

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

Факультет инфокоммуникационных технологий

## **ЛАБОРАТОРНАЯ РАБОТА №2**

**Запросы на выборку и модификацию данных,  
представления и индексы в PostgreSQL  
по дисциплине:  
«Проектирование и реализация баз данных»**

**Выполнил студент:**

Тюмин Никита Сергеевич

Группа №K32402

**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.

## Программное обеспечение:

СУБД PostgreSQL 14, 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.

## Ход работы:

Работа проводилась в созданной ранее базе данных, ER диаграмма представлена на Рисунке 1.

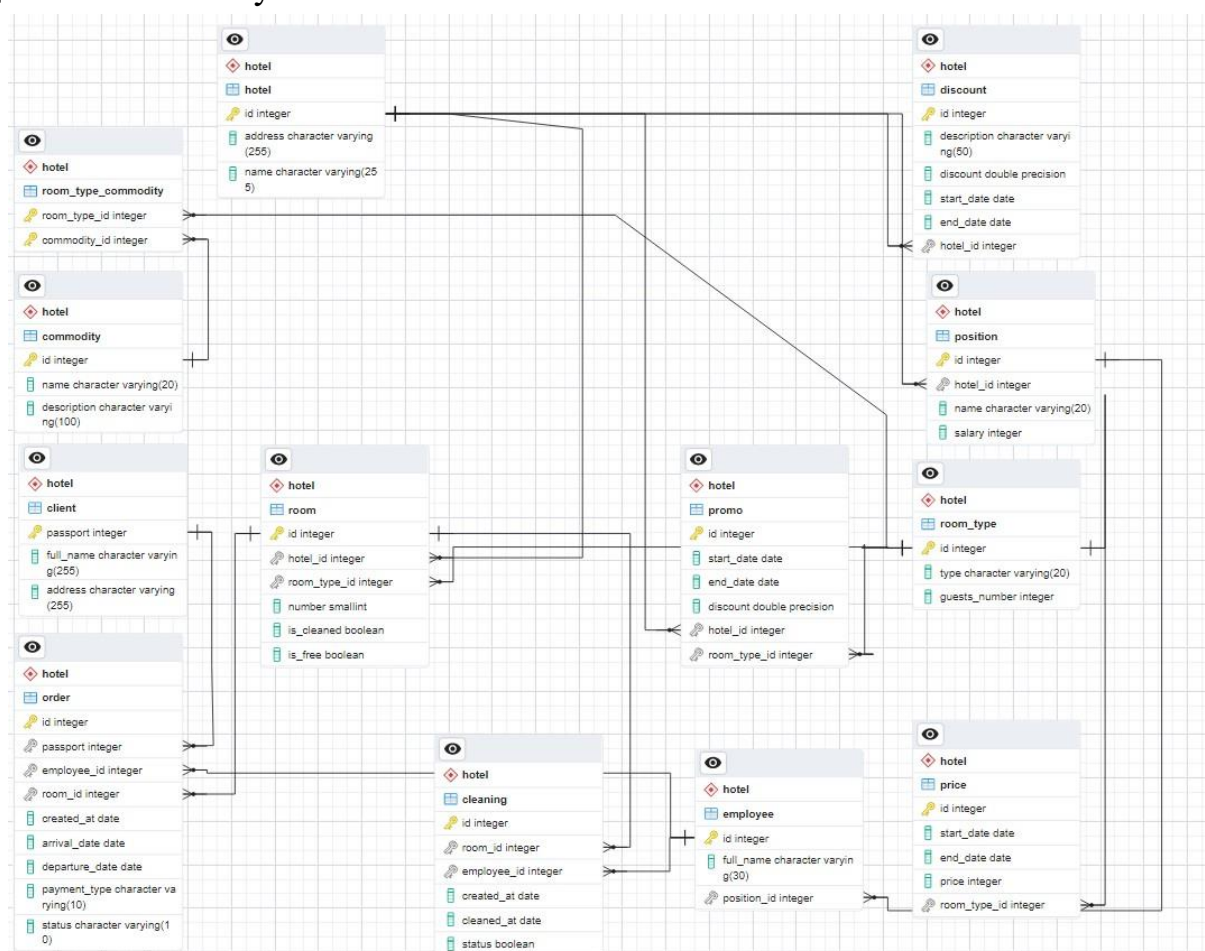


Рисунок 1 – ER диаграмма базы данных

## Запросы к базе данных:

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

```
with ids as (  
  select rt.id  
    from hotel.room_type rt  
   join hotel.price p  
  on rt.id = p.room_type_id  
 where p.end_date > CURRENT_DATE  
    and p.price < 200000  
    and rt.guests_number = 2  
  order by p.price desc  
)  
  
select *  
from hotel.room  
where room_type_id in (select * from ids)  
order by array_position(array(select * from ids), room_type_id)
```

Query Query History

```
1 with ids as (  
2   select rt.id  
3   from hotel.room_type rt  
4   join hotel.price p  
5   on rt.id = p.room_type_id  
6   where p.end_date > CURRENT_DATE  
7   and p.price < 200000  
8   and rt.guests_number = 2  
9   order by p.price desc  
10 )  
11  
12 select *  
13 from hotel.room  
14 where room_type_id in (select * from ids)  
15 order by array_position(array(select * from ids), room_type_id)  
16
```

Data Output Messages Notifications

	id [PK] integer	hotelId integer	room_type_id integer	number smallint	is_cleaned boolean	is_free boolean
1	181	6	4	410	true	true
2	141	5	4	510	true	true
3	172	6	4	401	true	true
4	173	6	4	402	true	true
5	174	6	4	403	true	true
6	175	6	4	404	true	true
7	176	6	4	405	true	true
8	177	6	4	406	true	true
9	178	6	4	407	true	true

Total rows: 40 of 40 Query complete 00:00:00.059 Ln 9, Col 18

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

```
select * from hotel.order  
where departure_date > CURRENT_DATE - interval '14 day'
```

Query

Query History

1

select \* from hotel.order

2

where departure\_date > CURRENT\_DATE - interval '14 day'

Data Output

Messages

Notifications

id

[PK] integer

passport

integer

employee\_id

integer

room\_id

integer

created\_at

date

arrival\_date

date

departure\_date

date

payment\_type

character varying (10)

status

character varying (10)

1

6

1167417325

1

11

2023-03-11

2023-03-11

2023-04-15

card

processed

2

7

1167417325

1

1

2023-03-11

2023-03-11

2023-04-15

card

processed

Total rows: 2 of 2

Query complete 00:00:00.320

Ln 2, Col 56

Query Query History

```

1 with tmp as(
2   select o.created_at, h.name, p.price, o.departure_date - o.arrival_date as days
3   from hotel.order o
4   join hotel.room r
5   on o.room_id = r.id
6   join hotel.hotel h
7   on h.id = r.hotel_id
8   join hotel.room_type rt
9   on rt.id = r.room_type_id
10  join hotel.price p
11  on rt.id = p.room_type_id
12  where p.end_date > CURRENT_DATE
13  and o.created_at > CURRENT_DATE - interval '1 month'
14 )
15
16 select tmp.name, sum(tmp.price * tmp.days) / 30
17 from tmp
18 group by tmp.name
19 -- select * from hotel.price

```

Data Output Messages Notifications

	name character varying (255)	?column? bigint
1	Grand Budapest	800
2	Hyatt	2666

Total rows: 2 of 2 Query complete 00:00:00.139 Ln 15, Col 1

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

```

select "number"
from hotel.room
where hotel_id = 2
and is_free is true

```

Query Query History

```

1 select "number"
2 from hotel.room
3 where hotel_id = 2
4 and is_free is true

```

Data Output Messages Notifications

	number smallint
10	110
11	201
12	202
13	203
14	204
15	205
16	206
17	207
18	208
19	209
20	210
21	301
22	302
23	303
24	304
25	305

Total rows: 30 of 30 Query complete 00:00:00.151 Ln 4, Col 20

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

```

with tmp as (
  select rt.id, count(*) as rooms, p.price
  from hotel.room r
  join hotel.room_type rt
  on r.room_type_id = rt.id
  join hotel.price p
  on rt.id = p.room_type_id
  where r.is_free is true
  and p.end_date > CURRENT_DATE

```

```

        group by rt.id, p.price
    )

select sum(tmp.rooms * tmp.price)
from tmp

```

The screenshot shows a SQL query editor with the following query:

```

1 with tmp as (
2     select rt.id, count(*) as rooms, p.price
3     from hotel.room r
4     join hotel.room_type rt
5     on r.room_type_id = rt.id
6     join hotel.price p
7     on rt.id = p.room_type_id
8     where r.is_free is true
9     and p.end_date > CURRENT_DATE
10    group by rt.id, p.price
11 )
12
13 select sum(tmp.rooms * tmp.price)
14 from tmp

```

The results viewer shows the following output:

	sum numeric
1	2330000

At the bottom, the status bar indicates: Total rows: 1 of 1 | Query complete 00:00:00.146 | Ln 13, Col 33

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

```

with tmp as (
    select h.name, count(*) as rooms
    from hotel.hotel h
    join hotel.room r
    on h.id = r.hotel_id
    where r.is_free is true
    group by h.name
)

select tmp.name, tmp.rooms
from tmp
where tmp.rooms = (
    select max(tmp.rooms)
    from tmp
)

```

Query Query History

```

1 with tmp as (
2   select h.name, count(*) as rooms
3   from hotel.hotel h
4   join hotel.room r
5   on h.id = r.hotel_id
6   where r.is_free is true
7   group by h.name
8 )
9
10 select tmp.name, tmp.rooms
11 from tmp
12 where tmp.rooms = (
13   select max(tmp.rooms)
14   from tmp
15 )

```

Data Output Messages Notifications

	name character varying (255)	rooms bigint
1	St Regis Hotels	50
2	Hyatt	50

Total rows: 2 of 2 Query complete 00:00:00.247

✓ Successfully run. Total query runtime: 247 msec. 2 rows affected. ✕

## 7. Определить самый популярный тип номеров за последний год.

```

select count(*) as most_popular
from hotel.order o
join hotel.room r
on o.room_id = r.id
where created_at > (
    select date(now() - interval '1 year')
)
group by r.room_type_id
order by count(*) desc
limit 1

```

Query Query History

```

1 select count(*) as most_popular
2 from hotel.order o
3 join hotel.room r
4 on o.room_id = r.id
5 where created_at > (
6   select date(now() - interval '1 year')
7 )
8 group by r.room_type_id
9 order by count(*) desc
10 limit 1

```

Data Output Messages Notifications

	most_popular bigint
1	3

Total rows: 1 of 1 Query complete 00:00:00.045

Ln 1, Col 32

## Представления:

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

```

create view TourAgent
as select h.name, rt.type, count(rt.type) as available

```

```

from hotel.hotel h
join hotel.room r
on h.id = r.hotel_id
join hotel.room_type rt
on rt.id = r.room_type_id
where r.is_free is true
group by h.name, rt.type
order by h.name;

```

```
select * from TourAgent
```

name	type	available
Grand Budapest	type1	10
Grand Budapest	type2	10
Grand Budapest	type3	10
Hyatt	type1	10
Hyatt	type2	10
Hyatt	type3	30
Marriott	type1	30
St Regis Hotels	type1	30
St Regis Hotels	type3	20

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

```

create view HotelRevenue
as with tmp as(
    select o.created_at, h.name, p.price, o.departure_date - o.arrival_date as
    days
    from hotel.order o
    join hotel.room r
    on o.room_id = r.id
    join hotel.hotel h
    on h.id = r.hotel_id
    join hotel.room_type rt
    on rt.id = r.room_type_id
    join hotel.price p
    on rt.id = p.room_type_id
    where p.end_date > CURRENT_DATE
    and o.created_at > CURRENT_DATE - interval '1 month'
)

```

```

select tmp.name, sum(tmp.price * tmp.days)
from tmp
group by tmp.name;

```

```
select * from HotelRevenue
```



Query Query History

```

1 create view HotelRevenue
2 as with tmp as(
3 select o.created_at, h.name, p.price, o.departure_date - o.arrival_date as days
4 from hotel.order o
5 join hotel.room r
6 on o.room_id = r.id
7 join hotel.hotel h
8 on h.id = r.hotel_id
9 join hotel.room_type rt
10 on rt.id = r.room_type_id
11 join hotel.price p
12 on rt.id = p.room_type_id
13 where p.end_date > CURRENT_DATE
14 and o.created_at > CURRENT_DATE - interval '1 month'
15 )
16
17 select tmp.name, sum(tmp.price * tmp.days)
18 from tmp
19 group by tmp.name;
20
21 select * from HotelRevenue

```

Data Output Messages Notifications

	name character varying (255)	sum bigint
1	Grand Budapest	24000
2	Hyatt	80000

Total rows: 2 of 2 Query complete 00:00:00.166 Ln 10, Col 27

## Запросы на модификацию данных:

1. Продление даты действия цены на месяц, если до конца действия осталось менее одной недели

```

update hotel.price
set end_date = end_date + interval '1 month'
where end_date < CURRENT_DATE + interval '1 week'
and end_date > CURRENT_DATE;

```

```
select * from hotel.price;
```

До выполнения запроса:

Query Query History

```

1 -- продление даты действия цены на месяц, если до конца действия осталось менее одной недели
2 select * from hotel.price;
3

```

Data Output Messages Notifications

	id [PK] integer	start_date date	end_date date	price integer	room_type_id integer
1	2	2023-01-01	2025-01-01	10000	2
2	3	2023-01-01	2030-01-01	15000	3
3	4	2023-01-01	2030-01-01	20000	4
4	1	2023-01-01	2025-01-01	6000	1
5	6	2020-01-01	2022-12-31	5000	1
6	5	2023-01-01	2023-04-20	25000	5

Total rows: 6 of 6 Query complete 00:00:00.226 Rows selected: 1 Ln 2, Col 27

После:

Query Query History

```

1  -- продление даты действия цены на месяц, если до конца действия осталось менее одной недели
2  update hotel.price
3  set end_date = end_date + interval '1 month'
4  where end_date < CURRENT_DATE + interval '1 week'
5  and end_date > CURRENT_DATE;
6
7  select * from hotel.price;
8

```

Data Output Messages Notifications

	id [PK] integer	start_date date	end_date date	price integer	room_type_id integer
1	2	2023-01-01	2025-01-01	10000	2
2	3	2023-01-01	2030-01-01	15000	3
3	4	2023-01-01	2030-01-01	20000	4
4	1	2023-01-01	2025-01-01	6000	1
5	6	2020-01-01	2022-12-31	5000	1
6	5	2023-01-01	2023-05-20	25000	5

Total rows: 6 of 6 Query complete 00:00:00.245 Rows selected: 1 Ln 5, Col 29

## 2. Повышение уборщика, сделавшего больше всех уборок

```

with tmp as (
  select e.id, count(c.id) as cleanings
  from hotel.employee e
  left join hotel.cleaning c
  on e.id = c.employee_id
  where position_id in (
    select id
    from hotel.position
    where name = 'Уборщик'
  )
  and c.status is true
  group by e.id
)

```

```

update hotel.employee
set position_id = (
  select id
  from hotel.position
  where name = 'Старший уборщик'
  and hotel_id = (
    select hotel_id
    from hotel.position p
    where p.id = position_id
  )
)
where id in (
  select tmp.id
  from tmp
  where tmp.cleanings = (
    select max(tmp.cleanings)
    from tmp
  )
);

```

```
select * from hotel.employee where id = 14;
```

## До выполнения запроса:

Query Query History

```
1 select * from hotel.employee where id = 14;
```

Data Output Messages Notifications

	id [PK] Integer	full_name character varying (30)	position_id Integer
1	14	Yurem Oconnell	6

Total rows: 1 of 1 Query complete 00:00:00.049 Ln 1, Col 44

## После:

Query Query History

```
1 -- повышение уборщика, сделавшего больше всех уборок
2 with tmp as (
3     select e.id, count(c.id) as cleanings
4     from hotel.employee e
5     left join hotel.cleaning c
6     on e.id = c.employee_id
7     where position_id in (
8         select id
9         from hotel.position
10        where name = 'Уборщик'
11    )
12    and c.status is true
13    group by e.id
14 )
15
16 update hotel.employee
17 set position_id = (
18     select id
19     from hotel.position
20     where name = 'Старший уборщик'
21     and hotel_id = (
22         select hotel_id
23         from hotel.position
24         where name = 'Уборщик'
25         order by cleanings desc
26         limit 1
27     )
28 )
```

Data Output Messages Notifications

	id [PK] Integer	full_name character varying (30)	position_id Integer
1	14	Yurem Oconnell	37

Total rows: 1 of 1 Query complete 00:00:00.125 Ln 36, Col 44

## Создание индексов:

Для более наглядного влияния индекса на запрос в таблицу client были добавлены данные:

Query

Query History

1

`select count(*) from hotel.client`

Data Output

Messages

Notifications

+

≡

↓

📄

🗑

📦

📥

📈

	count bigint
1	75000

Total rows: 1 of 1

Query complete 00:00:00.046

✓ Successfully run. Total query runtime: 46 msec. 1 rows affected

Вначале был создан простой индекс на таблице client. Запрос до создания индекса:

Query

Query History

1

2

3

4

5

`explain analyze select full_name from hotel.client where full_name not like '%Tracy%';`

Data Output

Messages

Notifications

+

≡

↓

📄

🗑

📦

📥

📈

	QUERY PLAN text
1	Seq Scan on client (cost=0.00..3228.50 rows=74993 width=14) (actual time=0.007..9.503 rows=74789 loops...
2	Filter: ((full_name)::text !~ '%Tracy%':text)
3	Rows Removed by Filter: 211
4	Planning Time: 0.066 ms
5	Execution Time: 10.933 ms

Total rows: 5 of 5

Query complete 00:00:00.115

Ln 4, Col 1

После создания:



Query Query History

1  
2  
3  
4  
5  
6

```
create index on hotel.client("full_name", passport);  
explain analyze select full_name from hotel.client where full_name like '%Tracy%' and passport > 1132016141;
```

Data Output Messages Notifications

+

📄

📄

🗑️

📄

📄

📄

📄

QUERY PLAN

text

🔒

1

Index Only Scan using client\_full\_name\_passport\_idx on client (cost=0.42..2821.83 rows=7 width=14) (actual time=2.469..8.709 rows=186 loop...

2

Index Cond: (passport > 1132016141)

3

Filter: ((full\_name)::text ~~ '%Tracy%':text)

4

Rows Removed by Filter: 66345

5

Heap Fetches: 0

6

Planning Time: 0.096 ms

7

Execution Time: 8.730 ms

Total rows: 7 of 7 Query complete 00:00:00.044

Ln 3, Col 8

Ситуация аналогична предыдущей, однако теперь мы выиграли пару миллисекунд в производительности: 11.4мс против 7.7мс.

## Выводы:

1. Были созданы запросы на выборку и модификацию данных с помощью подзапросов
2. Были созданы представления.
3. Были созданы индексы и проанализирована производительность запросов до и после создания индексов.