

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

**ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ № 2**

по теме:

*«Запросы на выборку и модификацию данных, представления и
индексы в PostgreSQL»*

по дисциплине: Проектирование и реализация баз данных

Специальность:

09.03.03 Мобильная и сетевая разработка

Проверила: Говорова

М.М. Дата: «...» ...

2023 г.

Оценка _____

Выполнил:

студент группы

K32392

Золотухин А.Н.

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.

Выполнение работы:

Предметная область – Отель (вариант 1)

Наименование БД – hotel_db

Схемы логической модели базы данных

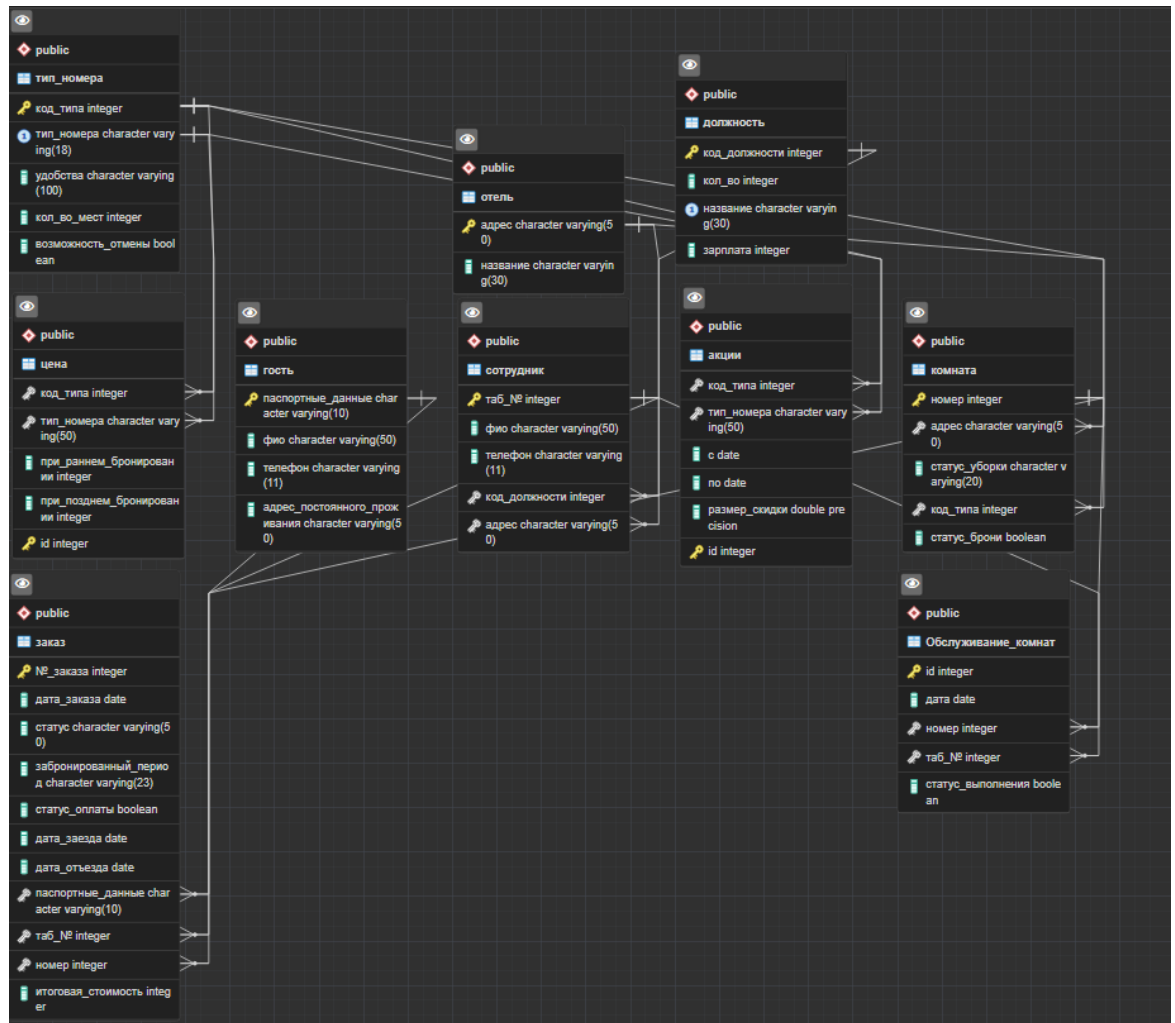


Рисунок 1 – ERD базы данных 1

Запросы на выборку.

- 1) Составить список всех 2-местных номеров отелей, с ценой менее 200 тысяч, упорядочить в порядке уменьшения стоимости.

Запрос:

```
SELECT k.номер, t.тип_номера, c.при_раннем_бронировании
FROM комната AS k
JOIN тип_номера AS t ON k.код_типа = t.код_типа
JOIN цена AS c ON t.код_типа = c.код_типа
WHERE t.кол_во_мест = 2 AND c.при_раннем_бронировании < 200000
ORDER BY c.при_раннем_бронировании DESC;
```

Вывод:

	номер integer	тип_номера character varying (18)	при_раннем_бронировании integer
1	50	Лиза	8000
2	51	Лиза	8000
3	52	Лиза	8000
4	53	Лиза	8000
5	101	Стандартный	6000
6	1	Стандарт	2000

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

Запрос:

```
SELECT *
FROM заказ
WHERE дата_отъезда BETWEEN CURRENT_DATE - INTERVAL '14 days'
AND CURRENT_DATE;
```

Вывод:

	№_заказа [PK] integer	дата_заказа date	статус character varying (50)	забронированный_период character varying (23)	статус_оплаты boolean	дата_заезда date	дата_отъезда date	паспортные_данные character varying (10)	таб_№ integer	номер integer	итоговая_стоимость integer
1	2	2023-04-01	забронирован	2023-04-10 - 2023-04-26	true	2023-04-10	2023-04-26	111111117	303	229	8000
2	3	2023-04-02	забронирован	2023-04-11 - 2023-04-27	true	2023-04-11	2023-04-27	111111118	303	232	20000
3	4	2023-04-02	забронирован	2023-04-11 - 2023-04-27	true	2023-04-11	2023-04-27	111111119	303	232	20000


2	"2023-04-01"	"забронирован"	"2023-04-10 - 2023-04-26"	true
	"2023-04-10"	"2023-04-26"	"111111117"	303 229 8000
3	"2023-04-02"	"забронирован"	"2023-04-11 - 2023-04-27"	true
	"2023-04-11"	"2023-04-27"	"111111118"	303 232 20000
4	"2023-04-02"	"забронирован"	"2023-04-11 - 2023-04-27"	true
	"2023-04-11"	"2023-04-27"	"111111119"	303 232 20000

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

Запрос:

```
SELECT SUM(z.итоговая_стоимость) AS общий_доход
FROM заказ AS z
JOIN комната AS k ON z.номер = k.номер
JOIN отель AS o ON k.адрес = o.адрес
WHERE z.дата_заезда BETWEEN CURRENT_DATE - INTERVAL '1 month'
AND CURRENT_DATE
GROUP BY o.адрес;
```

Вывод:


	общий_доход bigint 
1	48000

4) Составить список свободных номеров одного из отелей на текущий день.

Запрос:

```
SELECT k.номер
FROM комната AS k
JOIN отель AS o ON k.адрес = o.адрес
LEFT JOIN заказ AS z ON k.номер = z.номер
WHERE o.адрес = 'ул. Лиза, д.2' AND
(z.дата_заезда > CURRENT_DATE OR z.дата_отъезда < CURRENT_DATE
OR z.номер IS NULL);
```

Вывод:

	номер [PK] integer 
1	51
2	55
3	52
4	50
5	54
6	53

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

Запрос:

```
SELECT SUM(с.при_раннем_бронировании) AS потери
FROM комната AS k
JOIN тип_номера AS t ON k.код_типа = t.код_типа
JOIN цена AS с ON t.код_типа = с.код_типа
LEFT JOIN заказ AS z ON k.номер = z.номер
WHERE (z.дата_заезда > CURRENT_DATE OR z.дата_отъезда <
CURRENT_DATE OR z.номер IS NULL);
```

Вывод:

	потери bigint
1	244000

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

Запрос:

```
SELECT о.адрес, COUNT(k.номер) AS количество_незанятых_номеров
FROM комната AS k
JOIN отель AS о ON k.адрес = о.адрес
WHERE k.статус_брони = false
GROUP BY о.адрес
ORDER BY количество_незанятых_номеров DESC
LIMIT 1;
```

Вывод:

	адрес [PK] character varying (50)	количество_незанятых_номеров bigint
1	ул. Артем, д.1	7

- 7) Определить самый популярный тип номеров за последний год.

Запрос:



```
SELECT t.тип_номера, COUNT(z.номер) AS количество_бронирований
```

```

FROM заказ AS z
JOIN комната AS k ON z.номер = k.номер
JOIN тип_номера AS t ON k.код_типа = t.код_типа
WHERE z.дата_заезда BETWEEN CURRENT_DATE - INTERVAL '1 year'
AND CURRENT_DATE
GROUP BY t.тип_номера
ORDER BY количество_бронирований DESC
LIMIT 1;

```

Вывод:

	тип_номера character varying (18) 	количество_бронирований bigint 
1	Ультра-Артем	2

Создание Представлений.

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

Запрос:

```

CREATE VIEW свободные_номера AS
SELECT о.название AS отель, k.номер, t.тип_номера,
с.при_раннем_бронировании AS стоимость
FROM комната AS k
JOIN отель AS о ON k.адрес = о.адрес
JOIN тип_номера AS t ON k.код_типа = t.код_типа
JOIN цена AS с ON t.код_типа = с.код_типа
LEFT JOIN заказ AS z ON k.номер = z.номер
WHERE (z.дата_заезда > CURRENT_DATE OR z.дата_отъезда <
CURRENT_DATE OR z.номер IS NULL)
ORDER BY о.название, k.номер;

```

Вывод:

```

SELECT *
FROM свободные_номера;

```

	отель character varying (30)	номер integer	тип_номера character varying (18)	стоимость integer
1	Отель "Ленин"	101	Стандартный	6000
2	Отель Артем	228	Артем	8000
3	Отель Артем	229	Артем	8000
4	Отель Артем	229	Артем	8000
5	Отель Артем	230	Артем	8000
6	Отель Артем	231	Артем	8000
7	Отель Артем	232	Ультра-Артем	20000
8	Отель Артем	232	Ультра-Артем	20000
9	Отель Артем	232	Ультра-Артем	20000
10	Отель Артем	233	Ультра-Артем	20000
11	Отель Артем	234	Ультра-Артем	20000
12	Отель Артем	235	Ультра-Артем	20000
13	Отель Гамма	1	Стандарт	2000
14	Отель Лиза	50	Лиза	8000
15	Отель Лиза	51	Лиза	8000
16	Отель Лиза	52	Лиза	8000
17	Отель Лиза	53	Лиза	8000
18	Отель Лиза	54	Ультра-Лиза	20000
19	Отель Лиза	55	Ультра-Лиза	20000
20	Отель Эпсилон	2	Люкс	4000

2. Для владельца компании (информация о доходах за прошедший месяц)

Запрос:

```
CREATE VIEW доходы_отелей_за_месяц AS
SELECT о.название AS отель, SUM(z.итоговая_стоимость) AS общий_доход
FROM заказ AS z
JOIN комната AS k ON z.номер = k.номер
JOIN отель AS о ON k.адрес = о.адрес
WHERE z.дата_заезда BETWEEN CURRENT_DATE - INTERVAL '1 month'
AND CURRENT_DATE
GROUP BY о.название
ORDER BY общий_доход DESC;
```

Вывод:

```
SELECT *
```


FROM доходы_отелей_за_месяц;

	отель character varying (30)	общий_доход bigint
1	Отель Артем	48000

Запросы на модификацию данных.

- 1) INSERT: добавить новую акцию со скидкой 15% на все двухместные номера с текущей даты на две недели вперед.

Запрос:

```
INSERT INTO акции (код_типа, тип_номера, "с", "по", размер_скидки)
SELECT код_типа, тип_номера, CURRENT_DATE, CURRENT_DATE +
INTERVAL '14 days', 0.15
FROM тип_номера
WHERE кол_во_мест = 2;
```

Вывод:

```
select * from "акции"
```

	код_типа integer	тип_номера character varying (50)	с date	по date	размер_скидки double precision	id [PK] integer
1	1	Стандартный	2023-04-01	2023-04-10	0.2	1
2	2	Стандарт	2023-06-01	2023-06-30	0.1	2
3	3	Люкс	2023-07-01	2023-07-31	0.15	3
4	1	Стандартный	2023-05-03	2023-05-17	0.15	4
5	2	Стандарт	2023-05-03	2023-05-17	0.15	5
6	7	Лиза	2023-05-03	2023-05-17	0.15	6

- 2) UPDATE: Увеличить зарплату на 10% для всех сотрудников, работающих на должности с количеством сотрудников больше 5.

Запрос:

```
UPDATE должность
SET зарплата = зарплата * 1.1
WHERE код_должности IN (
  SELECT код_должности
  FROM должность
```

```
WHERE кол_во > 5  
);
```

Вывод:

```
select * from "должность"
```

	код_должности [PK] integer	кол_во integer	название character varying (30)	зарплата integer
1	1	5	Уборщик	15000
2	2	10	Администратор	33000

зарплата Администраторов увеличилась на 10% т.е с 30 тысяч до 33 тысяч.

3) DELETE: Удалить записи о заказах, которые были сделаны более 5 лет назад и статус заказа "отменен".

Запрос:

```
DELETE FROM заказ
```

```
WHERE дата_заказа < CURRENT_DATE - INTERVAL '5 years' AND статус =  
'отменен';
```

Вывод:

```
select * from заказ where дата_заказа < CURRENT_DATE - INTERVAL '5 years'
```

№_заказа [PK] integer	дата_заказа date	статус character varying (50)	забронированный_период character varying (23)	статус_оплаты boolean	дата_заезда date	дата_отъезда date	паспортные_данные character varying (10)	таб_№ integer	номер integer	итоговая_стоимость integer

Сравнение запросов с использованием индексирования.

Для демонстрации создания простого и составного индексов и сравнения времени выполнения запросов я выберу два запроса:

- 1) Запрос на получение всех заказов с определенными паспортными данными.
- 2) Запрос на получение всех обслуживаний комнат с указанной датой и статусом выполнения.

Запросы без индексов:

```
EXPLAIN ANALYZE SELECT * FROM заказ WHERE паспортные_данные =  
'1111111112';
```

EXPLAIN ANALYZE SELECT * FROM "Обслуживание_комнат" WHERE дата = '2023-05-01' AND статус_выполнения = true;
Вывод к запросам без индексов:

	QUERY PLAN
	text
1	Seq Scan on "заказ" (cost=0.00..13.63 rows=1 width=24...
2	Filter: (("паспортные_данные")::text = '111111112'::te...
3	Rows Removed by Filter: 10
4	Planning Time: 0.061 ms
5	Execution Time: 0.352 ms

	QUERY PLAN
	text
1	Seq Scan on "Обслуживание_комнат" (cost=0.00..32.63 rows=5 width=17) (actual time=0.033..0....
2	Filter: ("статус_выполнения" AND ("дата" = '2023-05-01'::date))
3	Rows Removed by Filter: 8
4	Planning Time: 2.587 ms
5	Execution Time: 0.044 ms

Создание индексов:

Создание простого индекса для столбца "паспортные_данные" в таблице "заказ".

CREATE INDEX idx_заказ_паспортные_данные ON заказ
(паспортные_данные);

Создание составного индекса для столбцов "дата" и "статус_выполнения" в таблице "Обслуживание_комнат".


CREATE INDEX idx_обслуживание_комнат_дата_статус ON
"Обслуживание_комнат" (дата, статус_выполнения);


Запросы с использованием индексов:

EXPLAIN ANALYZE SELECT * FROM заказ WHERE паспортные_данные = '111111112';

EXPLAIN ANALYZE SELECT * FROM "Обслуживание_комнат" WHERE дата = '2023-05-01' AND статус_выполнения = true;

Выводы к запросам с использованием индексов:

	QUERY PLAN	
	text	
1	Seq Scan on "заказ" (cost=0.00..1.14 rows=1 width=249) (actual time=0.009..0.010 rows=1 loops=1)	
2	Filter: (("паспортные_данные")::text = '111111112'::text)	
3	Rows Removed by Filter: 10	
4	Planning Time: 0.418 ms	
5	Execution Time: 0.021 ms	

	QUERY PLAN	
	text	
1	Seq Scan on "Обслуживание_комнат" (cost=0.00..1.20 rows=1 width=...	
2	Filter: ("статус_выполнения" AND ("дата" = '2023-05-01'::date))	
3	Rows Removed by Filter: 8	
4	Planning Time: 0.135 ms	
5	Execution Time: 0.054 ms	

Если индексы используются правильно, мы должны увидеть разницу в плане выполнения запроса, где индексы могут существенно улучшить производительность запросов.

Обратим внимание, что время выполнения и план запроса могут зависеть от системы и объема данных в таблицах. Также нужно учесть, что создание и поддержка индексов может замедлить операции записи в базе данных, поэтому следует использовать их с умом.

Выводы:

В ходе выполнения данной лабораторной работе были реализованы запросы на выборку данных и представления к базе данных на PostgreSQL согласно индивидуальному заданию. Более того были смоделированы различные штатные ситуации и имплементированы 3 запроса на модификацию данных. Также был проведен анализ графического представления всех запросов. Созданы простые и составные индексы для различных запросов и проанализировано их время выполнения с использованием индексов.