

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет прикладной информатики
Образовательная программа Мобильные и сетевые технологии
Направление подготовки 09.03.03 Мобильные и сетевые технологии

ОТЧЕТ по Лабораторной работе № 4
по дисциплине «Проектирование и реализация баз данных»

Обучающийся: Шукалов Андрей Денисович
Преподаватель: Говорова Марина Михайловна

Санкт-Петербург, 2025

Описание задания

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 лабораторной работы №2, 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.

Схема базы данных

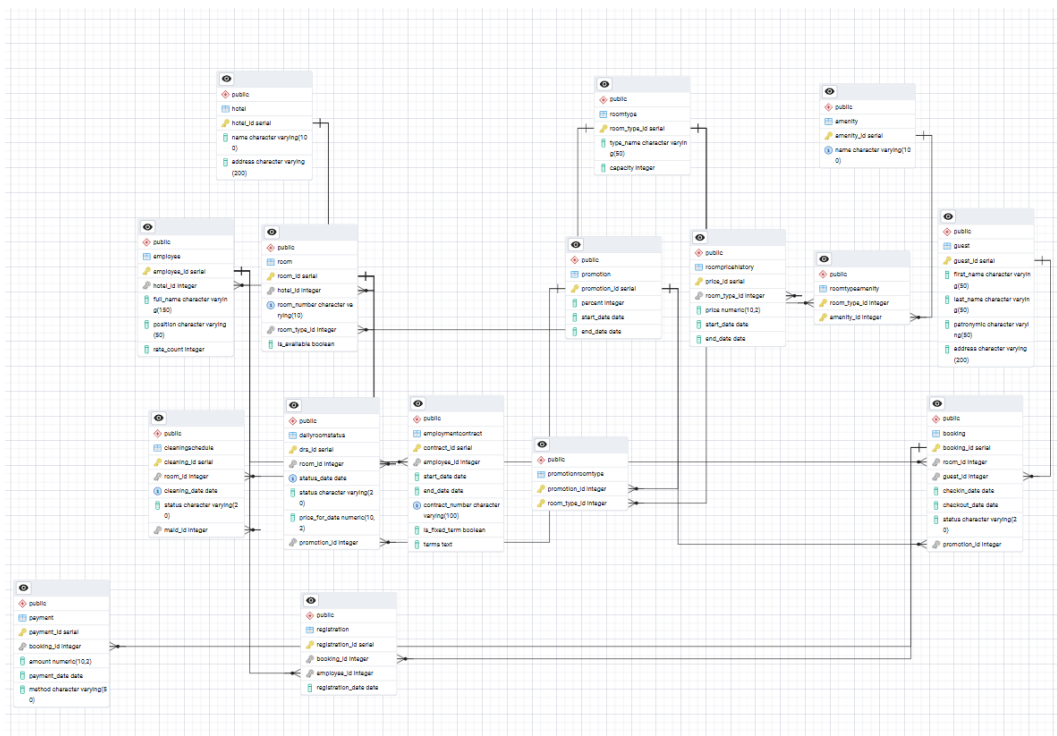


Рис. 1: ER диаграмма

Ход работы

1 Запросы к базе данных PostgreSQL

Запрос 1

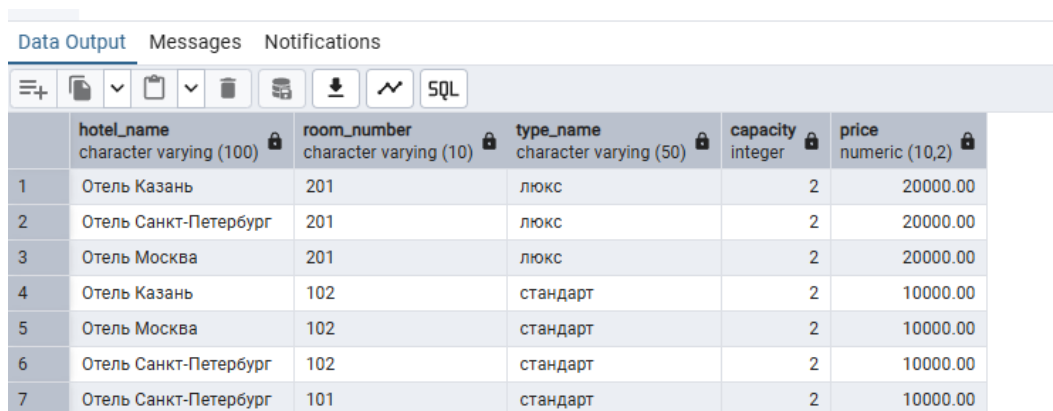
Составить список всех 2-местных номеров отелей, с ценой менее 200 т.р., упорядочив данные в порядке уменьшения стоимости.

```
SELECT h.name AS hotel_name, r.room_number, rt.type_name, rt.capacity, ph.price
FROM Room r
JOIN Hotel h ON r.hotel_id = h.hotel_id
JOIN RoomType rt ON r.room_type_id = rt.room_type_id
JOIN RoomPriceHistory ph ON ph.room_type_id = rt.room_type_id
    AND ph.start_date <= current_date
    AND (ph.end_date IS NULL OR ph.end_date >= current_date)
WHERE rt.capacity = 2
    AND ph.price < 200000
ORDER BY ph.price DESC;
```

Соединяем Hotel, RoomType и RoomPriceHistory, среди последних берёт только те, у которых в промежуток цены попадает текущая дата.

С помощью WHERE ставим ограничение на размер номера равный 2 и цену меньше 200 т.р.

С помощью ORDER BY сортируем запросы по уменьшению стоимости.



	hotel_name character varying (100)	room_number character varying (10)	type_name character varying (50)	capacity integer	price numeric (10,2)
1	Отель Казань	201	люкс	2	20000.00
2	Отель Санкт-Петербург	201	люкс	2	20000.00
3	Отель Москва	201	люкс	2	20000.00
4	Отель Казань	102	стандарт	2	10000.00
5	Отель Москва	102	стандарт	2	10000.00
6	Отель Санкт-Петербург	102	стандарт	2	10000.00
7	Отель Санкт-Петербург	101	стандарт	2	10000.00

Рис. 2: Результат запроса 1

Запрос 2

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

```
SELECT reg.*  
FROM Registration reg  
JOIN Booking b ON reg.booking_id = b.booking_id  
WHERE b.checkout_date BETWEEN (current_date - INTERVAL '14 days') AND current_date;
```

Объединим Registration и Booking.

Выбираем регистрации такие, что дата их выезда попадает в интервал двух недель.

Data Output Messages Notifications				
	registration_id [PK] integer	booking_id integer	employee_id integer	registration_date date
1	1	4	2	2025-09-25
2	2	6	4	2025-09-26
3	3	5	5	2025-09-28

Рис. 3: Результат запроса 2

Запрос 3

Чему равен общий суточный доход каждого отеля за последний месяц?

```
SELECT
    h.name AS hotel_name,
    COALESCE(SUM(drs.price_for_date), 0) AS total_revenue
FROM Hotel h
LEFT JOIN Room r ON r.hotel_id = h.hotel_id
LEFT JOIN DailyRoomStatus drs
    ON drs.room_id = r.room_id
    AND drs.status = 'occupied'
    AND drs.status_date >= current_date - INTERVAL '1 month'
    AND drs.status_date <= current_date
GROUP BY h.name;
```

Объединяем DailyRoomStatus, Room и Hotel, суммируем по drs.price_for_date, где статус у комнаты занят и дата попадает за последний месяц.

Воспользуемся LEFT JOIN, чтобы получить данные просуммировать по пустым отелям.

Data Output Messages Notifications		
<div><div>≡+</div><div></div><div>▼</div><div></div><div>▼</div><div></div><div></div><div></div><div></div><div>SQL</div></div>		
	hotel_name character varying (100)	total_revenue numeric
1	Отель Москва	9000.00
2	Отель Казань	0
3	Отель Санкт-Петербург	0

Рис. 4: Результат запроса 3

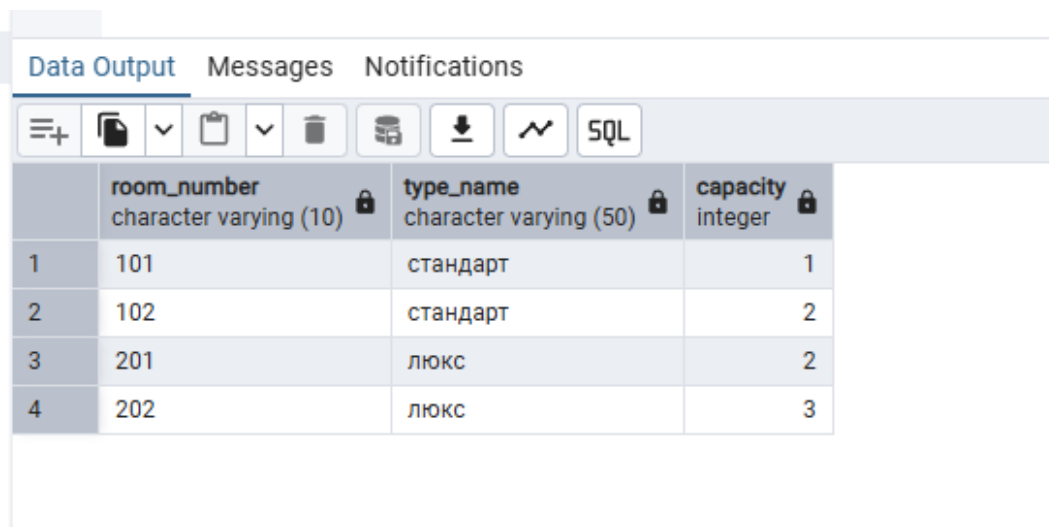
Запрос 4

Составить список свободных номеров заданного отеля на текущий день.

```
SELECT r.room_number, rt.type_name, rt.capacity
FROM Room r
JOIN RoomType rt ON r.room_type_id = rt.room_type_id
LEFT JOIN DailyRoomStatus drs ON drs.room_id = r.room_id AND drs.status_date = current_date
-- Возьмем заданный отель с индексом 1.
WHERE r.hotel_id = 1
      AND (drs.status = 'free' OR drs.status IS NULL);
```

Объединяем Room И RoomType по room_type_id, также объединяем DailyRoomStatus с помощью LEFT JOIN, чтобы не пропускать строки, где нету записей в drs.

Фильтруем по заданному индексу отеля (например, 1) и чтобы статус комнаты был 'free'.



The screenshot shows a database query result interface. At the top, there are tabs for 'Data Output', 'Messages', and 'Notifications'. Below the tabs is a toolbar with icons for various actions like expand, copy, paste, delete, and download. The main area displays a table with the following data:

	room_number character varying (10)	type_name character varying (50)	capacity integer
1	101	стандарт	1
2	102	стандарт	2
3	201	люкс	2
4	202	люкс	3

Рис. 5: Результат запроса 4

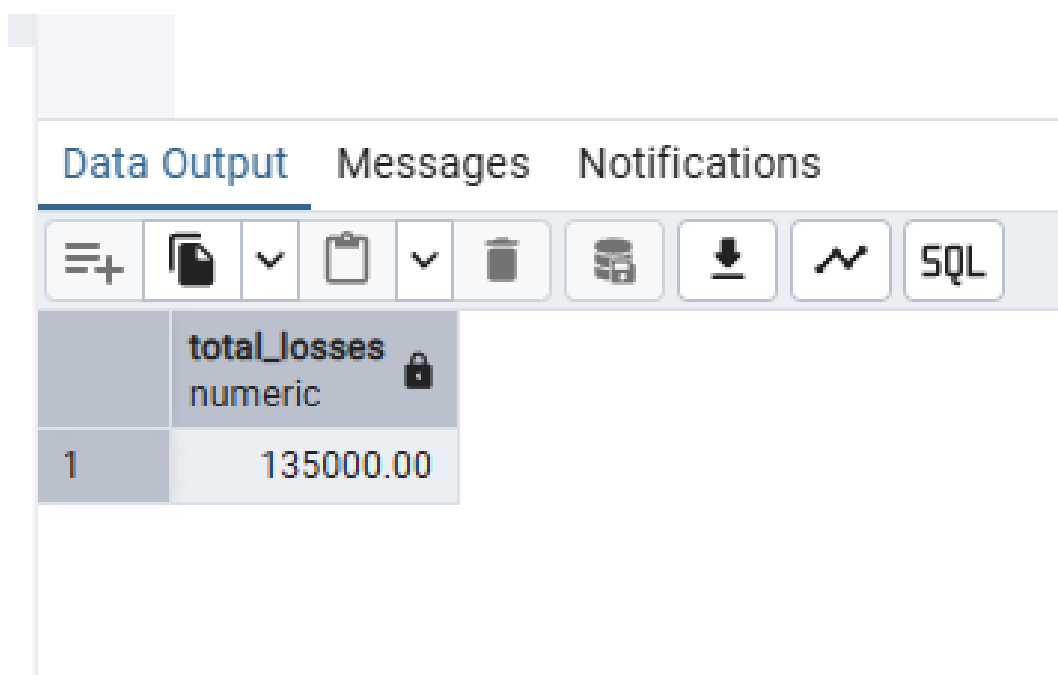
Запрос 5

Найти общие потери от незанятых номеров за текущий день по всей сети.

```
SELECT SUM(COALESCE(drs.price_for_date, ph.price)) AS total_losses
FROM Room r
LEFT JOIN DailyRoomStatus drs ON drs.room_id = r.room_id AND drs.status_date = current_date
LEFT JOIN RoomPriceHistory ph ON ph.room_type_id = r.room_type_id
    AND ph.start_date <= current_date
    AND (ph.end_date IS NULL OR ph.end_date >= current_date)
WHERE drs.status = 'free' OR drs.status IS NULL;
```

Объединяем Room с DailyRoomStatus и RoomPriceHistory, используя LEFT JOIN чтобы взять все значения комнат, даже тех, о которых нет записей в двух последних таблицах.

Суммируем все эти комнаты, если нету записи в DailyRoomStatus, то используем стандартную цену комнаты из RoomPriceHistory.



The screenshot shows a database query result interface. At the top, there are three tabs: "Data Output" (selected), "Messages", and "Notifications". Below the tabs is a toolbar with icons for various actions: expand/collapse, copy, paste, delete, save, download, and a "SQL" button. The main area displays a table with the following data:

	total_losses numeric
1	135000.00

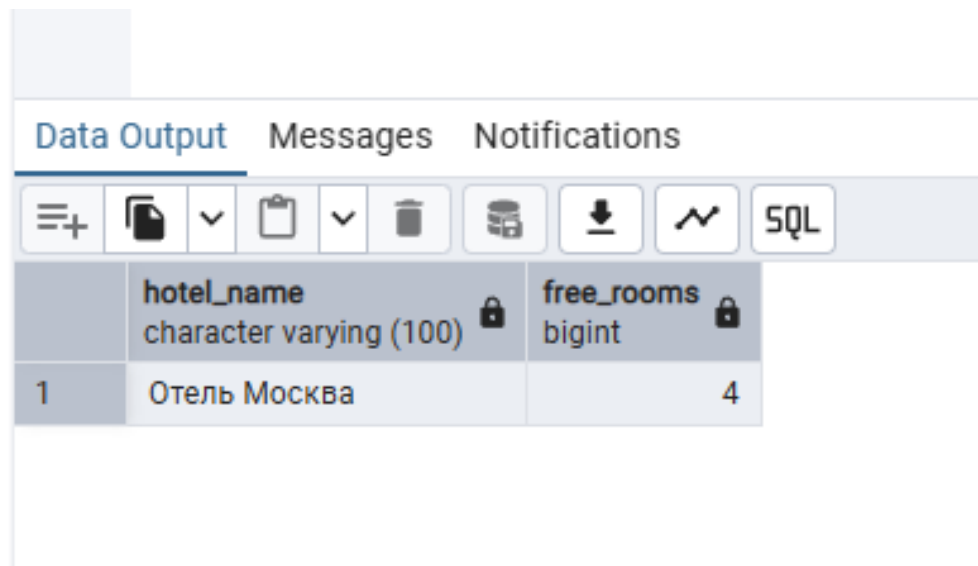
Рис. 6: Результат запроса 5

Запрос 6

Определить, в каком отеле имеется наибольшее количество незанятых номеров на текущие сутки.

```
SELECT h.name AS hotel_name, COUNT(r.room_id) AS free_rooms
FROM Room r
JOIN Hotel h ON r.hotel_id = h.hotel_id
LEFT JOIN DailyRoomStatus drs ON drs.room_id = r.room_id AND drs.status_date = current_date
WHERE drs.status = 'free' OR drs.status IS NULL
GROUP BY h.name
ORDER BY free_rooms DESC
LIMIT 1;
```

Группируем Room, Hotel по hotel_id, делаем LEFT JOIN с DailyRoomStatus, чтобы узнать статус каждой комнаты по отдельности, считаем количество таких комнат в free_rooms, сортируем по убыванию и берём только 1 запись.



	hotel_name character varying (100)	free_rooms bigint
1	Отель Москва	4

Рис. 7: Результат запроса 6

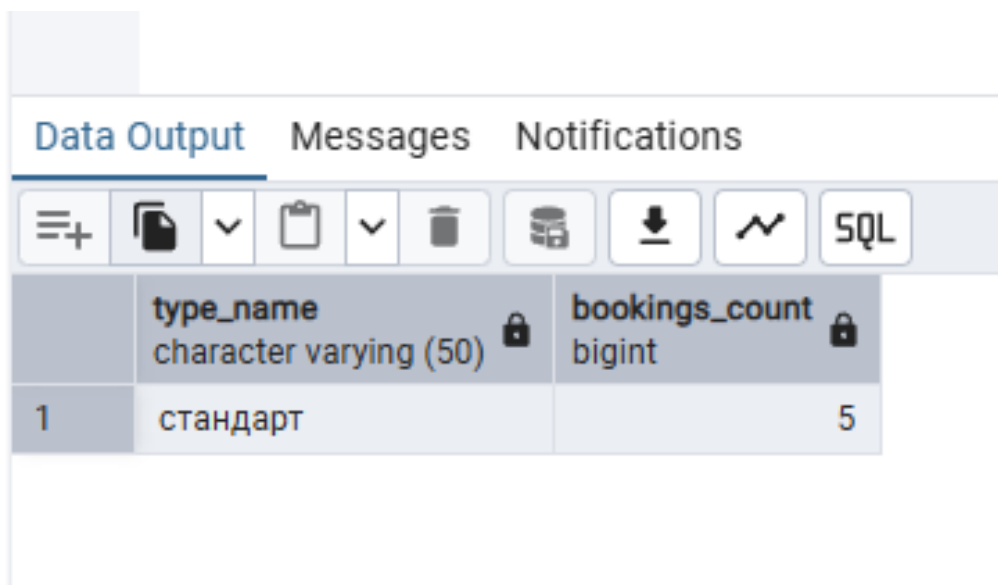
Запрос 7

Определить самый популярный тип номеров за последний год.

```
SELECT rt.type_name, COUNT(*) AS bookings_count
FROM Booking b
JOIN Room r ON b.room_id = r.room_id
JOIN RoomType rt ON r.room_type_id = rt.room_type_id
WHERE b.checkin_date BETWEEN (current_date - INTERVAL '1 year') AND current_date
GROUP BY rt.type_name
ORDER BY bookings_count DESC
LIMIT 1;
```

Рассматриваем все бронирования, группируем с комнатами и типами комнат.

Форматируем по условию, чтобы дата заселения была в течении последнего года, сортируем по количеству таких комнат и берём только 1 запись.



The screenshot shows a database query result interface. At the top, there are tabs for "Data Output", "Messages", and "Notifications". Below the tabs is a toolbar with various icons for actions like expand, copy, paste, delete, and download. The main area displays a table with the following structure:

	type_name character varying (50)	bookings_count bigint
1	стандарт	5

Рис. 8: Результат запроса 7

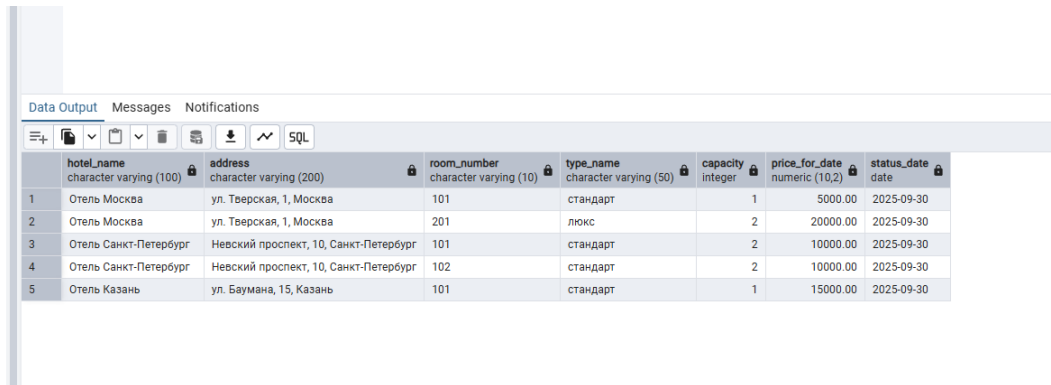
2 Представления

Представление 1

Для турагентов (поиск свободных номеров в отелях).

```
CREATE VIEW AvailableRoomsForAgents AS
SELECT h.name AS hotel_name, h.address, r.room_number, rt.type_name, rt.capacity,
       drs.price_for_date, drs.status_date
FROM DailyRoomStatus drs
JOIN Room r ON drs.room_id = r.room_id
JOIN RoomType rt ON r.room_type_id = rt.room_type_id
JOIN Hotel h ON r.hotel_id = h.hotel_id
WHERE drs.status = 'free' AND drs.status_date = current_date;
```

Создаём представление, выбирая среди всех записей DailyRoomStatus свободные комнаты на сегодняшнюю дату, объединяем с Room, RoomType и Hotel для получения названия отеля, адреса, номера комнаты, типа комнаты, цены на номер.



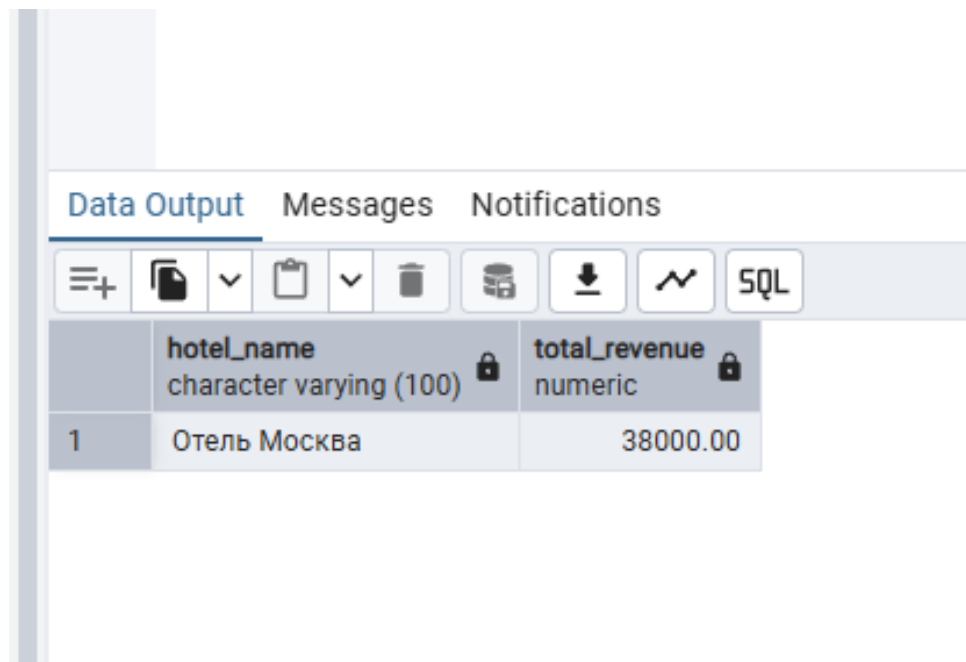
	hotel_name character varying (100)	address character varying (200)	room_number character varying (10)	type_name character varying (50)	capacity integer	price_for_date numeric (10,2)	status_date date
1	Отель Москва	ул. Тверская, 1, Москва	101	стандарт	1	5000.00	2025-09-30
2	Отель Москва	ул. Тверская, 1, Москва	201	люкс	2	20000.00	2025-09-30
3	Отель Санкт-Петербург	Невский проспект, 10, Санкт-Петербург	101	стандарт	2	10000.00	2025-09-30
4	Отель Санкт-Петербург	Невский проспект, 10, Санкт-Петербург	102	стандарт	2	10000.00	2025-09-30
5	Отель Казань	ул. Баумана, 15, Казань	101	стандарт	1	15000.00	2025-09-30

Рис. 9: Результат представления 1

Представление 2

Для владельца компании (информация о доходах каждого отеля в сети за прошедший месяц).

```
CREATE VIEW HotelMonthlyRevenue AS
WITH last_month AS (
    SELECT date_trunc('month', current_date) - INTERVAL '1 month' AS start_dt, -- дата
    начала прошлого месяца
           date_trunc('month', current_date) - INTERVAL '1 day' AS end_dt -- дата
    конца прошлого месяца
)
SELECT h.name AS hotel_name, SUM(drs.price_for_date) AS total_revenue -- суммируем
цены
FROM DailyRoomStatus drs
JOIN Room r ON drs.room_id = r.room_id
JOIN Hotel h ON r.hotel_id = h.hotel_id
JOIN last_month lm ON drs.status_date BETWEEN lm.start_dt AND lm.end_dt -- дата
статуса комнаты должна быть в прошлом месяце
WHERE drs.status = 'occupied' -- статус занято
GROUP BY h.name;
```



	hotel_name character varying (100)	total_revenue numeric
1	Отель Москва	38000.00

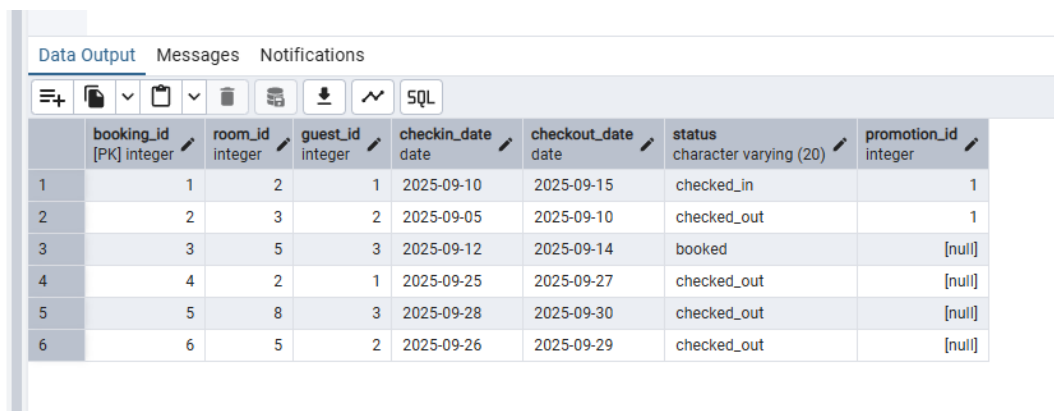
Рис. 10: Результат представления 2

3 Запросы на модификацию данных

3.1 Запрос INSERT

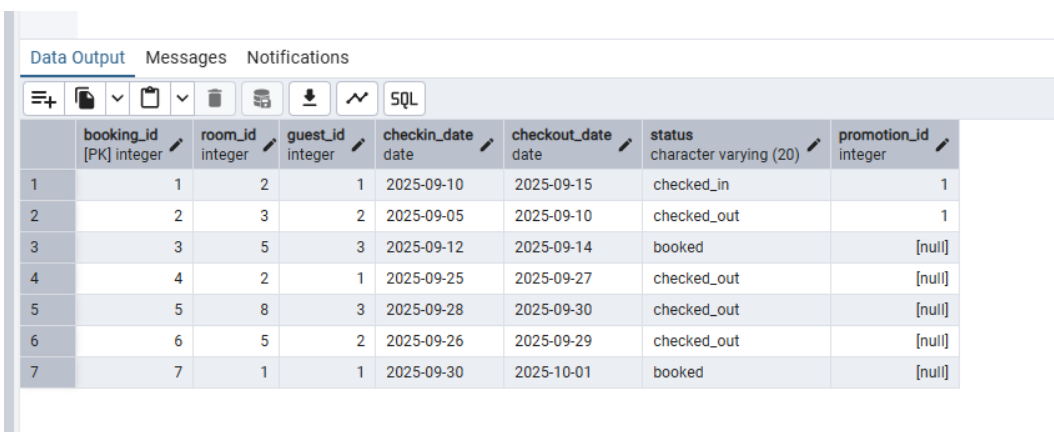
Бронирование любой свободной комнаты на сегодняшний день для клиента с id 1.

```
INSERT INTO Booking (room_id, guest_id, checkin_date, checkout_date, status) -- вставляем
в Booking
VALUES (
    (SELECT room_id
     FROM DailyRoomStatus drs
     WHERE drs.status = 'free'
        AND drs.status_date = current_date
     LIMIT 1), -- Выбираем номер комнаты, которая свободна сегодня
    1,
    current_date,
    current_date + INTERVAL '1 days',
    'booked'
);
```



	booking_id [PK] integer	room_id integer	guest_id integer	checkin_date date	checkout_date date	status character varying (20)	promotion_id integer
1	1	2	1	2025-09-10	2025-09-15	checked_in	1
2	2	3	2	2025-09-05	2025-09-10	checked_out	1
3	3	5	3	2025-09-12	2025-09-14	booked	[null]
4	4	2	1	2025-09-25	2025-09-27	checked_out	[null]
5	5	8	3	2025-09-28	2025-09-30	checked_out	[null]
6	6	5	2	2025-09-26	2025-09-29	checked_out	[null]

Рис. 11: До выполнения запроса



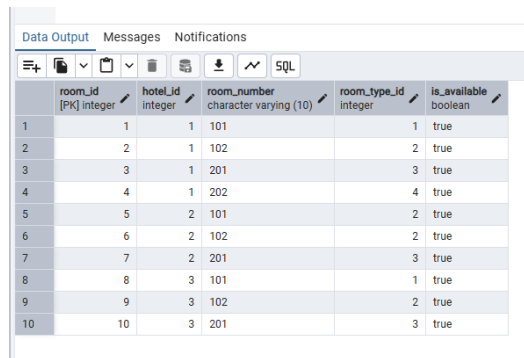
	booking_id [PK] integer	room_id integer	guest_id integer	checkin_date date	checkout_date date	status character varying (20)	promotion_id integer
1	1	2	1	2025-09-10	2025-09-15	checked_in	1
2	2	3	2	2025-09-05	2025-09-10	checked_out	1
3	3	5	3	2025-09-12	2025-09-14	booked	[null]
4	4	2	1	2025-09-25	2025-09-27	checked_out	[null]
5	5	8	3	2025-09-28	2025-09-30	checked_out	[null]
6	6	5	2	2025-09-26	2025-09-29	checked_out	[null]
7	7	1	1	2025-09-30	2025-10-01	booked	[null]

Рис. 12: После выполнения запроса

3.2 Запрос UPDATE

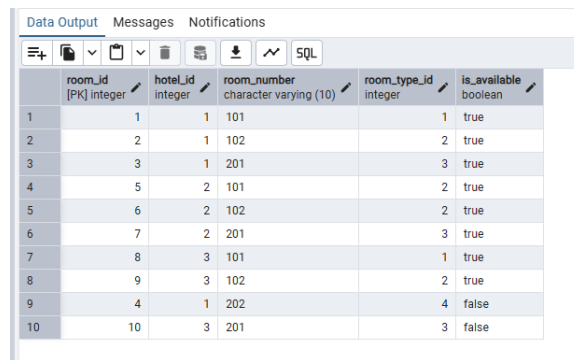
Установим `is_available = false` для всех комнат, которые сегодня заняты.

```
UPDATE Room
SET is_available = false
WHERE room_id IN (
    SELECT room_id
    FROM DailyRoomStatus
    WHERE status != 'free' -- Статус комнаты не равен 'free'
    AND status_date = current_date -- Текущая дата
);
```



	room_id [PK] integer	hotel_id integer	room_number character varying (10)	room_type_id integer	is_available boolean
1	1	1	101	1	true
2	2	1	102	2	true
3	3	1	201	3	true
4	4	1	202	4	true
5	5	2	101	2	true
6	6	2	102	2	true
7	7	2	201	3	true
8	8	3	101	1	true
9	9	3	102	2	true
10	10	3	201	3	true

Рис. 13: До выполнения запроса



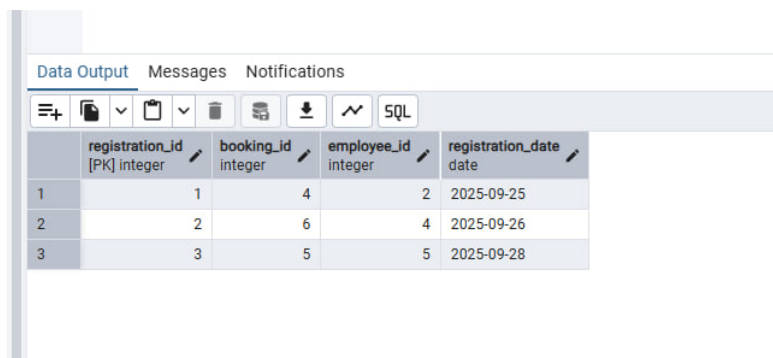
	room_id [PK] integer	hotel_id integer	room_number character varying (10)	room_type_id integer	is_available boolean
1	1	1	101	1	true
2	2	1	102	2	true
3	3	1	201	3	true
4	5	2	101	2	true
5	6	2	102	2	true
6	7	2	201	3	true
7	8	3	101	1	true
8	9	3	102	2	true
9	4	1	202	4	false
10	10	3	201	3	false

Рис. 14: После выполнения запроса

3.3 Запрос DELETE

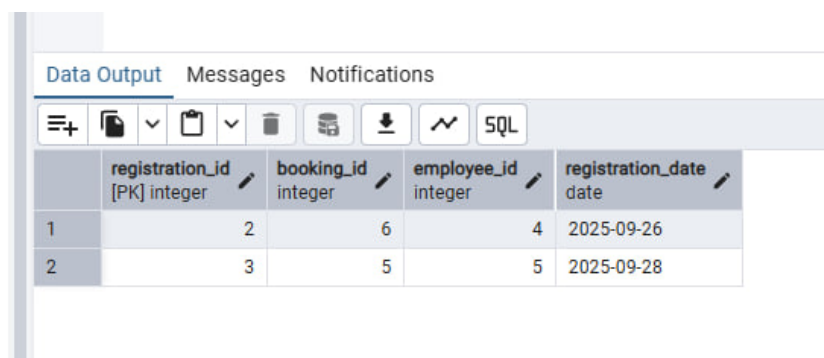
Удаление всех записей о заселениях, дата которых больше 2-ух дней.

```
DELETE FROM Registration
WHERE booking_id IN (
    SELECT booking_id
    FROM Booking
    WHERE checkout_date < current_date - INTERVAL '2 days'
);
```



	registration_id [PK] integer	booking_id integer	employee_id integer	registration_date date
1	1	4	2	2025-09-25
2	2	6	4	2025-09-26
3	3	5	5	2025-09-28

Рис. 15: До выполнения запроса



	registration_id [PK] integer	booking_id integer	employee_id integer	registration_date date
1	2	6	4	2025-09-26
2	3	5	5	2025-09-28

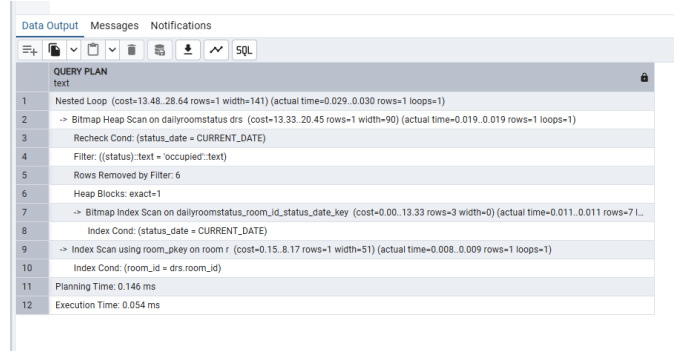
Рис. 16: После выполнения запроса

4 Создание индексов

Для примера был выбран запрос:

```
SELECT *
FROM DailyRoomStatus drs
JOIN Room r ON drs.room_id = r.room_id
WHERE drs.status = 'occupied' -- комната занята
      AND drs.status_date = current_date; -- занята на текущий день
```

План запроса без индексов: Теперь для запроса создадим индекс на drs.status и drs.status_date:



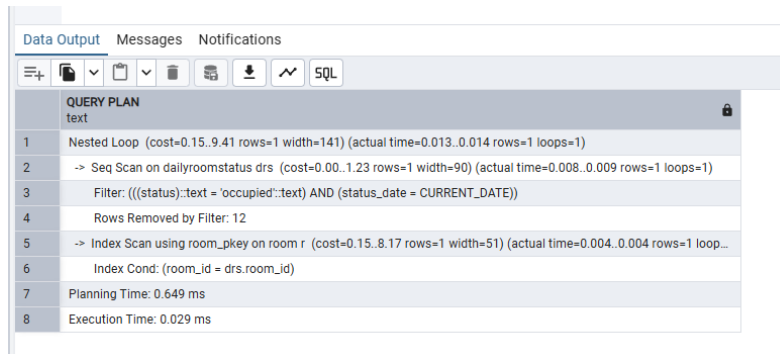
The screenshot shows a SQL query plan for a query without indexes. The plan is displayed in a window with tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a 'QUERY PLAN' table. The plan consists of 12 steps:

Step	Operation
1	Nested Loop (cost=13.48..28.64 rows=1 width=141) (actual time=0.029..0.030 rows=1 loops=1)
2	-> Bitmap Heap Scan on dailyroomstatus drs (cost=13.33..20.45 rows=1 width=90) (actual time=0.019..0.019 rows=1 loops=1)
3	Recheck Cond: (status_date = CURRENT_DATE)
4	Filter: ((status)::text = 'occupied'::text)
5	Rows Removed by Filter: 6
6	Heap Blocks: exact=1
7	-> Bitmap Index Scan on dailyroomstatus_room_id_status_date_key (cost=0.00..13.33 rows=3 width=0) (actual time=0.011..0.011 rows=7 loops=1)
8	Index Cond: (status_date = CURRENT_DATE)
9	-> Index Scan using room_pkey on room r (cost=0.15..8.17 rows=1 width=51) (actual time=0.008..0.009 rows=1 loops=1)
10	Index Cond: (room_id = drs.room_id)
11	Planning Time: 0.146 ms
12	Execution Time: 0.054 ms

Рис. 17: План запроса без индексов

```
CREATE INDEX idx_drs_status_date
ON DailyRoomStatus (status, status_date);
```

План запроса с индексами: Как видно, время выполнения уменьшилось благодаря индексам.



The screenshot shows a SQL query plan for the same query but with indexes. The plan is displayed in a window with tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a 'QUERY PLAN' table. The plan consists of 8 steps:

Step	Operation
1	Nested Loop (cost=0.15..9.41 rows=1 width=141) (actual time=0.013..0.014 rows=1 loops=1)
2	-> Seq Scan on dailyroomstatus drs (cost=0.00..1.23 rows=1 width=90) (actual time=0.008..0.009 rows=1 loops=1)
3	Filter: (((status)::text = 'occupied'::text) AND (status_date = CURRENT_DATE))
4	Rows Removed by Filter: 12
5	-> Index Scan using room_pkey on room r (cost=0.15..8.17 rows=1 width=51) (actual time=0.004..0.004 rows=1 loops=1)
6	Index Cond: (room_id = drs.room_id)
7	Planning Time: 0.649 ms
8	Execution Time: 0.029 ms

Рис. 18: План запроса с индексами

Удаление индексов:

```
DROP INDEX IF EXISTS idx_drs_status_date;
```


Выводы

В ходе лабораторной работы были получены знания по использованию запросов в QueryTool, создание views, их просмотр, создания запросов с подзапросами.

Также была проведена работа с использованием индексов для оптимизации запросов и использовании плана запроса.