Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №4 «Запросы на выборку и модификацию данных. Представления. Работа с индексами.»

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

Автор: Ковалев Г. П.

Факультет: ПИН

Группа: К3241

Преподаватель: Говорова М.М.

Санкт-Петербург 2025

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 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 1. БД «Отель»

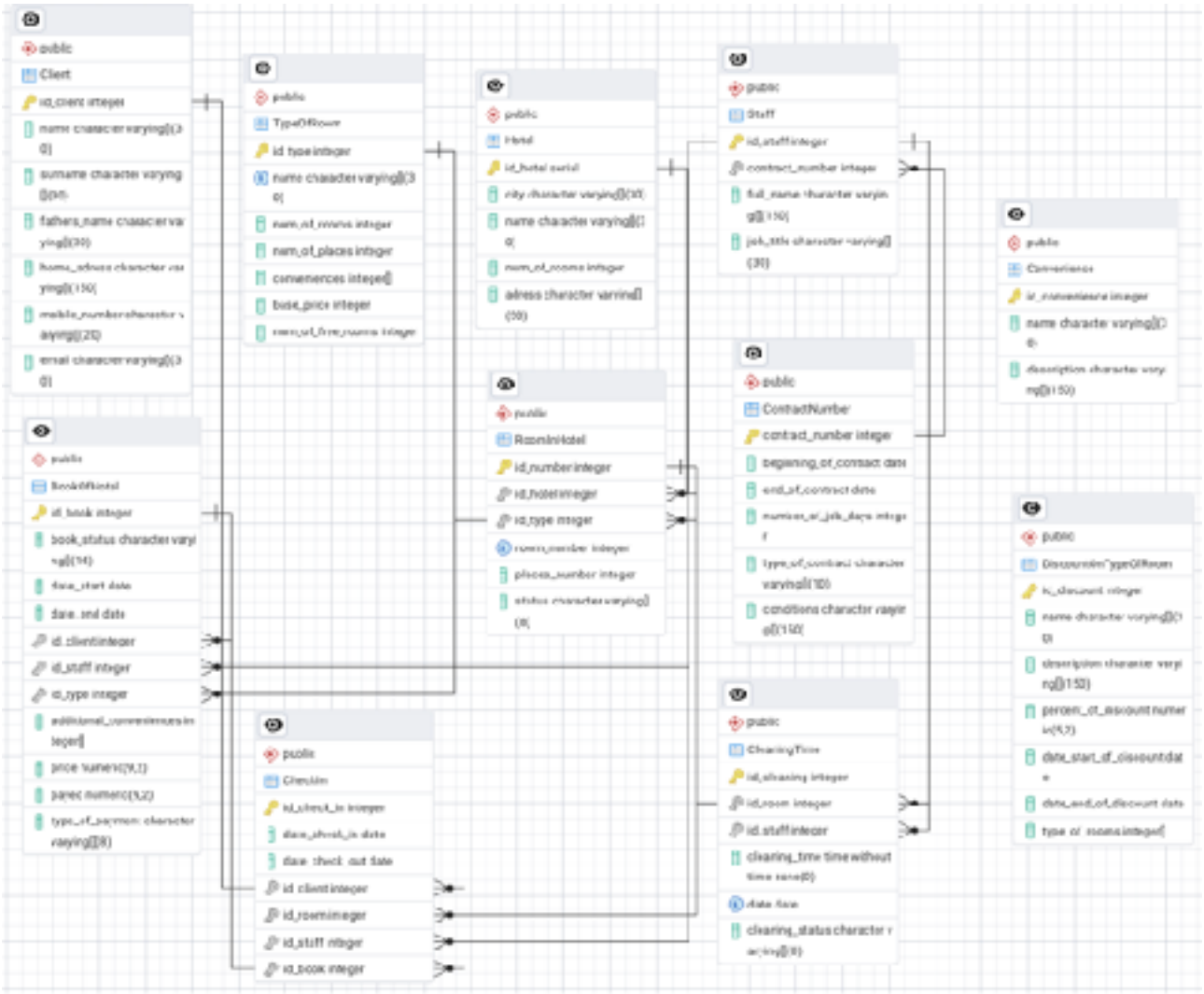
Необходимые запросы:

- Составить список всех 2-местных номеров отелей, с ценой менее 200 т.р., упорядочив данные в порядке уменьшения стоимости.
- Выбрать все записи регистрации постояльцев, которые выехали из отелей в течение двух последних недель.
- Чему равен общий суточный доход каждого отеля за последний месяц?
- Составить список свободных номеров заданного отеля на текущий день.
- Найти общие потери от незанятых номеров за текущий день по всей сети.
- Определить, в каком отеле имеется наибольшее количество незанятых номеров на текущие сутки.
- Определить самый популярный тип номеров за последний год.

Необходимые представления:

- Для турагентов (поиск свободных номеров в отелях).
- Для владельца компании (информация о доходах каждого отеля в сети за прошедший месяц).

Схема БД:



Часть 1: Запросы

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

	id_number [PK] integer	id_hotel integer	id_type integer	room_number integer	places_number integer	status character varying[] (8)
1	1	4	2	101	2	{Свободен}
2	2	4	1	102	1	{Занят}
3	3	5	3	201	3	{Свободен}
4	4	5	2	202	2	{Занят}
5	5	6	1	301	1	{Свободен}

Запрос:

```
SELECT room.id_number, room.places_number, room.room_number, room.id_hotel,
roomType.name,roomType.base_price, room.status
FROM public."RoomInHotel" room
JOIN public."TypeOfRoom" roomType ON room.id_type = roomType.id_type
WHERE roomType.num_of_places = 2 AND roomType.base_price < 200000
ORDER BY roomType.base_price DESC;
```

Результат:

	id_number integer	places_number integer	room_number integer	id_hotel integer	name character varying[] (30)	base_price integer	status character varying[] (8)
1	1	2	101	4	{Двухместный}	5000	{Свободен}
2	4	2	202	5	{Двухместный}	5000	{Занят}

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

	id_check_in [PK] integer	date_check_in date	date_check_out date	id_client integer	id_room integer	id_staff integer	id_book integer
1	1	2025-05-10	2025-05-15	1	1	1	1
2	2	2025-06-01	2025-06-05	2	2	5	2
3	3	2025-07-12	2025-07-20	3	3	5	3

Запрос:

```
SELECT * FROM public."CheckIn"
WHERE date_check_out >= CURRENT_DATE - INTERVAL '14 days' AND
date_check_out <= CURRENT_DATE;
```

	id_check_in [FK] integer	date_check_in date	date_check_out date	id_client integer	id_room integer	id_staff integer	id_book integer
1	1	2025-05-10	2025-05-15	1	1	1	1

3) Чему равен общий суточный доход каждого отеля за последний месяц?

	id_hotel [PK] integer	city character varying[] (30)	name character varying[] (30)	num_of_rooms integer	adress character varying[] (50)
1	4	{Санкт-Петербург}	{"Отель Невский"}	80	{"Невский проспект"}
2	5	{Казань}	{Татарстан}	150	{"ул Баумана"}
3	6	{Новосибирск}	{Сибирь}	95	{"Красный проспект"}
4	7	{Екатеринбург}	{Урал}	70	{"ул Ленина"}

Запрос:

```
SELECT hotel.id_hotel, hotel.name AS hotel_name, ROUND(SUM(book.price /
GREATEST((book.date_end - book.date_start), 1)), 2) AS daily_income
FROM public."Hotel" hotel
LEFT JOIN public."RoomInHotel" room ON hotel.id_hotel = room.id_hotel
LEFT JOIN public."CheckIn" checkIn ON room.id_number = checkIn.id_room AND
checkIn.date_check_in >= CURRENT_DATE - INTERVAL '1 month' AND checkIn.date_check_in
<= CURRENT_DATE
LEFT JOIN public."BookOfHotel" book ON checkIn.id_book = book.id_book
GROUP BY hotel.id_hotel, hotel.name
ORDER BY daily_income DESC;
```

	id_hotel [PK] integer	hotel_name character varying[] (30)	daily_income numeric
1	7	{Урал}	[null]
2	6	{Сибирь}	[null]
3	5	{Татарстан}	[null]
4	4	{"Отель Невский"}	3000.00

4) Составить список свободных номеров заданного отеля на текущий день.

Допустим мы взяли отель с id = 5.

	id_number [PK] integer	id_hotel integer	id_type integer	room_number integer	places_number integer	status character varying[] (8)
1	3	5	3	201	3	{Свободен}
2	4	5	2	202	2	{Занят}

Запрос:

```
SELECT * FROM public."RoomInHotel"
WHERE id_hotel = 5 AND status[1] = ‘Свободен';
```

	id_number [PK] integer	id_hotel integer	id_type integer	room_number integer	places_number integer	status character varying[] (8)
1	3	5	3	201	3	{Свободен}

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

Запрос:

```
SELECT SUM(typeRoom.base_price) FROM public."RoomInHotel" room
JOIN public."TypeOfRoom" typeRoom on room.id_type = typeRoom.id_type
WHERE room.status[1] = 'Свободен';
```

	sum bigint
1	17000

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

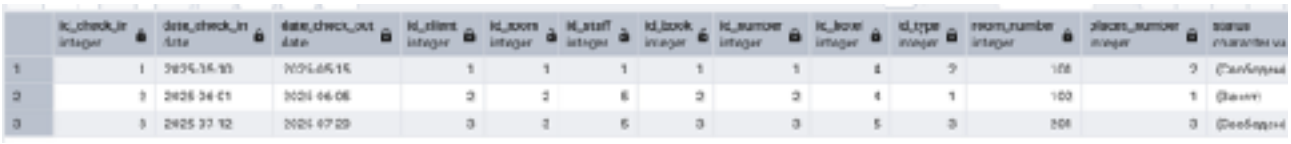
	id_number [PK] integer	id_hotel integer	id_type integer	room_number integer	places_number integer	status character varying[] (8)
1	1	4	2	101	2	{Свободен}
2	2	4	1	102	1	{Занят}
3	3	5	3	201	3	{Свободен}
4	4	5	2	202	2	{Занят}
5	5	6	1	301	1	{Свободен}
6	6	4	2	104	2	{Свободен}

Запрос:

```
SELECT room.id_hotel, COUNT(room.id_number) AS free_room_count
FROM public."RoomInHotel" room
WHERE room.status[1] = 'Свободен'
GROUP BY room.id_hotel
HAVING COUNT(room.id_number) = (
        SELECT MAX(sub.count_per_hotel)
        FROM (
                SELECT COUNT(id_number) AS count_per_hotel
                FROM public."RoomInHotel"
                WHERE status[1] = 'Свободен'
                GROUP BY id_hotel
        ) sub
);
```

	id_hotel integer	free_room_count bigint
1	4	2

7) Определить самый популярный тип номеров за последний год.



Запрос:

```
	SELECT room.id_type, COUNT(room.id_number) as count_per_type FROM public."CheckIn" checkIN
	JOIN public."RoomInHotel" room on checkIN.id_room = room.id_number
	GROUP BY room.id_type
	HAVING COUNT(room.id_number) = (
		SELECT MAX(sub.count_per_type)
		FROM (
			SELECT COUNT(room.id_number) as count_per_type FROM public."CheckIn" checkIN
			JOIN public."RoomInHotel" room on checkIN.id_room = room.id_number
			GROUP BY room.id_type
		) sub
	);
```

	id_type integer	count_per_type bigint
1	1	1
2	2	1
3	3	1

Часть 2: Представления

1) Для турагентов (поиск свободных номеров в отелях).

```
CREATE OR REPLACE VIEW public."AvailableRoomsForAgents" AS
SELECT hotel.name[1] AS hotel_name, hotel.city[1] AS city, hotel.adress[1] as adress,
room.room_number, room.places_number, typeRoom.name[1] AS room_type,
typeRoom.base_price
FROM public."RoomInHotel" room
JOIN public."Hotel" hotel ON room.id_hotel = hotel.id_hotel
JOIN public."TypeOfRoom" typeRoom ON room.id_type = typeRoom.id_type
WHERE room.status[1] = ‘Свободен';
```

```
CREATE VIEW

Query returned successfully in 50 msec.
```

	hotel_name character varying (30)	city character varying (30)	adress character varying (50)	room_number integer	places_number integer	room_type character varying (30)	base_price integer
1	Отель Невский	Санкт-Петербург	Невский проспект	101	2	Двухместный	5000
2	Татарстан	Казань	ул Баумака	201	3	Люкс	9000
3	Сибирь	Новосибирск	Красный проспект	301	1	Одноместный	3000
4	Отель Невский	Санкт-Петербург	Невский проспект	104	2	Двухместный	5000

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

Запрос:

```
CREATE OR REPLACE VIEW public."MonthlyIncomePerHotel" AS
SELECT hotel.id_hotel, hotel.name[1] AS hotel_name,
ROUND(COALESCE(SUM(book.price / GREATEST((checkIn.date_check_out -
checkIn.date_check_in), 1)), 0), 2) AS monthly_income
FROM public."Hotel" hotel
LEFT JOIN public."RoomInHotel" room ON hotel.id_hotel = room.id_hotel
LEFT JOIN public."CheckIn" checkIn ON room.id_number = checkIn.id_room AND
checkIn.date_check_in >= CURRENT_DATE - INTERVAL '1 month' AND checkIn.date_check_in
<= CURRENT_DATE
LEFT JOIN public."BookOfHotel" book ON checkIn.id_book = book.id_book
GROUP BY hotel.id_hotel, hotel.name[1];
```

```
CREATE VIEW

Query returned successfully in 43 msec.
```

	id_hotel integer	hotel_name character varying (30)	monthly_income numeric
1	4	Отель Невский	3000.00
2	7	Урал	0.00
3	5	Татарстан	0.00
4	6	Сибирь	0.00

Часть 3: Запросы на модификацию данных

1) Insert

Добавить одноместный номер в тот отель, где минимальное количество номеров среди всех отелей

Запрос:

```
INSERT INTO public."RoomInHotel" (id_number, id_hotel, id_type, room_number, places_number, status)
SELECT
    (SELECT MAX(id_number) + 1 FROM public."RoomInHotel") AS id_number,
    sub.id_hotel,
    1 AS id_type,
    COALESCE(
        (SELECT MAX(room.room_number) + 1
        FROM public."RoomInHotel" room
        WHERE room.id_hotel = sub.id_hotel),
        1
    ) AS room_number,
    1 AS places_number,
    ARRAY['Свободен'] AS status
FROM (
    SELECT hotel.id_hotel
    FROM public."Hotel" as hotel
    LEFT JOIN public."RoomInHotel" as room on hotel.id_hotel = room.id_hotel
    GROUP BY hotel.id_hotel
    ORDER BY COUNT(*) ASC
    LIMIT 1
) AS sub;
```

До:

```
1    4    2    101    2    "{Свободен}"
2    4    1    102    1    "{Занят}"
3    5    3    201    3    "{Свободен}"
4    5    2    202    2    "{Занят}"
5    6    1    301    1    "{Свободен}"
```

6	4	2	104	2	"{Свободен}"
7	6	1	302	1	"{Свободен}"
8	6	1	303	1	"{Свободен}"
9	5	1	203	1	"{Свободен}"
10	6	1	304	1	"{Свободен}"
11	5	1	204	1	"{Свободен}"
12	4	1	105	1	"{Свободен}"
13	6	1	305	1	"{Свободен}"
14	5	1	205	1	"{Свободен}"
15	7	1	1	1	"{Свободен}"
16	7	1	2	1	"{Свободен}"
17	7	1	3	1	"{Свободен}"
18	7	1	4	1	«{Свободен}"

После двух запросов:

"id_number"	"id_hotel"	"id_type"	"room_number"	"places_number"	"status"
1	4	2	101	2	"{Свободен}"
2	4	1	102	1	"{Занят}"
3	5	3	201	3	"{Свободен}"
4	5	2	202	2	"{Занят}"
5	6	1	301	1	"{Свободен}"
6	4	2	104	2	"{Свободен}"
7	6	1	302	1	"{Свободен}"
8	6	1	303	1	"{Свободен}"
9	5	1	203	1	"{Свободен}"
10	6	1	304	1	"{Свободен}"
11	5	1	204	1	"{Свободен}"
12	4	1	105	1	"{Свободен}"
13	6	1	305	1	"{Свободен}"
14	5	1	205	1	"{Свободен}"
15	7	1	1	1	"{Свободен}"
16	7	1	2	1	"{Свободен}"
17	7	1	3	1	"{Свободен}"
18	7	1	4	1	"{Свободен}"
19	4	1	106	1	"{Свободен}"
20	7	1	5	1	«{Свободен}"

2) Update

Обновить количество номеров в таблице Hotel до актуального

Запрос:

```
UPDATE public."Hotel" hotel
SET num_of_rooms = sub.actual_count
FROM (
	SELECT id_hotel, COUNT(*) AS actual_count
```

```
        FROM public."RoomInHotel"
        GROUP BY id_hotel
    ) AS sub
    WHERE hotel.id_hotel = sub.id_hotel;
```

До:

```
"id_hotel"      "city"  "name""num_of_rooms"      "adress"
4       "{Санкт-Петербург}"         "{""Отель Невский""}"       80      "{""Невский
проспект""}"
5       "{Казань}"    "{Татарстан}"         150     "{""ул Баумана""}"
6       "{Новосибирск}"     "{Сибирь}"    95      "{""Красный проспект""}"
7       "{Екатеринбург}"    "{Урал}"      70      "{""ул Ленина""}"
```

После:

```
4       "{Санкт-Петербург}"         "{""Отель Невский""}"       5       "{""Невский
проспект""}"
5       "{Казань}"    "{Татарстан}"         5       "{""ул Баумана""}"
6       "{Новосибирск}"     "{Сибирь}"    5       "{""Красный проспект""}"
7       "{Екатеринбург}"    "{Урал}"      5       "{""ул Ленина»"}"
```

3) Delete

Удалим тех клиентов, которых нет в CheckIn или BookOfHoteel

Запрос:

```
    DELETE FROM public."Client"
    WHERE id_client NOT IN (
        SELECT DISTINCT id_client FROM public."CheckIn"
    )
    AND id_client NOT IN (
        SELECT DISTINCT id_client FROM public."BookOfHotel"
    );
```

До:

```
1       "{Иван}"      "{Петров}"    "{Сергеевич}"         "{""г Москва ул Арбат д
10""}" "{+79161234567}"    "{ivan.petrov@mail.ru}"
2       "{Мария}"     "{Иванова}"  "{Алексеевна}"       "{""г Санкт-Петербург пр Невский д
25""}" "{+79261234567}"    "{m.ivanova@example.com}"
3       "{Александр}"       "{Смирнов}" "{Игоревич}""{""г Казань ул Баумана д
5""}"   "{+79371234567}"    "{smirnov.alex@mail.ru}"
4       "{Ольга}"     "{Кузнецова}"       "{Владимировна}"   "{""г Екатеринбург ул Ленина
д 8""}""{+79501234567}"    "{olga_kuz@example.com}"
5       "{Дмитрий}" "{Соколов}"  "{Михайлович}"      "{""г Новосибирск ул Красный
проспект д 15""}"     "{+79051234567}"    "{d.sokolov@domain.ru}"
```

6 "{Наталья}" "{Козлова}" "{Викторовна}" "{""г Самара ул Победы д 20""}" "{+79170000000}" «{n.kozlova@example.com}"

После:

1 "{Иван}" "{Петров}" "{Сергеевич}" "{""г Москва ул Арбат д 10""}" "{+79161234567}" "{ivan.petrov@mail.ru}"
2 "{Мария}" "{Иванова}" "{Алексеевна}" "{""г Санкт-Петербург пр Невский д 25""}" "{+79261234567}" "{m.ivanova@example.com}"
3 "{Александр}" "{Смирнов}" "{Игоревич}" "{""г Казань ул Баумана д 5""}" "{+79371234567}" "{smirnov.alex@mail.ru}"
4 "{Ольга}" "{Кузнецова}" "{Владимировна}" "{""г Екатеринбург ул Ленина д 8""}""{+79501234567}" "{olga_kuz@example.com}"
5 "{Дмитрий}" "{Соколов}" "{Михайлович}" "{""г Новосибирск ул Красный проспект д 15""}" "{+79051234567}" «{d.sokolov@domain.ru}"

4) Индексация

Я буду делать индексацию в таблице «RoomInHotel» по id_hotel, чтобы ускорить поиск номеров в конкретном отеле. Посмотрим на скорость без индекса.

```
EXPLAIN ANALYZE
SELECT * FROM public."RoomInHotel"
WHERE id_hotel = 7;
```

"Seq Scan on ""RoomInHotel"" (cost=0.00..1.25 rows=1 width=52) (actual time=0.029..0.032 rows=5 loops=1)"
" Filter: (id_hotel = 7)"
" Rows Removed by Filter: 15"
"Planning Time: 0.609 ms"
"Execution Time: 0.065 ms»

```
CREATE INDEX idx_id_hotel
ON public."RoomInHotel" ((id_hotel));
```

"Index Scan using idx_id_hotel on ""RoomInHotel"" (cost=0.14..8.15 rows=1 width=52) (actual time=0.096..0.099 rows=5 loops=1)"
" Index Cond: (id_hotel = 7)"
"Planning Time: 0.108 ms"
"Execution Time: 0.114 ms»

Прироста в скорости не произошло, потому что в моей таблице всего лишь 20 строк в такой таблице будет лучше работать именно Seq Scan, поэтому в конце добавим удаление индекса.

```
DROP INDEX IF EXISTS idx_id_hotel;
```

5) Выводы

Работа показала важность оптимального проектирования запросов, индексов и структур базы данных. Использование подзапросов, представлений и индексов позволяет существенно повысить производительность и читаемость работы с реальными данными.