

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»**

**Выполнила:**  
студентка II курса ИКТ  
группы К3243  
Махнева Анастасия Денисовна

**Проверила:**  
Говорова Марина Михайловна

Санкт-Петербург  
2022

**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 1X, 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.

**Выполнение:**

Наименование БД – «Аэропорт». Схема логической модели БД, сгенерированная в Generate ERD:

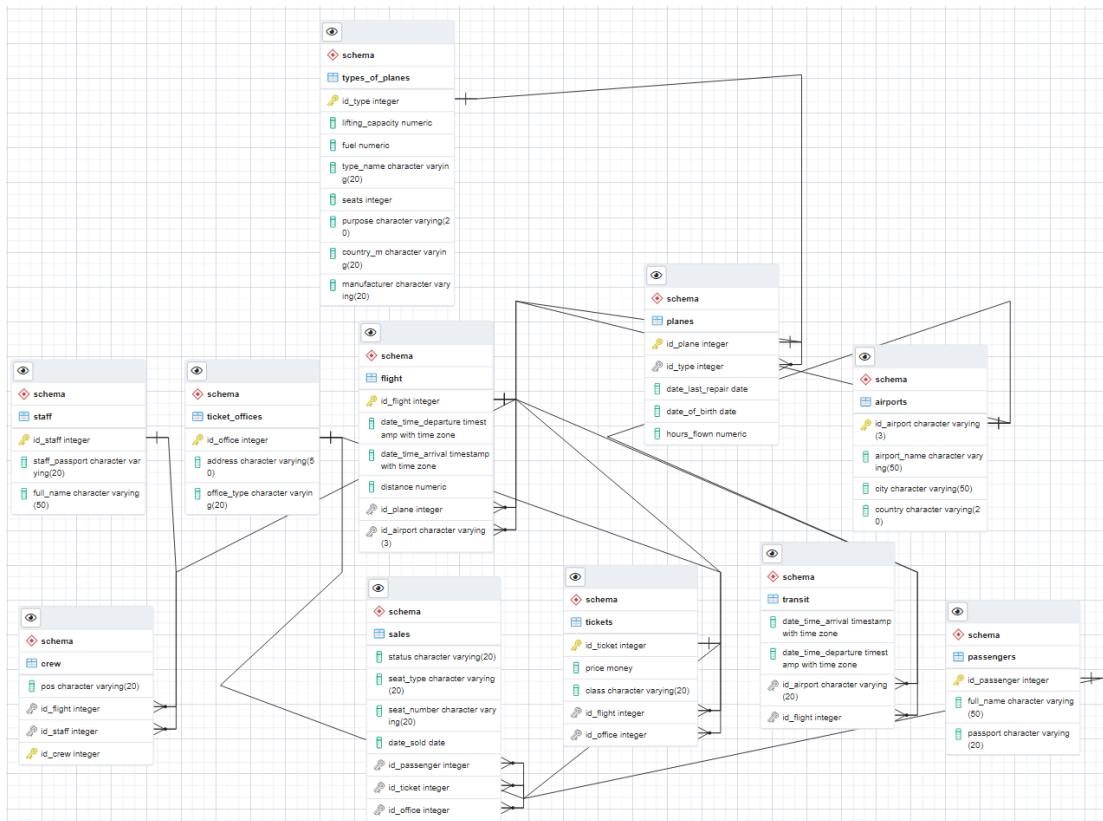


Рисунок 42 – Схема базы данных

## Запросы:

1. Определить расчетное время полета по всем маршрутам

```
1 select *, date_time_arrival-date_time_departure as time_flying from schema.flight
```

Рисунок 1 – Формулировка запроса 1

### Data Output

	id_flight [PK] integer	date_time_departure timestamp with time zone	date_time_arrival timestamp with time zone	distance numeric	id_plane integer	id_airport character varying (3)	time_flying interval
1	1	2022-06-10 13:00:00+03	2022-06-10 15:00:00+03	634	2	LED	02:00:00
2	3	2022-06-09 11:00:00+03	2022-06-09 12:50:00+03	1014	3	MMK	01:50:00
3	2	2022-06-09 12:00:00+03	2022-06-09 15:00:00+03	2878	5	TOF	03:00:00

Рисунок 2 – Результат выполнения запроса 1

2. Определить расход топлива по всем маршрутам.

```

1 select schema.flight.id_flight,
2 schema.types_of_planes.fuel, schema.flight.distance,
3 distance * fuel as fuel_used
4 from schema.flight
5 left join schema.planes on schema.flight.id_plane=schema.planes.id_plane
6 left join schema.types_of_planes on schema.planes.id_type=schema.types_of_planes.id_type

```

Рисунок 3 – Формулировка запроса 2

	id_flight integer	fuel numeric	distance numeric	fuel_used numeric
1	1	3	634	1902
2	3	5	1014	5070
3	2	1.3	2878	3741.4

Рисунок 4 – Результат выполнения запроса 2

3. Вывести данные о том, сколько свободных мест оставалось в самолетах, совершавших полет по заданному из рейсов за вчерашний день.

```

1 select flight.id_flight, types_of_planes.seats-count(tickets.id_ticket) as seats_left
2 from schema.flight
3 left join schema.planes on flight.id_plane=planes.id_plane
4 left join schema.types_of_planes on planes.id_type=types_of_planes.id_type
5 left join schema.tickets on flight.id_flight=tickets.id_flight
6 group by flight.id_flight, types_of_planes.seats

```

Рисунок 5 – Формулировка запроса 3.1 (рейсы за все время)

	id_flight [PK] integer	seats_left bigint
1	4	120
2	1	119
3	2	19
4	5	2
5	3	199

Рисунок 6 – Результат выполнения запроса 3.1

```

1 select flight.id_flight, types_of_planes.seats-count(tickets.id_ticket) as seats_left
2 from schema.flight
3 left join schema.planes on flight.id_plane=planes.id_plane
4 left join schema.types_of_planes on planes.id_type=types_of_planes.id_type
5 left join schema.tickets on flight.id_flight=tickets.id_flight
6 where flight.date_time_departure > current_date - interval '1 day'
7 group by flight.id_flight, types_of_planes.seats

```

Рисунок 7 – Формулировка запроса 3.2 (рейсы за вчерашний день)

	id_flight [PK] integer	seats_left bigint

Рисунок 8 – Результат выполнения запроса 3.2

4. Рассчитать убытки компании за счет непроданных билетов за вчерашний день.

(Рейсов вчера не было, как показал предыдущий запрос, поэтому будем искать убытки по рейсам за весь период.)

```

1 select flight.id_flight, (types_of_planes.seats-count(tickets.id_ticket))*tickets.price as money_lost
2 from schema.flight
3 left join schema.planes on flight.id_plane=planes.id_plane
4 left join schema.types_of_planes on planes.id_type=types_of_planes.id_type
5 left join schema.tickets on flight.id_flight=tickets.id_flight
6 group by flight.id_flight, types_of_planes.seats, tickets.price

```

Рисунок 9 – Формулировка запроса 4

	id_flight [PK] integer	money_lost money
1	1	1 428 000,00 ?
2	2	380 000,00 ?
3	3	3 383 000,00 ?

Рисунок 10 – Результат выполнения запроса 4

5. Определить, какой тип самолетов чаще всего летал в заданный аэропорт назначения.

(В разработанной БД не учитывается аэропорт назначения, а только аэропорт отправления, поэтому будем определять, какой тип самолетов чаще всего летал из заданного аэропорта отправления.)

```

1 select airports.id_airport, airports.city, types_of_planes.type_name, count(types_of_planes.id_type) as max_num_of_flights
2 from schema.airports
3 left join schema.flight on airports.id_airport=flight.id_airport
4 left join schema.planes on flight.id_plane=planes.id_plane
5 left join schema.types_of_planes on planes.id_type=types_of_planes.id_type
6 group by airports.id_airport, types_of_planes.type_name

```

Рисунок 11 – Формулировка запроса 5

	id_airport character varying (3)	city character varying (50)	type_name character varying (20)	max_num_of_flights bigint
1	DMB	Taraz	[null]	0
2	SVO	Moscow	first	1
3	KRO	Kurgan	[null]	0
4	PEE	Perm	[null]	0
5	LED	Saint Petersburg	first	1
6	SVO	Moscow	third	1
7	MSQ	Minsk	[null]	0
8	MMK	Murmansk	second	1
9	TOF	Tomsk	fourth	1

Рисунок 12 – Результат выполнения запроса 5

6. Вывести список самолетов, “возраст” которых превышает средний “возраст” самолетов этого типа.

```

1 select planes.id_plane, planes.id_type
2 from schema.planes
3 where (select avg(current_date-date_of_birth) from schema.planes) < current_date-planes.date_of_birth
4 order by planes.id_type

```

Рисунок 13 – Формулировка запроса 6

	id_plane [PK] integer	id_type integer
1	6	4
2	7	4

Рисунок 14 – Результат выполнения запроса 6

7. Определить тип самолетов, летающих во все аэропорты назначения.

(Аналогично запросу 5: в разработанной БД не учитывается аэропорт назначения, а только аэропорт отправления, поэтому будем определять тип самолетов, летающих из всех аэропортов отправления.)

```

1 select planes.id_type
2 from schema.airports, schema.planes
3 left join schema.flight on planes.id_plane=flight.id_plane
4 where (airports.id_airport) = all (select (id_airport)
5                                   from schema.flight, schema.planes
6                                   where flight.id_plane=planes.id_plane
7                                   group by planes.id_type, flight.id_airport)

```

Рисунок 15 – Формулировка запроса 7

	id_type	
	integer	

Рисунок 16 – Результат выполнения запроса 7

(По запросу таких типов не было найдено, потому что их действительно нет.)

### Представления:

1. Для пассажиров авиакомпании о рейсах в Москву на ближайшую неделю. (Т.к. БД не учитывает аэропорт назначения, будет рассматривать рейсы ИЗ Москвы на ближайшую неделю.)

```

1 create view schema.timetable_MSK as
2 select
3 flight.id_flight as "id_рейса",
4 flight.date_time_departure as "время вылета",
5 flight.id_airport as "код аэропорта",
6 airports.city as "город вылета"
7 from schema.flight, schema.airports
8 where flight.id_airport = airports.id_airport and airports.city='Moscow'
9 and flight.date_time_departure >= current_date
10 and flight.date_time_departure <= current_date + interval '1 week'
11 order by date_time_departure|

```

Рисунок 17 – Создание представления timetable\_MSK

	id_рейса integer	время вылета timestamp with time zone	код аэропорта character varying (3)	город вылета character varying (50)
1	6	2022-06-14 18:00:00+03	SVO	Moscow
2	7	2022-06-15 17:00:00+03	SVO	Moscow
3	8	2022-06-16 16:00:00+03	SVO	Moscow
4	9	2022-06-17 15:00:00+03	SVO	Moscow
5	10	2022-06-18 14:00:00+03	SVO	Moscow

Рисунок 18 – Готовое представление timetable\_MSK

2. Количество самолетов каждого типа, летавшего за последний месяц.

```

1 create view schema.types_last_month as
2 select
3 types_of_planes.id_type as "тип самолета",
4 count(planes.id_plane) as "количество самолетов"
5 from schema.types_of_planes, schema.planes, schema.flight
6 where planes.id_type = types_of_planes.id_type
7 and flight.id_plane = planes.id_plane
8 and flight.date_time_departure <= current_date
9 and flight.date_time_departure >= current_date - interval '1 month'
10 group by types_of_planes.id_type
11 order by types_of_planes.id_type|

```

Рисунок 19 – Создание представления types\_last\_month

	тип самолета integer	количество самолетов bigint
1	4	2
2	5	1
3	6	1
4	7	1

Рисунок 20 – Готовое представление types\_last\_month

**Запросы на модификацию данных с использованием подзапросов:**

1. Запрос с INSERT: добавить информацию о пилоте (должность «captain») Нестерове Владиславе на рейс с id\_flight «2».

```

1 insert into schema.crew (pos, id_flight, id_staff) values
2 ('captain', '2', (select id_staff from schema.staff where full_name='Nesterov Vladislav'))

```



Рисунок 21 – формулировка запроса с INSERT

	pos character varying (20)	id_flight integer	id_staff integer	id_crew [PK] integer
1	captain	1	6	1
2	first officer	1	2	2
3	service manager	1	7	3
4	flight attendant	1	4	4

Рисунок 22 – содержание таблицы crew до выполнения запроса

	pos character varying (20)	id_flight integer	id_staff integer	id_crew [PK] integer
1	captain	1	6	1
2	first officer	1	2	2
3	service manager	1	7	3
4	flight attendant	1	4	4
5	captain	2	1	5

Рисунок 23 – содержание таблицы crew после выполнения запроса  
(добавлена строка 5)

- Запрос с UPDATE: изменить данные о старшем бортпроводнике (должность «service manager») на рейсе «1». Заменить сотрудника с id\_staff «7» на сотрудника с номером паспорта «345».

```

1 update schema.crew
2 set id_staff=(select id_staff from schema.staff where staff_passport='345')
3 where id_flight='1' and pos='service manager'

```

Рисунок 24 – Формулировка запроса с UPDATE

	pos character varying (20)	id_flight integer	id_staff integer	id_crew [PK] integer
1	captain	1	6	1
2	first officer	1	2	2
3	service manager	1	7	3
4	flight attendant	1	4	4

Рисунок 25 – содержание таблицы crew до выполнения запроса

	pos character varying (20)	id_flight integer	id_staff integer	id_crew [PK] integer
1	captain	1	6	1
2	first officer	1	2	2
3	flight attendant	1	4	4
4	service manager	1	5	3

Рисунок 26 – содержание таблицы crew после выполнения запроса

3. Запрос с DELETE: удалить данные об экипаже рейса, отправляющегося в следующие дату и время: 2022-06-09 12:00:00.

```

1 delete from schema.crew
2 where id_flight=(select id_flight from schema.flight
3                  where date_time_departure='2022-06-09 12:00:00')
```

Рисунок 27 – Формулировка запроса с DELETE

	pos character varying (20)	id_flight integer	id_staff integer	id_crew [PK] integer
1	captain	1	6	1
2	first officer	1	2	2
3	service manager	1	7	3
4	flight attendant	1	4	4
5	captain	2	1	5

Рисунок 28 – содержание таблицы crew до выполнения запроса

	pos character varying (20)	id_flight integer	id_staff integer	id_crew [PK] integer
1	captain	1	6	1
2	first officer	1	2	2
3	service manager	1	7	3
4	flight attendant	1	4	4

Рисунок 29 – содержание таблицы crew после выполнения запроса

После выполнения первой и второй части практического задания, я просмотрела историю выполнения запросов.

Query History	
Show queries generated internally by pgAdmin?	<input checked="" type="checkbox"/>
Today - 15.06.2022	
▶	update schema.crew set id_staff=(select id_staff from... 01:26:35
▶	update schema.crew set id_staff=(select id_staff from... 01:25:54
▶	update schema.crew set id_staff=(select id_staff from... 01:25:46
▶	delete from schema.crew where id_flight=(select id_fl... 01:19:28
▶	insert into schema.crew (pos, id_flight, id_staff) va... 01:13:55
▶	insert into schema.crew (pos, id_flight, id_staff) va... 01:13:47
▶	insert into schema.crew (pos, id_flight, id_staff) va... 01:09:52
▶	select * from schema.crew insert into schema.crew (po... 01:09:35
▶	select * from schema.crew insert into schema.crew (po... 01:08:28
▶	select id_flight from schema.flight, schema.planes wh... 01:06:17
▶	select id_flight from schema.flight, schema.planes wh... 01:06:10
▶	select id_flight from schema.flight, schema.planes, w... 01:05:59

Рисунок 30 – История выполнения запросов

Далее я рассмотрела выполнение первого («определить расчетное время полета по всем маршрутам») и второго («определить расход топлива по всем маршрутам») запросов с использованием EXPLAIN. Время выполнения первого запроса составило 41 мс, второго – 31 мс.


QUERY PLAN	
▲	text 
1	Seq Scan on flight (cost=0.00..20.13 rows=810 width=88) (actual time=0.021..0.024 rows=10 loops=1)
2	[...] Buffers: shared hit=1
3	Planning Time: 4.735 ms
4	Execution Time: 0.040 ms

Рисунок 31 – Часть плана выполнения запроса 1 с использованием EXPLAIN

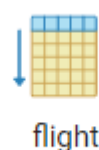


Рисунок 32 – Графическое представление запроса 1

	QUERY PLAN
	text
1	Hash Left Join (cost=49.48..73.90 rows=810 width=100) (actual time=0.042..0.048 rows=10 loops=1)
2	[...] Hash Cond: (planes.id_type = types_of_planes.id_type)
3	[...] Buffers: shared hit=3
4	[...] -> Hash Left Join (cost=34.08..54.31 rows=810 width=40) (actual time=0.030..0.033 rows=10 loops=1)
5	[...] Hash Cond: (flight.id_plane = planes.id_plane)
6	[...] Buffers: shared hit=2
7	[...] -> Seq Scan on flight (cost=0.00..18.10 rows=810 width=40) (actual time=0.011..0.012 rows=10 loops=1)
8	[...] Buffers: shared hit=1
9	[...] -> Hash (cost=20.70..20.70 rows=1070 width=8) (actual time=0.010..0.010 rows=7 loops=1)
10	[...] Buckets: 2048 Batches: 1 Memory Usage: 17kB
11	[...] Buffers: shared hit=1
12	[...] -> Seq Scan on planes (cost=0.00..20.70 rows=1070 width=8) (actual time=0.006..0.007 rows=7 loops=1)
13	[...] Buffers: shared hit=1

Рисунок 33 – Часть плана выполнения запроса 2 с использованием EXPLAIN

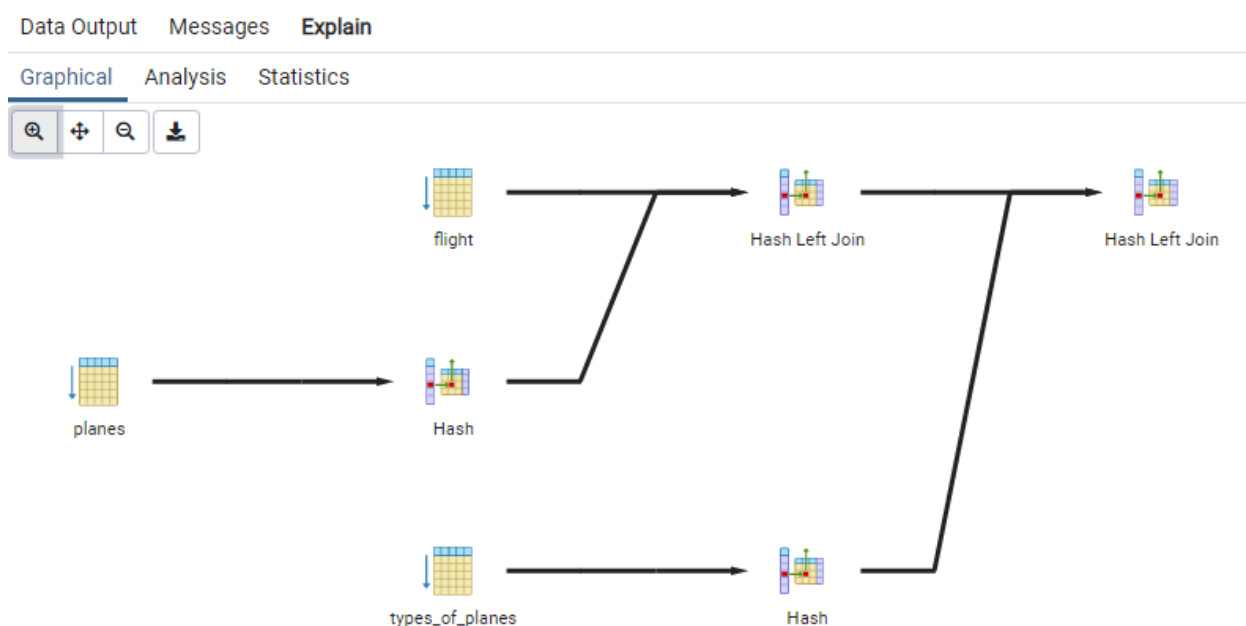


Рисунок 34 – Графическое представление запроса 2

Затем я начала работу с созданием индексов. В первую очередь я создала простой индекс для первого запроса («определить расчетное время полета по всем маршрутам»).

```
1 create index indx on schema.flight(id_flight)
```

Рисунок 35 – создание простого индекса для первого запроса

	QUERY PLAN	
	text	
1	Seq Scan on flight (cost=0.00..1.13 rows=10 width=88) (actual time=0.016..0.019 rows=10 loops=1)	
2	[...] Buffers: shared hit=1	

Рисунок 36 – План выполнения первого запроса после создания индекса

Время выполнения с индексом – 36 мс.

Далее я создала составной индекс для второго запроса («определить расход топлива по всем маршрутам»).

```
1 create index indx1 on schema.flight(id_flight, distance)
```

Рисунок 37 – создание составного индекса для второго запроса

	QUERY PLAN	
	text	
1	Nested Loop Left Join (cost=1.37..28.14 rows=10 width=100) (actual time=0.108..0.130 rows=10 loops=1)	
2	[...] Buffers: shared hit=25	
3	[...] -> Hash Right Join (cost=1.23..26.04 rows=10 width=40) (actual time=0.060..0.068 rows=10 loops=1)	
4	[...] Hash Cond: (planes.id_plane = flight.id_plane)	
5	[...] Buffers: shared hit=5	
6	[...] -> Seq Scan on planes (cost=0.00..20.70 rows=1070 width=8) (actual time=0.009..0.010 rows=7 loops=1)	
7	[...] Buffers: shared hit=1	
8	[...] -> Hash (cost=1.10..1.10 rows=10 width=40) (actual time=0.028..0.028 rows=10 loops=1)	
9	[...] Buckets: 1024 Batches: 1 Memory Usage: 9kB	

Рисунок 38 – Часть плана выполнения второго запроса после создания индекса

Время выполнения запроса с индексом составило 46 мс.

Перед удалением индексов убедимся в их наличии. Затем удалим ранее добавленные индексы indx и indx1.

	indexname name	indexdef text
1	flight_pkey	CREATE UNIQUE INDEX flight_pkey ON schema.flight USING btree (id_flight) INCLUDE (id_flight)
2	rr10	CREATE UNIQUE INDEX rr10 ON schema.flight USING btree (id_flight) INCLUDE (id_flight)
3	indx	CREATE INDEX indx ON schema.flight USING btree (id_flight)
4	indx1	CREATE INDEX indx1 ON schema.flight USING btree (id_flight, distance)

Рисунок 39 – Индексы таблицы schema.flight после добавления indx и indx1

```
1 drop index schema.indx1
```

Рисунок 40 – Пример удаления индекса

	indexname name	indexdef text
1	flight_pkey	CREATE UNIQUE INDEX flight_pkey ON schema.flight USING btree (id_flight) INCLUDE (id_flight)
2	rr10	CREATE UNIQUE INDEX rr10 ON schema.flight USING btree (id_flight) INCLUDE (id_flight)

Рисунок 41 – Индексы таблицы schema.flight после удаления indx и indx1

**Выводы:** SQL-запросы – это удобный способ как извлечь, так и модифицировать данные в таблицах. Я узнала о различных инструментах работы с базой данных в pgadmin4, о создании и применении индексов.