

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: Ананьев Н. В.

Факультет: ИКТ

Группа: K3240

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Цель работы	3
Практическое задание	3
Схема базы данных	4
Выполнение	5
Вывод	15

Цель работы

Цель: 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

Задание 2. Создать запросы:

- Вывести фамилии водителей и номера автобусов, отправившиеся в рейсы до 12 часов текущего дня.
- Рассчитать выручку от продажи билетов за прошедший день.
- Вывести список водителей, которые не выполнили ни одного рейса за прошедший день.
- Вывести сумму убытков из-за непроданных мест в автобусе за прошедшую неделю.
- Сколько рейсов выполнил каждый водитель за последний месяц.
- Вывести тип автобуса, который используется на всех рейсах.
- Вывести данные водителя, который провел максимальное время в пути за прошедшую неделю.

Задание 3. Создать представление для пассажиров:

- количество свободных мест на все рейсы на завтра;
 - самый популярный маршрут этой зимой.
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.

Схема базы данных

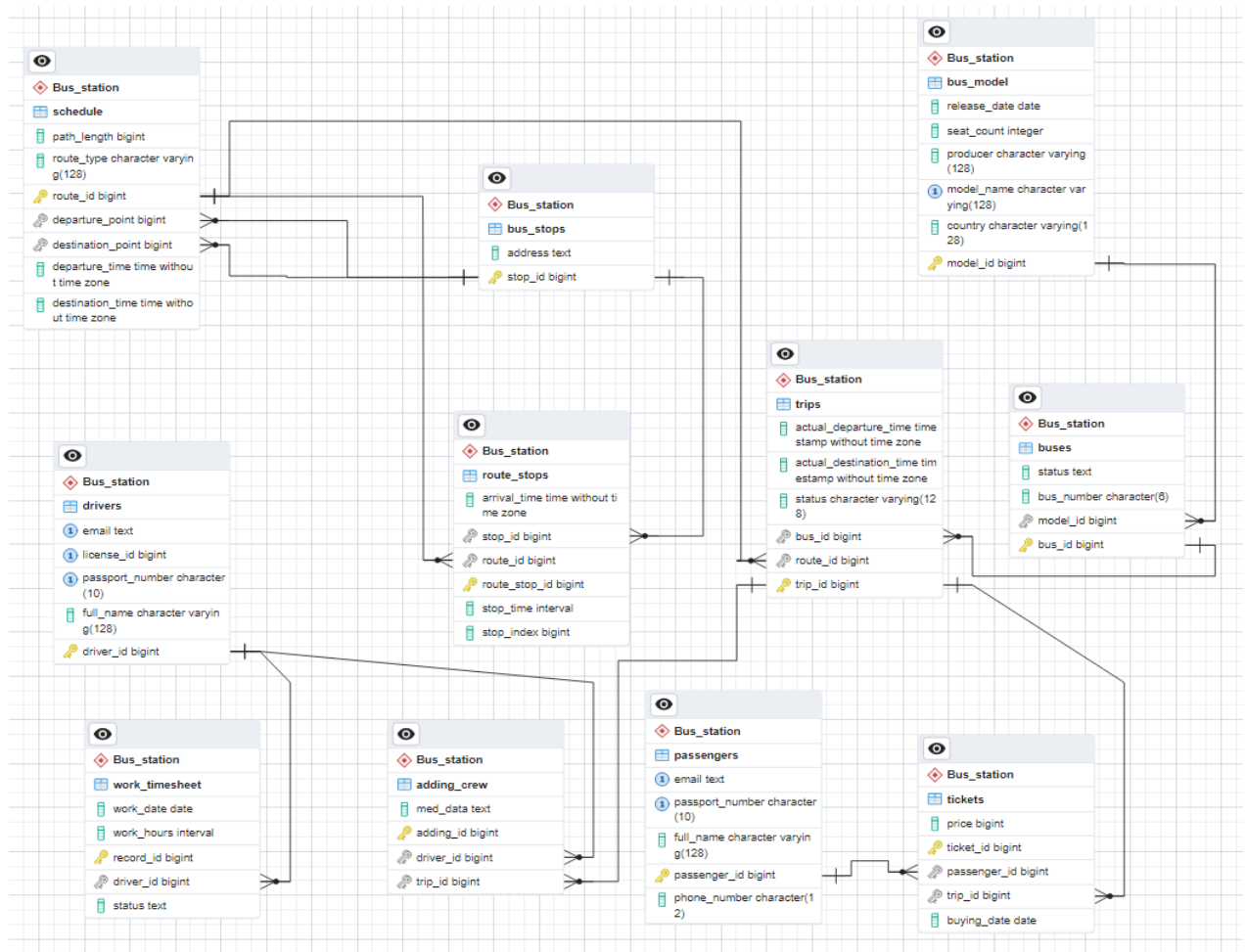


Рисунок 1 — ERD-схема базы данных

1.2 Создание запросов.

ДНЯ:

Query

Query History

```

1 SELECT full_name, license_id, driver_id FROM "Bus_station".drivers
2 WHERE driver_id IN
3 (SELECT driver_id FROM "Bus_station".adding_crew
4 WHERE trip_id IN
5 (SELECT trip_id FROM "Bus_station".trips
6 WHERE actual_departure_time > '2023-11-08 00:00:00' AND actual_departure_time < '2023-11-08 12:00:00'))

```

Data Output

Messages

Notifications

+

📄

▼

📋

▼

🗑️

🔄

⬇️










📶


	full_name character varying (128)	license_id bigint	driver_id [PK] bigint
1	Alexey	1045009	21
2	Ljudmila	1037128	27
3	Nastja	1065543	36
4	Artem	1101680	43
5	Nikolay	1025003	48
6	Nikita	1088706	56
7	Alina	1076340	57
8	Oleg	1067697	58
9	Ivan	1071176	60
10	Ksenia	1024619	61
11	Vlad	1085989	63
12	Natalija	1102619	64
13	Kirill	1091160	66

Рассчитать выручку от продажи билетов за прошедший день:

```
Query    Query History
1  SELECT SUM(price) FROM "Bus_station".tickets
2  WHERE trip_id IN
3  (SELECT trip_id FROM "Bus_station".trips
4  WHERE actual_departure_time > CURRENT_DATE-1 AND actual_departure_time < CURRENT_DATE)
```

Data Output Messages Notifications



	sum numeric 
1	570800

Вывести список водителей, которые не выполнили ни одного рейса за прошедший день.

QueryQuery History

1SELECT * FROM "Bus_station".drivers

2WHERE driver_id NOT IN

3(SELECT driver_id FROM "Bus_station".work_timesheet

4WHERE work_date=CURRENT_DATE-1 AND status = 'ok');|

Data OutputMessagesNotifications

	email text	license_id bigint	passport_number character	full_name character varying (128)	driver_id [PK] bigint
1	driver.Nastja.cda64@yandex.ru	1099101	2020068551	Nastja	32
2	driver.Viktorija.aM22@mail.ru	1032145	2020127540	Viktorija	35
3	driver.Maks.Ej63@gmail.com	1083233	2086690201	Maks	37
4	driver.Shamil.H34@yandex.ru	1077059	2091462780	Shamil	45
5	driver.Nikita.cKyr94@yandex.ru	1090313	2071513811	Nikita	51
6	driver.Anastasiya.ra191@yandex.ru	1044937	2089729314	Anastasiya	52
7	driver.Oleg.JCN86@mail.ru	1058910	2004382910	Oleg	55

Сколько рейсов выполнил каждый водитель за последний месяц.

QueryQuery History

1SELECT COUNT(*), driver_id FROM "Bus_station".adding_crew

2WHERE trip_id IN

3(SELECT trip_id FROM "Bus_station".trips

4WHERE actual_departure_time > CURRENT_DATE-30 AND actual_destination_time < CURRENT_DATE)

5GROUP BY driver_id

6

Data OutputMessagesNotifications

	count bigint	driver_id bigint
1	15	62
2	14	55
3	10	27
4	15	23
5	8	56
6	16	58
7	14	44
8	11	42
9	12	29
10	15	54
11	15	68
12	9	34

Вывести сумму убытков из-за непроданных мест в автобусе за прошедшую неделю:

```
Query Query History
1 SELECT SUM((P.seat_count - P.count) * P.price) FROM
2 (SELECT * FROM
3     (SELECT count(*), trip_id
4       FROM "Bus_station".tickets
5       GROUP BY trip_id) AS A
6 JOIN "Bus_station".trips
7 ON A.trip_id = "Bus_station".trips.trip_id
8 JOIN "Bus_station".buses
9 ON "Bus_station".trips.bus_id = "Bus_station".buses.bus_id
10 JOIN "Bus_station".bus_model
11 ON "Bus_station".buses.model_id = "Bus_station".bus_model.model_id
12 WHERE actual_departure_time > CURRENT_DATE - 8) AS P
13
```

Data Output Messages Notifications

	sum numeric 🔒
1	511400

Вывести тип автобуса, который используется на всех рейсах. (такого автобуса не оказалось)

```
Query Query History
1 SELECT P.release_date, P.seat_count, P.producer, P.model_name, P.country FROM
2 (SELECT * FROM
3 (SELECT count(DISTINCT(route_id)), bus_id FROM "Bus_station".trips
4 GROUP BY bus_id) AS A
5 JOIN "Bus_station".buses ON A.bus_id = "Bus_station".buses.bus_id
6 JOIN "Bus_station".bus_model ON "Bus_station".buses.model_id = "Bus_station".bus_model.model_id) AS P
7 WHERE P.count IN (SELECT COUNT(*) FROM "Bus_station".schedule);
8
```

Data Output Messages Notifications

						
release_date	seat_count	producer	model_name	country		
date	integer	character varying (128)	character varying (128)	character varying (128)		

Вывести данные водителя, который провел максимальное время в пути за прошедшую неделю:

QueryQuery History

12345678910

```
SELECT * FROM "Bus_station".drivers
WHERE driver_id IN(
SELECT A.driver_id FROM
(SELECT SUM(work_hours), driver_id FROM "Bus_station".work_timesheet
WHERE work_date > CURRENT_DATE - 30
GROUP BY driver_id) AS A
WHERE A.sum >= ALL(SELECT P.sum FROM (SELECT SUM(work_hours), driver_id
FROM "Bus_station".work_timesheet
WHERE work_date > CURRENT_DATE - 30
GROUP BY driver_id) AS P));
```

Data OutputMessagesNotifications

	email text	license_id bigint	passport_number character	full_name character varying (128)	driver_id [PK] bigint
1	driver.Artem.MIC08@gmail.com	1033892	2054233271	Artem	65

1.3 Создание представлений










Создание представления для количества свободных мест на завтра:


```
Query Query History
1 CREATE VIEW TomorrowFreeSeatCount AS
2   SELECT SUM(P.seat_count - P.count) FROM
3   (SELECT * FROM
4     (SELECT count(*), trip_id
5      FROM "Bus_station".tickets
6      GROUP BY trip_id) AS A
7   JOIN "Bus_station".trips
8   ON A.trip_id = "Bus_station".trips.trip_id
9   JOIN "Bus_station".buses
10  ON "Bus_station".trips.bus_id = "Bus_station".buses.bus_id
11  JOIN "Bus_station".bus_model
12  ON "Bus_station".buses.model_id = "Bus_station".bus_model.model_id
13  WHERE actual_departure_time > CURRENT_DATE + 1 AND actual_departure_time < CURRENT_DATE + 2) AS P
```

Query Query History

```
1 SELECT * FROM TomorrowFreeSeatCount
```

Data Output Messages Notifications



	sum numeric 
1	204

Создание представления для самого популярного маршрута этой зимой:

QueryQuery History

```
1 CREATE VIEW MostPopularWinterTrip AS
2   SELECT A.route_id FROM
3     (SELECT COUNT(*), route_id FROM "Bus_station".trips
4      WHERE (actual_departure_time > '2023-12-01' AND actual_departure_time < '2024-02-28')
5      GROUP BY route_id) AS A
6   WHERE A.count >= ALL(SELECT P.count FROM(SELECT COUNT(*), route_id FROM "Bus_station".trips
7     WHERE (actual_departure_time > '2023-12-01' AND actual_departure_time < '2024-02-28')
8     GROUP BY route_id) AS P)
```

QueryQuery History

1 SELECT * FROM MostPopularWinterTrip

Data OutputMessagesNotifications

≡+

▼

▼

	route_id bigint	
1	205	

2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов:

1. INSERT

Query Query History

1 INSERT INTO "Bus_station".adding_crew (med_data, driver_id, trip_id, status)

2 VALUES

3 (

4 'some medical data',

5 (SELECT FLOOR(RANDOM()*(50)) + 21),

6 (SELECT trip_id FROM "Bus_station".trips

7 WHERE (actual_departure_time > '2023-12-01') LIMIT 1),

8 'допущен'

9)

Query Query History

1 SELECT * FROM "Bus_station".adding_crew

1747	some medical data	2114	27	1752	[null]
1748	some medical data	2115	39	1753	[null]
1749	yet medical data	2116	68	1754	[null]
1750	another medical info	2117	43	1755	[null]
1751	some medical data	2118	39	2464	допущен

2. UPDATE

Query Query History

1 UPDATE "Bus_station".trips

2 SET price = price + 100

3 WHERE trip_id IN

4 (SELECT trip_id FROM "Bus_station".trips

5 WHERE (actual_departure_time > '2023-12-02' AND bus_id=39) LIMIT 1)

6

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 55 msec.

	actual_departure_time timestamp without time zone	actual_destination_time timestamp without time zone	status character varying (128)	bus_id bigint	route_id bigint	trip_id [PK] bigint	price bigint
2470	2023-11-30 20:02:00	2023-11-30 22:25:00	planned	40	219	2458	700
2471	2023-11-30 08:20:00	2023-11-30 14:20:00	planned	69	221	2459	600
2472	2023-11-30 16:30:00	2023-11-30 22:31:00	planned	43	223	2460	300
2473	2023-11-30 15:10:00	2023-11-30 15:19:00	planned	66	226	2461	400
2474	2023-11-30 18:14:00	2023-11-30 22:57:00	planned	45	227	2462	400
2475	2023-11-30 10:44:00	2023-11-30 14:21:00	planned	36	230	2463	300
2476	2023-12-02 13:47:00	2023-12-02 16:12:00	planned	39	210	2483	600
2477	2023-12-02 17:47:00	2023-12-02 21:09:00	planned	70	211	2484	500

2921	2023-12-30 08:33:00	2023-12-30 09:56:00	planned	77	208	2929	600
2922	2023-12-30 10:17:00	2023-12-30 12:06:00	planned	89	209	2930	600
2923	2023-12-30 13:51:00	2023-12-30 16:10:00	planned	71	210	2931	300
2924	2023-12-30 13:11:00	2023-12-30 18:28:00	planned	69	213	2932	600
2925	2023-12-30 17:33:00	2023-12-30 23:31:00	planned	83	214	2933	300
2926	2023-12-01 19:36:00	2023-12-01 21:23:00	planned	56	199	2464	900
2927	2023-12-02 13:47:00	2023-12-02 16:12:00	planned	39	210	2483	700

Total rows: 2927 of 2927	Query complete 00:00:00.095
--------------------------	-----------------------------

3. DELETE

Query

Query History

```

1 DELETE FROM "Bus_station".tickets
2 WHERE
3 (passenger_id IN (SELECT passenger_id FROM "Bus_station".passengers
4                     WHERE email='Danil.iB05@mail.ru')
5 AND trip_id IN (SELECT trip_id FROM "Bus_station".trips
6                  WHERE (actual_departure_time > '2022-12-31')))
```

Data Output

Messages

Notifications

DELETE 63

Query returned successfully in 104 msec.

4. Индексы.

Query

Query History

1

EXPLAIN (ANALYZE) SELECT * FROM "Bus_station".passengers










2

WHERE full_name LIKE 'Vlad%';

Data Output


Messages

Notifications



QUERY PLAN

text



1

Seq Scan on passengers (cost=0.00..35.50 rows=31 width=60) (actual time=0.021..0.128 rows=30 loops...

2

Filter: ((full_name)::text ~~ 'Vlad% '::text)

3

Rows Removed by Filter: 970

4

Planning Time: 0.124 ms

5

Execution Time: 0.141 ms

Query

Query History

1

CREATE INDEX Name_index ON "Bus_station".passengers(full_name text_pattern_ops)

2

-- EXPLAIN (ANALYZE) SELECT * FROM "Bus_station".passengers

3

-- WHERE full_name LIKE 'Vlad%'

4

5

Data Output

Messages

Notifications

CREATE INDEX

Query returned successfully in 32 msec.

Query

Query History

1

-- CREATE INDEX Name_index ON "Bus_station".passengers(full_name text_pattern_ops)

2

EXPLAIN (ANALYZE) SELECT * FROM "Bus_station".passengers

3










WHERE full_name LIKE 'Vlad%'

4

Data Output


Messages

Notifications



QUERY PLAN

text



1

Bitmap Heap Scan on passengers (cost=4.60..29.20 rows=31 width=60) (actual time=0.032..0.048 rows=30 loops=1)

2

Filter: ((full_name)::text ~~ 'Vlad% '::text)

3

Heap Blocks: exact=14

4

-> Bitmap Index Scan on name_index (cost=0.00..4.60 rows=32 width=0) (actual time=0.027..0.027 rows=30 loops...

5

Index Cond: (((full_name)::text ~>= ~ 'Vlad'::text) AND ((full_name)::text ~< ~ 'Vlae'::text))

6





Planning Time: 1.100 ms

7










Execution Time: 0.070 ms

Время выполнения запроса уменьшилось в 2 раза.

Составные индексы.

Query	Query History
1	EXPLAIN (ANALYZE) SELECT * FROM "Bus_station".passengers
2	WHERE passport_number LIKE '2023%' OR phone_number LIKE '+7911%'
3	
Data Output	Messages
       	
QUERY PLAN 	
text	
1	Seq Scan on passengers (cost=0.00..38.00 rows=20 width=60) (actual time=0.015..0.135 rows=21 loops...
2	Filter: ((passport_number ~~ '2023%':text) OR (phone_number ~~ '+7911%':text))
3	Rows Removed by Filter: 979
4	Planning Time: 0.095 ms
5	Execution Time: 0.146 ms

Query	Query History
1	CREATE INDEX passport_phone_index ON "Bus_station".passengers(passport_number bpchar_pattern_ops, phone_number bpchar_pattern_ops)
2	
3	-- EXPLAIN (ANALYZE) SELECT * FROM "Bus_station".passengers
4	-- WHERE passport_number LIKE '2023%' OR phone_number LIKE '+7911%'
5	
Data Output	Messages
CREATE INDEX	
Query returned successfully in 32 msec.	

Query	Query History
1	-- CREATE INDEX passport_phone_index ON "Bus_station".passengers(passport_number bpchar_p
2	SET enable_seqscan TO off;
3	EXPLAIN (ANALYZE) SELECT * FROM "Bus_station".passengers
4	WHERE passport_number LIKE '2023%' OR phone_number LIKE '+7911%'
5	
Data Output	Messages
       	
QUERY PLAN 	
text	
1	Bitmap Heap Scan on passengers (cost=42.66..66.19 rows=20 width=60) (actual time=0.113..0.127 rows=21 loops=1)
2	Recheck Cond: ((passport_number ~~ '2023%':text) OR (phone_number ~~ '+7911%':text))
3	Filter: ((passport_number ~~ '2023%':text) OR (phone_number ~~ '+7911%':text))
4	Heap Blocks: exact=11
5	-> BitmapOr (cost=42.66..42.66 rows=20 width=0) (actual time=0.107..0.108 rows=0 loops=1)
6	-> Bitmap Index Scan on passport_phone_index (cost=0.00..4.38 rows=10 width=0) (actual time=0.026..0.026 rows=8 loops=1)
7	Index Cond: ((passport_number >= '2023':bpchar) AND (passport_number <= '2024':bpchar))
8	-> Bitmap Index Scan on passport_phone_index (cost=0.00..38.27 rows=10 width=0) (actual time=0.081..0.081 rows=13 loop...
9	Index Cond: ((phone_number >= '+7911':bpchar) AND (phone_number <= '+7912':bpchar))
10	Planning Time: 0.176 ms
11	Execution Time: 0.150 ms

В данном запросе при использовании составного индекса скорость выполнения упала на ~6%, Это связано с тем, что сканирование индекса требует нескольких операций ввода-вывода для каждой строки. В то время как последовательное сканирование требует только одного ввода-вывода для каждой строки. Поэтому когда выборка содержит 5-10 % данных таблицы, эффективнее будет последовательное сканирование.

Вывод

В ходе выполнения данной лабораторной работы я обрел практические навыки применения SQL запросов, в том числе запросов направленных на редактирование данных и вложенных запросов; а также узнал что такое представления и как с их помощью можно сэкономить время при выполнении однотипных запросов. Ознакомился с синтаксисом создания простых и составных индексов и с тем, как проанализировать их эффективность с помощью плана запросов.