

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»

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

Автор: Евдокимова У.В.

Факультет: ИКТ

Группа: К3240

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, 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.

1. Создать запросы:

Список всех 2-местных номеров отелей, с ценой менее 200 т.р., упорядочив данные в порядке уменьшения стоимости

```
SELECT rt.type_code, rt.capacity, r.hotel_code, p.price
FROM public."Room" r
INNER JOIN public."Room Type" rt ON r.type_code = rt.type_code
INNER JOIN public."Price" p ON rt.type_code = p.type_code
WHERE rt.capacity = 2
      AND p.price::numeric < 200
ORDER BY p.price DESC;
```

Query Query History

```
1 SELECT rt.type_code, rt.capacity, r.hotel_code, p.price
2 FROM public."Room" r
3 INNER JOIN public."Room Type" rt ON r.type_code = rt.type_code
4 INNER JOIN public."Price" p ON rt.type_code = p.type_code
5 WHERE rt.capacity = 2
6       AND p.price::numeric < 200
7 ORDER BY p.price DESC;
```

Data Output Messages Notifications

	type_code integer	capacity integer	hotel_code integer	price money
1	2	2	1	150,00 ?
2	2	2	1	150,00 ?
3	2	2	2	150,00 ?

Все записи регистрации постояльцев, которые выехали из отелей в течение двух последних недель.

```
SELECT G.*, O.end_of_stay, O.start_of_stay
FROM public."Guest" AS G
JOIN public."Order" AS O ON G.guest_pasport_details = O.guest_pasport_details
WHERE O.end_of_stay >= CURRENT_DATE - INTERVAL '2 weeks';
```


Список свободных номеров одного из отелей на текущий день

```
SELECT DISTINCT r.room_number
FROM public."Room" r
LEFT JOIN public."Order" o ON o.id_room = r.id_room
WHERE r.hotel_code = 1
AND r.cleaning_status = 'Clean'
AND r.occupancy_status = 'Vacant'
AND ( o.start_of_stay > current_date OR o.end_of_stay <= current_date)
```

Query Query History

```
1 SELECT DISTINCT r.room_number
2 FROM public."Room" r
3 LEFT JOIN public."Order" o ON o.id_room = r.id_room
4 WHERE r.hotel_code = 1
5 AND r.cleaning_status = 'Clean'
6 AND r.occupancy_status = 'Vacant'
7 AND ( o.start_of_stay > current_date OR o.end_of_stay <= current_date)
```

Data Output Messages Notifications

	room_number [PK] integer
1	1001
2	1003

Общие потери от незанятых номеров за текущий день по всей сети

```
SELECT SUM(p.price) AS total_loss
FROM public."Price" p
LEFT JOIN public."Room" r ON p.type_code = r.type_code
LEFT JOIN public."Order" o ON r.id_room = o.id_room
WHERE o.start_of_stay IS NULL OR o.start_of_stay > NOW() OR o.order_number = 1;
```

Query Query History

```

1 SELECT SUM(p.price) AS total_loss
2 FROM public."Price" p
3 LEFT JOIN public."Room" r ON p.type_code = r.type_code
4 LEFT JOIN public."Order" o ON r.id_room = o.id_room
5 WHERE o.start_of_stay IS NULL OR o.start_of_stay > NOW() OR o.order_number = 1;

```

Data Output Messages Notifications

	total_loss money
1	800,00 ?

В каком отеле имеется наибольшее количество незанятых номеров на текущие сутки

```

SELECT h.name, COUNT(r.room_number) AS available_rooms
FROM public."Hotel" h
JOIN public."Room" r ON h.hotel_code = r.hotel_code
LEFT JOIN public."Order" o ON r.id_room = o.id_room
WHERE o.start_of_stay IS NULL OR o.start_of_stay > NOW()
GROUP BY h.name
ORDER BY available_rooms DESC

```

Query Query History

```

1 SELECT h.name, COUNT(r.room_number) AS available_rooms
2 FROM public."Hotel" h
3 JOIN public."Room" r ON h.hotel_code = r.hotel_code
4 LEFT JOIN public."Order" o ON r.id_room = o.id_room
5 WHERE o.start_of_stay IS NULL OR o.start_of_stay > NOW()
6 GROUP BY h.name
7 ORDER BY available_rooms DESC
8

```

Data Output Messages Notifications

	name character varying (50)	available_rooms bigint
1	Hotel A	2
2	Hotel B	2

Самый популярный тип номеров за последний год.

```

SELECT r.type_code, COUNT(o.order_number) AS total_bookings
FROM public."Room" r
JOIN public."Order" o ON r.id_room = o.id_room

```

```
WHERE o.start_of_stay >= NOW() - INTERVAL '1 year'
GROUP BY r.type_code
ORDER BY total_bookings DESC
LIMIT 1;
```

Query Query History

```
1 SELECT r.type_code, COUNT(o.order_number) AS total_bookings
2 FROM public."Room" r
3 JOIN public."Order" o ON r.id_room = o.id_room
4 WHERE o.start_of_stay >= NOW() - INTERVAL '1 year'
5 GROUP BY r.type_code
6 ORDER BY total_bookings DESC
7 LIMIT 1;
```

Data Output Messages Notifications

	type_code integer	total_bookings bigint
1	1	2

Представления

Для турагентов: поиск свободных номеров в отелях

```
CREATE VIEW public."AvailableRooms" AS
SELECT r.id_room, r.room_number, r.type_code, h.name
FROM public."Room" r
JOIN public."Hotel" h ON r.hotel_code = h.hotel_code
WHERE r.id_room NOT IN (
    SELECT id_room
    FROM public."Order"
    WHERE start_of_stay <= NOW() AND end_of_stay >= NOW()
);
```

Query

Query History

1 SELECT * FROM public."AvailableRooms"

2










Data Output

Messages

Graph Visualiser

×

Notifications



	id_room integer	room_number integer	type_code integer	name character varying (50)
1	20	1002	2	Hotel A
2	30	1003	1	Hotel A
3	10	1001	1	Hotel A
4	3	103	3	Hotel A
5	1	101	1	Hotel A
6	4	104	4	Hotel B
7	2	102	2	Hotel B
8	40	1004	2	Hotel D

Для владельца компании : информация о доходах каждого отеля в сети за прошедший месяц

```
CREATE VIEW public."HotelRevenue" AS
SELECT h.hotel_code, h.name, SUM(o.order_cost) AS revenue
FROM public."Hotel" h
JOIN public."Order" o ON h.hotel_code = h.hotel_code
WHERE DATE_TRUNC('month', o.end_of_stay) = DATE_TRUNC('month', CURRENT_DATE -
INTERVAL '1 month')
GROUP BY h.hotel_code, h.name;
```

Query

Query History

1

2

SELECT * FROM public."HotelRevenue";

Data Output

Messages

Notifications

	hotel_code integer	name character varying (50)	revenue bigint
1	4	Hotel D	1500
2	2	Hotel B	1500
3	1	Hotel A	1500
4	3	Hotel C	1500

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

INSERT (добавление нового заказа)

```
INSERT INTO public."Order" (order_number, payment_status, booking_date,
start_of_stay, end_of_stay, order_cost, order_status, guest_pasport_details,
id_room)
SELECT
    MAX(order_number) + 1,
    10000,
    '2022-10-10',
    '2023-01-01',
    '2023-01-10',
    1000,
    'Completed',
    '123456',
    (
        SELECT id_room FROM public."Room"
        WHERE type_code = 1
        LIMIT 1
    )
FROM public."Order";
```

Query Query History

```
1 SELECT * FROM public."Order"
2 ORDER BY order_number ASC
```

Data Output Messages Notifications

	order_number [PK] integer	payment_status money	booking_date date	start_of_stay date	end_of_stay date	order_cost integer	order_status character varying (10)	guest_pasport_details character varying (100)	id_room integer
1	1	100,00 ?	2023-10-01	2023-10-10	2023-10-15	500	In Process	AB123456	
2	2	150,00 ?	2023-11-05	2023-11-10	2023-11-15	750	Completed	CD987654	
3	101	100,00 ?	2022-01-10	2022-01-20	2022-01-30	1000	Completed	123456	
4	102	200,00 ?	2022-02-10	2022-02-20	2022-02-25	1200	Completed	789101	
5	103	0,00 ?	2022-03-15	2022-03-20	2022-03-30	1500	Pending	112131	
6	104	150,00 ?	2022-04-10	2022-04-15	2022-04-22	1800	Completed	112133	
7	105	10 000,00 ?	2022-10-10	2023-01-01	2023-01-10	1000	Completed	123456	

UPDATE (изменение паспортных данных гостя)

```
UPDATE public."Guest"
SET registration = '02-Mar-2021'
WHERE guest_pasport_details IN (
    SELECT guest_pasport_details
    FROM public."Guest"
    WHERE full_name = 'John Doe'
);
```

Query

Query History

1SELECT * FROM public."Guest"

2ORDER BY guest_pasport_details ASC

Data Output

Messages

Notifications

	guest_pasport_details [PK] character varying (100)	full_name character varying (50)	registration character varying (50)	phone_number integer	e-mail character varying (30)
1	112131	Alice Johnson	02-Mar-2021	1234567891	alicejohnson@email.com
2	112133	Joe Johnson	02-Mar-2021	1234567899	joejohnson@email.com
3	123456	John Doe	02-Mar-2021	1234567890	johndoe@email.com
4	789101	Jane Smith	15-Feb-2021	1876543210	janesmith@email.com
5	AB123456	John Doe	02-Mar-2019	1234567890	john@example.com
6	CD987654	Jane Smith	Los Angeles	1176543210	jane@example.com

DELETE (удаление акции из списка доступных)

```
DELETE FROM public."Sale"
WHERE sale_code IN (
  SELECT sale_code
  FROM public."Sale" s
  WHERE sale_end < '2023-04-03'
);
```

Query

Query History

1

SELECT * FROM public."Sale"

2

ORDER BY sale_code ASC

Data Output

Messages

Notifications

	sale_code [PK] integer	type_code integer	sale_start date	sale_end date	sale_size integer
1	3	3	2023-03-01	2023-04-21	20
2	4	4	2023-04-01	2023-04-30	25

4. Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами.

Гости, которые забронировали номер по акции

Простой индекс:

```
CREATE INDEX guest_booking_date_idx ON "Order" (booking_date);
```

Составной индекс:

```
CREATE INDEX guest_booking_sale_idx ON "Order" (booking_date,  
guest_pasport_details);
```

Запрос без индексов:

Data Output Messages Explain × Notifications

Graphical Analysis Statistics

```
graph LR; Order[Order] --> Materialize[Materialize]; Sale[Sale] --> Join[Nested Loop Inner Join]; Materialize --> Join; Join --> Aggregate[Aggregate]
```

Query Query History

```
1 SELECT DISTINCT o.guest_pasport_details
2 FROM public."Order" o
3 JOIN public."Sale" s ON o.booking_date >= s.sale_start AND o.booking_date <= s.sale_end
```

Data Output Messages Notifications

Successfully run. Total query runtime: 706 msec.
2 rows affected.

С индексами:

Query Query History

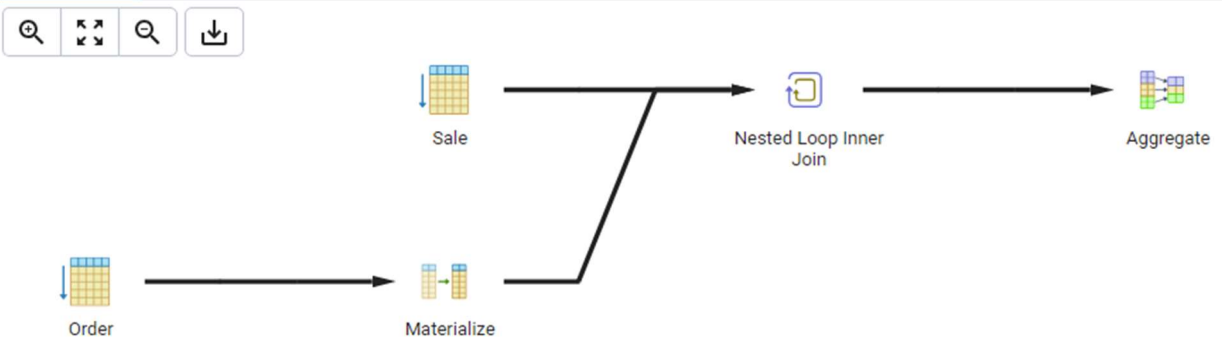
```
1 SELECT DISTINCT o.guest_pasport_details
2 FROM public."Order" o
3 JOIN public."Sale" s ON o.guest_pasport_details = o.guest_pasport_details
4 WHERE o.booking_date >= s.sale_start AND o.booking_date <= s.sale_end
```

Data Output Messages Notifications

Successfully run. Total query runtime: 630 msec.
2 rows affected.

Data Output Messages Explain × Notifications

Graphical Analysis Statistics



Список самых популярных за год номеров по сети

Простой индекс

```
CREATE INDEX idx_order_id_room ON public."Order" (id_room);
```

Составной индекс

```
CREATE INDEX idx_order_hotel_code_booking_date ON public."Order" (hotel_code, booking_date);
```

Запрос без индексов

Query Query History

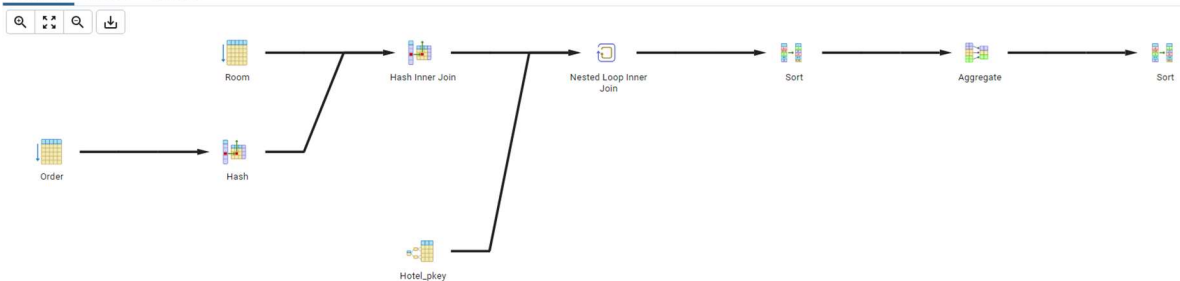
```
1 SELECT r.room_number, h.name, r.hotel_code, COUNT(*) AS booking_count
2 FROM public."Room" r
3 JOIN public."Order" o ON r.id_room = o.id_room
4 JOIN public."Hotel" h ON r.hotel_code = h.hotel_code
5 WHERE booking_date >= '2022-01-01' AND booking_date <= '2022-12-31'
6 GROUP BY r.room_number, r.hotel_code, h.name
7 ORDER BY booking_count DESC;
8
```

Data Output Messages Notifications

Successfully run. Total query runtime: 188 msec.
4 rows affected.

Data Output Messages Explain x Notifications

Graphical Analysis Statistics



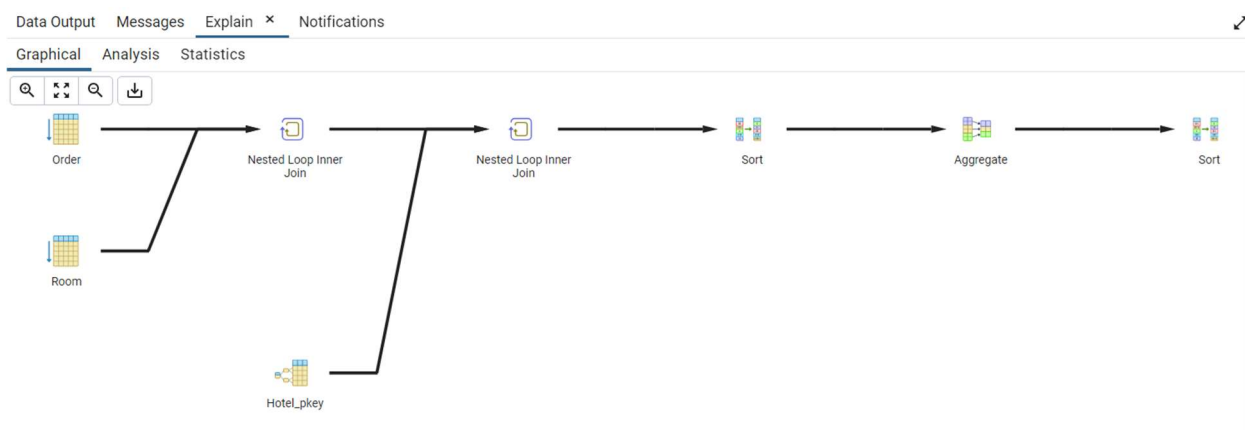
С индексами

Query Query History

```
1 SELECT r.room_number, h.name, r.hotel_code, COUNT(*) AS booking_count
2 FROM public."Room" r
3 JOIN public."Order" o ON r.id_room = o.id_room
4 JOIN public."Hotel" h ON r.hotel_code = h.hotel_code
5 WHERE booking_date >= '2022-01-01' AND booking_date <= '2022-12-31'
6 GROUP BY r.room_number, r.hotel_code, h.name
7 ORDER BY booking_count DESC;
```

Data Output Messages Notifications

Successfully run. Total query runtime: 82 msec.
4 rows affected.



5. Вывод

В ходе лабораторной работы я научилась работе с различными SQL-запросами к базе данных, а также созданию представлений и индексов. После сравнения времени выполнения запросов, выяснилось, что запросы с индексами выполняются быстрее