64..25.64 rows=2 width=174)	
8	Sort Key: p.provider_id	
9	-> Nested Loop (cost=2.42..25.63 rows=2 width=174)	
10	-> Hash Join (cost=2.28..17.27 rows=2 width=174)	
11	Hash Cond: (s.shipment_id = sc.shipment_id)	
12	-> Hash Join (cost=1.07..15.77 rows=74 width=174)	
13	Hash Cond: (s.provider_id = p.provider_id)	
14	-> Seq Scan on shipment s (cost=0.00..13.70 rows=370 width=16)	
15	-> Hash (cost=1.06..1.06 rows=1 width=166)	
16	-> Seq Scan on provider p (cost=0.00..1.06 rows=1 width=166)	
17	Filter: ((name_of_company)::text = 'Some Provider Name'::text)	
18	-> Hash (cost=1.09..1.09 rows=9 width=16)	
19	-> Seq Scan on shipment_content sc (cost=0.00..1.09 rows=9 width=16)	
20	-> Index Only Scan using product_pkey on product pr (cost=0.14..4.16 rows=1 width=...	
21	Index Cond: (product_id = sc.product_id)	

Рис. 26 - EXPLAIN для простого INDEX

Проведя серию замеров из 20 измерений, получилось сократить среднее время выполнения с 87мс до 77мс.

### Составной индекс:

```
EXPLAIN SELECT
  p.name_of_company,
  pr.product_name,
  MIN(sc.price) AS lowest_price
FROM
  provider p
  JOIN shipment s ON p.provider_id = s.provider_id
  JOIN shipment_content sc ON s.shipment_id = sc.shipment_id
  JOIN product pr ON sc.product_id = pr.product_id
WHERE
  sc.price = (
    SELECT
      MIN(sc2.price)
    FROM
      shipment_content sc2
      JOIN shipment s2 ON sc2.shipment_id = s2.shipment_id
    WHERE
      s2.provider_id = p.provider_id
      AND sc2.product_id = sc.product_id
  )
GROUP BY
  p.name_of_company, pr.product_name;
CREATE INDEX idx_shipment_content_shipment_id_product_id
ON shipment_content (shipment_id, product_id);
```

Рис. 27 - составной INDEX

Проведя серию замеров из 30 измерений, получилось сократить среднее время выполнения с 954мс до 872мс.

	QUERY PLAN text
1	GroupAggregate (cost=21.72..21.75 rows=1 width=320)
2	Group Key: p.name_of_company, pr.product_name
3	-> Sort (cost=21.72..21.73 rows=1 width=320)
4	Sort Key: p.name_of_company, pr.product_name
5	-> Nested Loop (cost=2.46..21.71 rows=1 width=320)
6	-> Hash Join (cost=2.32..17.52 rows=1 width=170)
7	Hash Cond: (s.provider_id = p.provider_id)
8	Join Filter: ((SubPlan 1) = sc.price)
9	-> Hash Join (cost=1.20..16.38 rows=9 width=20)
10	Hash Cond: (s.shipment_id = sc.shipment_id)
11	-> Seq Scan on shipment s (cost=0.00..13.70 rows=370 width=16)
12	-> Hash (cost=1.09..1.09 rows=9 width=20)
13	-> Seq Scan on shipment_content sc (cost=0.00..1.09 rows=9 width=20)
14	-> Hash (cost=1.05..1.05 rows=5 width=166)
15	-> Seq Scan on provider p (cost=0.00..1.05 rows=5 width=166)
16	SubPlan 1
17	-> Aggregate (cost=14.63..14.64 rows=1 width=4)
18	-> Nested Loop (cost=0.15..14.63 rows=1 width=4)
19	-> Seq Scan on shipment_content sc2 (cost=0.00..1.11 rows=1 width=12)
20	Filter: (product_id = sc.product_id)
21	-> Index Scan using shipment_pkey on shipment s2 (cost=0.15..8.17 rows=1 width=...
22	Index Cond: (shipment_id = sc2.shipment_id)
23	Filter: (provider_id = p.provider_id)
24	-> Index Scan using product_pkey on product pr (cost=0.14..4.16 rows=1 width=166)
25	Index Cond: (product_id = sc.product_id)

Рис. 28 - EXPLAIN для составного INDEX

### Выводы:

В процессе выполнения лабораторной работы получилось ознакомиться с составлением INSERT, UPDATE, DELETE запросов. Также удалось ознакомиться с графическим представлением запросов. Были составлены простые и составные индексы, что позволило наглядно увидеть уменьшение кол-ва этапов при выполнении запроса, а также уменьшение времени выполнения.