

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

Факультет инфокоммуникационных технологий

Отчет

по лабораторной работе № 2

**«ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ,
ПРЕДСТАВЛЕНИЯ И ИНДЕКСЫ В POSTGRESQL»**

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

Выполнил:
студент 2 курса ИКТ
группы К32422
Демидов Максим Евгеньевич

Преподаватель:
Говорова Марина Михайловна

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, 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.

Вариант 9. БД «Оптовая база»

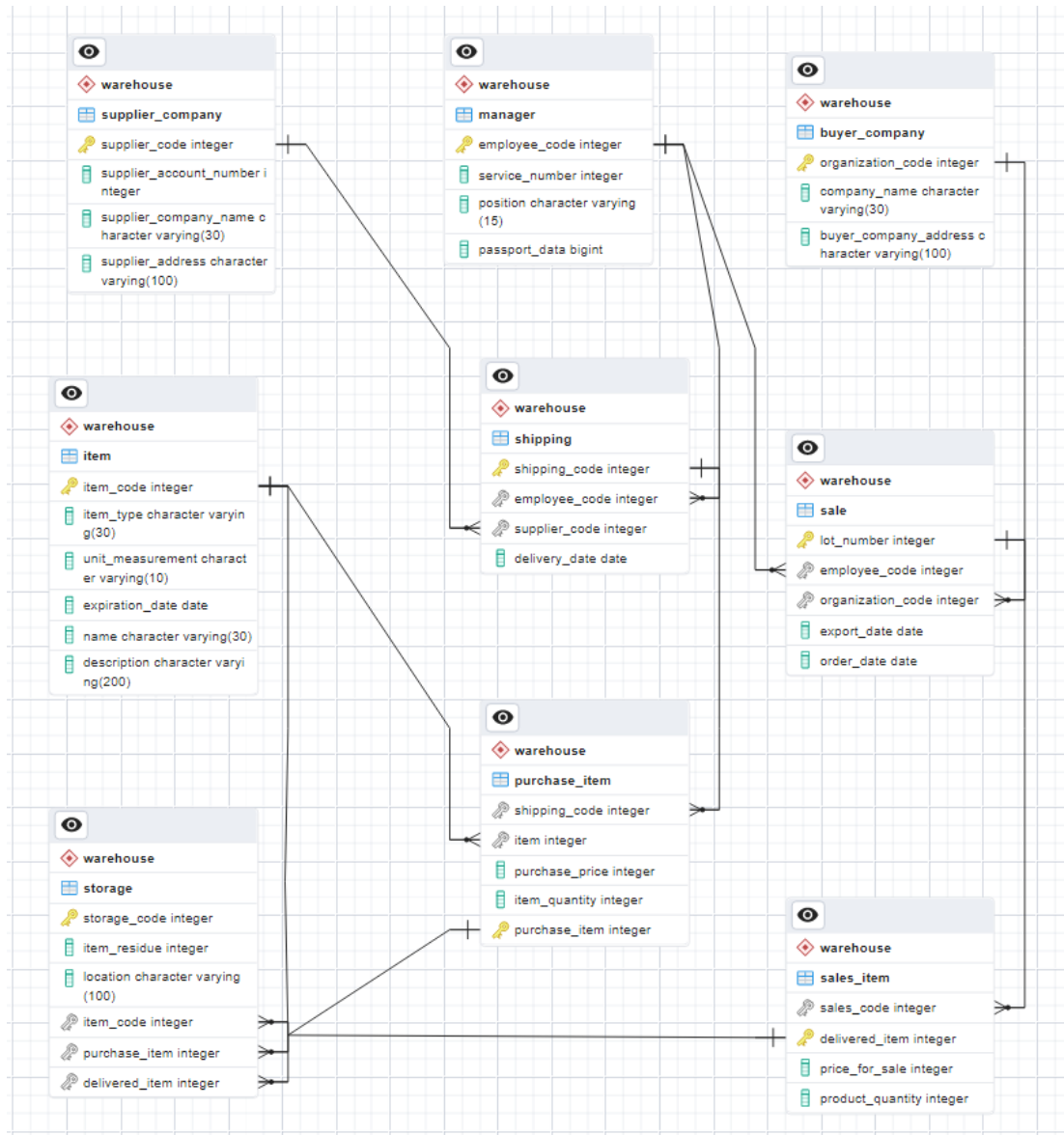
Описание предметной области: Оптовая база закупает товары у компаний-поставщиков и поставляет их компаниям – покупателям. Доход оптовой базы составляет не менее 5% от стоимости товара, проданного компании – покупателю. Один и тот же товар может доставляться несколькими поставщиками, и один и тот же поставщик может поставлять несколько видов товаров. Цены поставки товара у разных поставщиков могут отличаться. Поставки и заказы обслуживают менеджеры по работе с клиентами (по поставкам и продажам).

Задание 2. Создать запросы:

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

Задание 3. Создать представления:

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



Ход работы:

1. Создание запросов

- Вывести список поставщиков, которые поставляют все товары

```
SELECT supplier_code
```

```
FROM warehouse.shipping
```

```
JOIN warehouse.purchase_item ON shipping.shipping_code =  
purchase_item.shipping_code
```

```
GROUP BY shipping.supplier_code
```

```
HAVING COUNT(DISTINCT purchase_item.item) = (SELECT  
COUNT(*) FROM warehouse.purchase_item)
```

В этом запросе мы объединяем таблицы shipping и purchase_item по внешнему ключу shipping_code. Затем мы группируем результаты по supplier_code и используем оператор COUNT(DISTINCT pi.item) для подсчета количества уникальных товаров, которые поставяет каждый поставщик.

Далее, с помощью оператора HAVING, мы выбираем только те записи, у которых количество уникальных товаров, поставляемых поставщиком, равно общему количеству товаров в таблице purchase_item.

Таким образом, данный запрос вернет список всех поставщиков, которые поставяют все товары из таблицы purchase_item.

- Определить поставщика, который поставяет каждый из товаров по самой низкой цене.

```
SELECT
```

```
s.supplier_code,
```

```
pi.item
```

```
FROM
```

```
warehouse.shipping s
```

```
JOIN warehouse.purchase_item pi ON s.shipping_code =  
pi.shipping_code
```

```
JOIN (
```

```
SELECT
```

```
item,
```

```
MIN(purchase_price) AS min_price
```

```
FROM
```

```
warehouse.purchase_item
```

GROUP BY

item

) min_prices ON pi.item = min_prices.item AND pi.purchase_price = min_prices.min_price

GROUP BY

s.supplier_code,

pi.item

HAVING

COUNT(*) = (SELECT COUNT(DISTINCT item) FROM warehouse.purchase_item)

Этот запрос объединяет таблицы shipping и purchase_item по внешнему ключу shipping_code, затем присоединяет таблицу, содержащую минимальные цены для каждого товара, используя подзапрос. Затем запрос группирует результаты по поставщику и товару, исключая результаты, в которых поставщик не поставляет каждый товар по минимальной цене. Это достигается с помощью фильтрации через оператор HAVING, который выбирает только те группы, в которых количество строк равно общему количеству товаров в таблице purchase_item.

Результатом запроса будут пары значений (supplier_code, item), для каждой из которых соответствующий поставщик поставляет товар по минимальной цене.

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

SELECT

pi.item,

pi.purchase_price

FROM

warehouse.purchase_item pi

JOIN (

SELECT

```

        item,

        COUNT(DISTINCT purchase_price) AS price_count

FROM

        warehouse.purchase_item

GROUP BY

        item

    ) price_counts ON pi.item = price_counts.item

WHERE

        price_counts.price_count = 1

GROUP BY

        pi.item,

        pi.purchase_price

```

Этот запрос объединяет таблицу `purchase_item` саму с собой через подзапрос, чтобы найти все товары, для которых цена у всех поставщиков одинакова. Затем запрос фильтрует результаты, оставляя только те, для которых количество уникальных цен равно 1. Наконец, запрос группирует результаты по названию товара и цене.

Результатом запроса будут пары значений (`item`, `purchase_price`), для каждой из которых цена у всех поставщиков одинакова.

Data Output			Сообщения	Notifications
	item integer	purchase_price integer		
1	292462	120		
2	332074	30		
3	992649	50		

- Чему равен общий суточный доход оптового склада за прошедший день?

```

SELECT      sale.order_date,      SUM(sales_item.price_for_sale      *
sales_item.product_quantity) as daily_income

```

```
FROM warehouse.sale
```

```
JOIN warehouse.sales_item ON sale.lot_number = sales_item.lot_number
```

```
GROUP BY sale.order_date;
```

В этом запросе мы выбираем дату заказа (`order_date`) из таблицы `sale` и суммируем произведение цены на продажу (`price_for_sale`) и количества товара (`product_quantity`) из таблицы `sales_item` с помощью функции `SUM`. Затем мы группируем результаты по дате заказа с помощью функции `GROUP BY`. Результатом будет таблица, содержащая даты заказов и общий суточный доход склада за каждый день.

Data Output			Сообщения	Notifications
	order_date date	daily_income bigint		
1	2021-06-01	2100		
2	2023-01-22	11440		
3	2022-11-08	12960		

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

```
SELECT sales_item.delivered_item, SUM(sales_item.price_for_sale *  
sales_item.product_quantity) AS total_cost
```

```
FROM warehouse.sales_item
```

```
GROUP BY sales_item.delivered_item;
```

В этом запросе мы выбираем столбец `delivered_item` из таблицы `sales_item` и суммируем произведение цены на продажу (`price_for_sale`) и количества товара (`product_quantity`) с помощью функции `SUM`. Затем мы группируем результаты по виду товара с помощью функции `GROUP BY`. Результатом будет таблица, содержащая виды товаров и общую стоимость каждого вида товара, находящегося на базе.

	delivered_item [PK] integer	total_cost bigint
1	554532	11440
2	465232	12960
3	249231	2100

- В какой день было вывезено минимальное количество товара?

```
SELECT export_date, SUM(product_quantity) AS total_quantity

FROM warehouse.sales_item

INNER JOIN warehouse.sale ON sales_item.lot_number =
sale.lot_number

GROUP BY export_date

ORDER BY total_quantity ASC

LIMIT 1;
```

Этот запрос объединяет две таблицы, sales_item и sales, используя ключ lot_number, а затем группирует результаты по дате вывоза (export_date) и суммирует количество товара (product_quantity). Результаты сортируются по возрастанию суммарного количества товара и выбирается первая запись с помощью LIMIT 1, что соответствует дню с минимальным количеством товара.

Data Output			Сообщения	Notifications
	export_date date	total_quantity bigint		
1	2021-07-07	60		

- Сколько различных видов товара имеется на базе?

```
SELECT item_type FROM warehouse.item
```

Data Output		Сообщения	Notifications
	item_type character varying (30)		
1	Canned fish		
2	Confectionery products		
3	Fresh vegetables		

2. Создание представлений

- количество заказов фирм-покупателей за прошедший год

```
CREATE VIEW num_orders_last_year AS
```

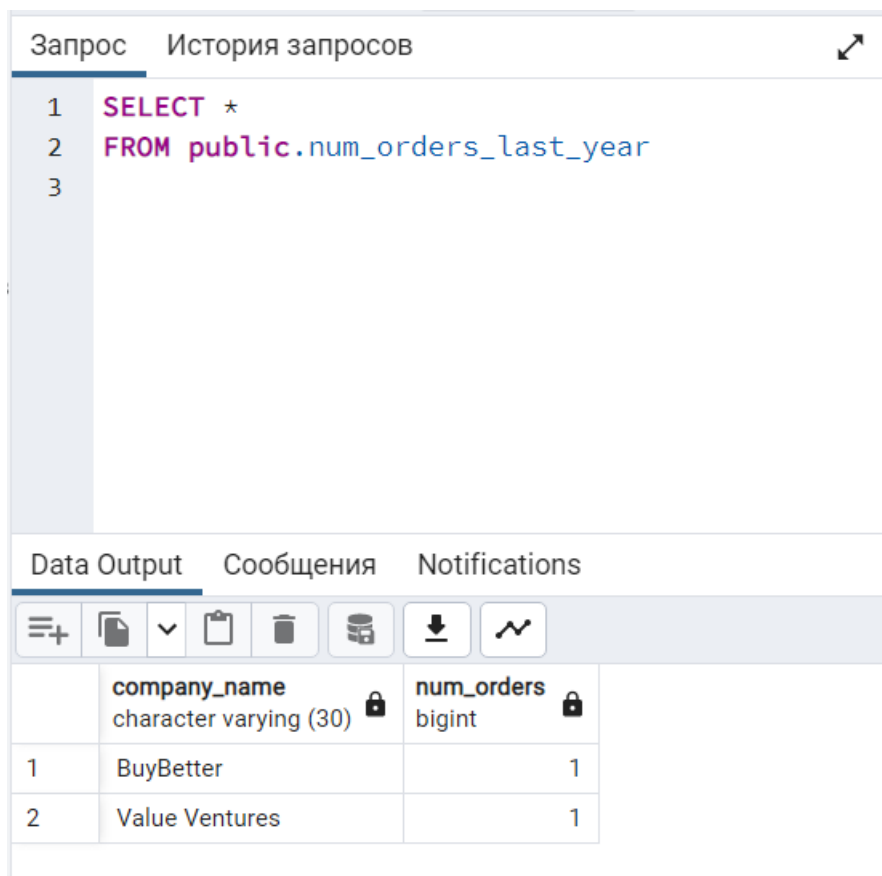
```
SELECT buyer_company.company_name, COUNT(sale.lot_number) AS  
num_orders
```

```
FROM warehouse.buyer_company
```

```
INNER JOIN warehouse.sale ON buyer_company.organization_code =  
sale.organization_code
```

```
WHERE sale.order_date BETWEEN DATE_TRUNC('year', NOW() -  
INTERVAL '1 year') AND NOW()
```

```
GROUP BY buyer_company.company_name;
```



The screenshot shows a database query tool interface. At the top, there are tabs for 'Запрос' (Query) and 'История запросов' (Query History). The 'Запрос' tab is active, displaying a SQL query with line numbers 1, 2, and 3. Below the query editor, there are tabs for 'Data Output', 'Сообщения' (Messages), and 'Notifications'. The 'Data Output' tab is active, showing a table with two columns: 'company_name' (character varying (30)) and 'num_orders' (bigint). The table contains two rows of data: 'BuyBetter' with 1 order and 'Value Ventures' with 1 order. The interface also includes a toolbar with icons for various actions like copy, paste, and execute.

	company_name character varying (30)	num_orders bigint
1	BuyBetter	1
2	Value Ventures	1

- доход базы за конкретный период.

```
CREATE VIEW revenue_2021 AS
```

```
SELECT sale.order_date, SUM(sales_item.price_for_sale *  
sales_item.product_quantity) AS daily_income
```

```
FROM warehouse.sale
```

```

JOIN warehouse.sales_item ON sale.lot_number = sales_item.lot_number

WHERE sale.order_date >= '2021-01-01' AND sale.order_date <= '2021-
12-31'

GROUP BY sale.order_date;

```

Запрос		История запросов	
1	SELECT *		
2	FROM public.revenue_2021		
3			

Data Output		Сообщения		Notifications	
	order_date date		daily_income bigint		
1	2021-06-01		2100		

3. Запросы на модификацию данных с использованием подзапросов

- **INSERT.** Добавим новый товар 'Cottage cheese' в таблицу item.

```

INSERT INTO warehouse.item (item_code, item_type,
unit_measurement, expiration_date, name, description)

```

```

SELECT '777776', 'Fermented milk products', 'Grams', '2023-06-11',
'Cottage cheese', 'Cottage cheese is a fresh cheese curd product with a mild
flavor. It is made from the curds of cows milk, and is a good source of
protein and calcium.'

```

```

WHERE NOT EXISTS (SELECT 1 FROM warehouse.item WHERE
item_code= '777776');

```

ЗапросИстория запросов

1

INSERT INTO warehouse.item (item_code, item_ty

2

SELECT '777776', 'Fermented milk products', 'G

3

WHERE NOT EXISTS (SELECT 1 FROM warehouse.item

4

Data OutputСообщенияNotifications

INSERT 0 1

Запрос завершён успешно, время выполнения: 48 мсес.

- **UPDATE.** Увеличим цену доставленного товара с id=249231 вдвое.

До:

	lot_number integer	delivered_item [PK] integer	price_for_sale integer
1	427492	249231	35

UPDATE warehouse.sales_item

SET price_for_sale = price_for_sale * 2

WHERE delivered_item = (SELECT delivered_item FROM warehouse.sales_item WHERE delivered_item = '249231');

ЗапросИстория запросов

```

1 UPDATE warehouse.sales_item
2 SET price_for_sale = price_for_sale * 2
3 WHERE delivered_item = (SELECT delivered_item

```

Data OutputСообщенияNotifications

UPDATE 1

Запрос завершён успешно, время выполнения: 45 msec.

После:

	lot_number integer	delivered_item [PK] integer	price_for_sale integer
1	427492	249231	70

- **DELETE.** Удалить склад, у которого остаток товара меньше 10.

До:

	storage_code [PK] integer	item_residue integer	location character varying (100)
1	111111	35	Russia, Rostov-on-Don, Zorge 52, 344103
2	222222	4356	Russia, Yamalo-Nenets Autonomous okrug (YANA)
3	333333	12	Russia, Khanty-Mansi Autonomous Okrug (Yugra),
4	898989	5	Russia, Grozny, Baltiyskaya 12, 364020

DELETE FROM warehouse.storage

WHERE item_residue < (

SELECT 10

);

Запрос

История запросов

↗

1

DELETE FROM warehouse.storage

2

WHERE item_residue < (

3

SELECT 10

4

);

Data Output

Сообщения

Notifications

DELETE 1

Запрос завершён успешно, время выполнения: 48 msec.

После:

	storage_code [PK] integer	item_residue integer	location character varying (100)
1	111111	35	Russia, Rostov-on-Don, Zorge 52, 3-
2	222222	4356	Russia, Yamalo-Nenets Autonomous
3	333333	12	Russia, Khanty-Mansi Autonomous

4. Создание простого и составного индекса. Сравнение времени выполнения запросов с индексами и без.

- Простой индекс

Запрос без индекса:

Запрос История запросов

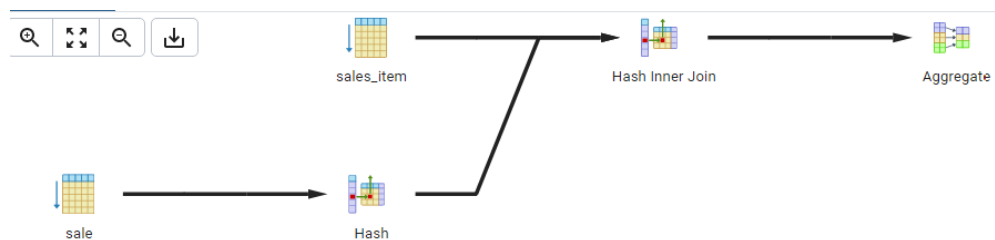
```

1 SELECT sale.order_date, SUM(sales_item.price_for_sale * s
2 FROM warehouse.sale
3 JOIN warehouse.sales_item ON sale.lot_number = sales_item
4 GROUP BY sale.order_date;

```

Data Output Сообщения План выполнения × Notifications

Запрос выполнен успешно. Общее время выполнения: 73 msec.
обработано строк: 3.



Создание индекса:

```
CREATE INDEX idx_lot_number ON warehouse.sales_item
(lot_number);
```

Запрос История запросов

```

1 CREATE INDEX idx_lot_number ON warehouse.sales_item(lot_n

```

Data Output Сообщения План выполнения × Notifications

CREATE INDEX

Запрос завершён успешно, время выполнения: 57 msec.

Запрос с индексом:

Запрос История запросов

```
1 SELECT sale.order_date, SUM(sales_item.price_for_sale * si
2 FROM warehouse.sale
3 JOIN warehouse.sales_item ON sale.lot_number = sales_item
4 GROUP BY sale.order_date;
```

Data Output Сообщения План выполнения × Notifications

Запрос выполнен успешно. Общее время выполнения: 65 msec.
обработано строк: 3.

- **Составной индекс**

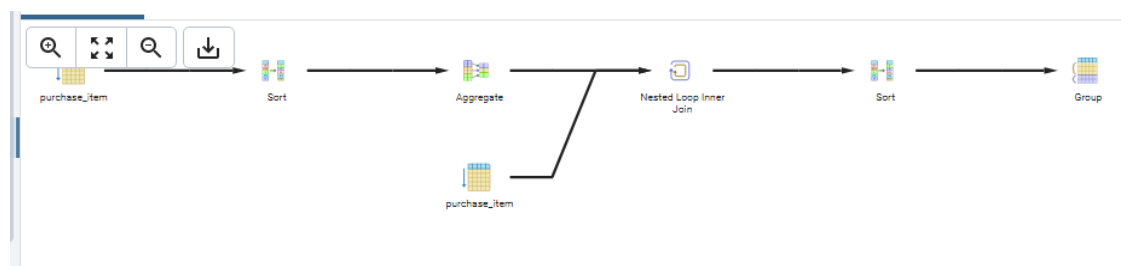
Запрос без индекса:

Запрос История запросов

```
1 SELECT
2     pi.item,
3     pi.purchase_price
4 FROM
5     warehouse.purchase_item pi
6 JOIN (
7     SELECT
8         item,
9         COUNT(DISTINCT purchase_price) AS price_count
10    FROM
11        warehouse.purchase_item
```

Data Output Сообщения Notifications

Запрос выполнен успешно. Общее время выполнения: 98 msec.
обработано строк: 3.



Создание индекса:

```
CREATE INDEX idx_purchase_item_item_price_count
ON warehouse.purchase_item (item, purchase_price);
```

```
1 CREATE INDEX idx_purchase_item_item_price_count
2 ON warehouse.purchase_item (item, purchase_price);
```

Data Output **Сообщения** Notifications

CREATE INDEX

Запрос завершён успешно, время выполнения: 72 мсек.

Запрос с индексом:

Запрос История запросов

```
1 SELECT
2 pi.item,
3 pi.purchase_price
4 FROM
5 warehouse.purchase_item pi
6 JOIN (
7 SELECT
8 item,
9 COUNT(DISTINCT purchase_price) AS price_count
10 FROM
11 warehouse.purchase_item
```

Data Output **Сообщения** Notifications

Запрос выполнен успешно. Общее время выполнения: 63 мсек.
обработано строк: 3.

Удаление индексов:

DROP INDEX warehouse.idx_lot_number

DROP INDEX warehouse.idx_purchase_item_item_price_count

Вывод: созданы запросы на выборку данных в PostgreSQL с помощью pgAdmin, составлены запросы на модификацию данных с использованием подзапросов. Созданы простой и составной индексы для двух запросов, которые, как показало сравнение, немного ускорили выполнение.