

**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: Филиппов А.Э.

Факультет: ИКТ

Группа: K3239

Преподаватель: Говорова М.М.

**ИТМО**

Санкт-Петербург 2023

## Оглавление

1. Цель работы.....	3
2. Вариант 6. БД «Пассажир».....	3-4
3. Выполнение ЛР.....	4-15
Вывод.....	15

**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, 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.

### **Вариант 6. БД «Пассажир»**

Описание предметной области: Информационная система служит для продажи железнодорожных билетов. Билеты могут продаваться на текущие сутки или предварительно (не более чем за 45 суток). Цена билета при предварительной продаже снижается на 5%. Билет может быть приобретен в кассе или онлайн. Если билет приобретен в кассе, необходимо знать, в какой. Для каждой кассы известны номер и адрес. Кассы могут располагаться в различных населенных пунктах.

Поезда курсируют по расписанию, но могут назначаться дополнительные поезда на заданный период или определенные даты.

По всем промежуточным остановкам на маршруте известны название, тип населенного пункта, время прибытия, отправления, время стоянки.

Необходимо учитывать, что местом посадки и высадки пассажира могут быть промежуточные пункты по маршруту.

БД должна содержать следующий минимальный набор сведений: Номер поезда. Название поезда. Тип поезда. Пункт назначения. Пункт назначения для проданного билета. Номер вагона. Тип вагона. Количество мест в вагоне. Цена билета. Дата отправления. Дата прибытия. Дата прибытия для пункта назначения проданного билета. Время отправления. Номер вагона в поезде. Номер билета. Место. Тип места. Фамилия пассажира. Имя пассажира. Отчество пассажира. Паспортные данные.

**Задание 1.1 (ЛР 1 БД).** Выполните инфологическое моделирование базы данных системы. (Ограничения задать самостоятельно.)

**Задание 1.2.** Создайте логическую модель БД, используя ИЛМ (задание 1.1). Используйте необходимые средства поддержки целостности данных в СУБД.

## **Задание 2. Создать запросы:**

- Свободные места на все поезда, отправляющиеся с вокзала в течение следующих суток.
- Список пассажиров, отправившихся в Москву всеми рейсами за прошедшие сутки.
- Номера поездов, на которые проданы все билеты на следующие сутки.
- Свободные места в купейные вагоны всех рейсов до Москвы на текущие сутки.
- Выручка от продажи билетов на все поезда за прошедшие сутки.
- Общее количество билетов, проданных по всем направлениям в вагоны типа "СВ".
- Номера и названия поездов, все вагоны которых были заполнены менее чем наполовину за прошедшие сутки.

## **Задание 3. Создать представление:**

- для пассажиров о наличии свободных мест на заданный рейс;
- количество непроданных билетов на все поезда, формирующиеся за прошедшие сутки (номер поезда, тип вагона, количество).

## **Задание 4. Создать хранимые процедуры:**

- Для повышения цен в пригородные поезда на 20%.
- Для создания нового рейса на поезд.
- Для формирования общей выручки по продаже билетов за сутки.

## **Выполнение ЛР**

### **1. Запросы к базе данных**

- Свободные места на все поезда, отправляющиеся с вокзала в течение следующих суток.

```
SELECT DISTINCT seat.*, schedule.planned_departure FROM schedule
INNER JOIN carriage ON carriage.schedule_id=schedule.id
INNER JOIN seat ON carriage.id=seat.carriage_id
LEFT OUTER JOIN ticket ON ticket.seat_code=seat.code
INNER JOIN route ON schedule.route_id=route.id
```

WHERE ticket.seat\_code is null AND (schedule.planned\_departure BETWEEN NOW() AND NOW() + INTERVAL '24 HOURS') AND (route.route\_point\_departure\_id = 1)

Query Query History

```

1 SELECT DISTINCT seat.*, schedule.planned_departure FROM schedule
2 INNER JOIN carriage ON carriage.schedule_id=schedule.id
3 INNER JOIN seat ON carriage.id=seat.carriage_id
4 LEFT OUTER JOIN ticket ON ticket.seat_code=seat.code
5 INNER JOIN route ON schedule.route_id=route.id
6 WHERE ticket.seat_code is null AND (schedule.planned_departure BETWEEN NOW() AND NOW() + INTERVAL '24 HOURS') AND (route.route_point_departure_id = 1)

```

Data Output Messages Notifications

	code character varying	number integer	type smallint	carriage_id bigint	manufacturer character varying	planned_departure timestamp with time zone
1	03A-1	3	1	3	Alstom	2023-11-12 20:13:00+03
2	05B-2	10	1	4	VR Group	2023-11-12 20:13:00+03

- Список пассажиров, отправившихся в Москву всеми рейсами за прошедшие сутки.

```

SELECT DISTINCT passenger.* FROM ticket
INNER JOIN passenger ON passenger.id = ticket.passenger_id
INNER JOIN seat ON ticket.seat_code = seat.code
INNER JOIN carriage ON seat.carriage_id = carriage.id
INNER JOIN schedule ON carriage.schedule_id = schedule.id
INNER JOIN route ON schedule.route_id = route.id
INNER JOIN route_point ON route.id = route_point.route_id
INNER JOIN station ON station.id = route_point.station_id
WHERE station.name = 'Москва Октябрьская';

```

```

1 SELECT DISTINCT passenger.* FROM ticket
2 INNER JOIN passenger ON passenger.id = ticket.passenger_id
3 INNER JOIN seat ON ticket.seat_code = seat.code
4 INNER JOIN carriage ON seat.carriage_id = carriage.id
5 INNER JOIN schedule ON carriage.schedule_id = schedule.id
6 INNER JOIN route ON schedule.route_id = route.id
7 INNER JOIN route_point ON route.id = route_point.route_id
8 INNER JOIN station ON station.id = route_point.station_id
9 WHERE station.name = 'Москва Октябрьская';|

```

Data Output Messages Notifications

	id [PK] bigint	fio character varying (255)	birth_place character varying (255)	passport bigint	birth_date date
1	3	Зеленский Володимир Олександрович	Киев	367585	1980-02-24
2	4	Поздняков Владислав Андреевич	Актобе	856734	1987-01-30
3	5	Гоев Александр Прохорович	Самосир	9789213	1991-01-31
4	6	Стручков Тимур Максимович	Лос-Анджелес	9324324	2004-02-23

- Номера поездов, на которые проданы все билеты на следующие сутки.

```

SELECT SUM(ticket_count) as ticket_count, SUM(place_count) as place_count,
train_name, train_type, schedule_id FROM
(SELECT COUNT(schedule.id) as ticket_count, schedule.id as
schedule_id, route.train_name, route.train_type, carriage.id,
carriage_type.place_count
FROM schedule
INNER JOIN route ON route.id=schedule.route_id
RIGHT JOIN carriage ON carriage.schedule_id=schedule.id
RIGHT JOIN carriage_type ON
carriage.carriage_type_id=carriage_type.id
INNER JOIN seat ON carriage.id=seat.carriage_id
INNER JOIN ticket ON ticket.seat_code=seat.code
WHERE schedule.planned_departure BETWEEN NOW() and NOW() + INTERVAL
'24 HOURS'
GROUP BY (schedule.id, route.train_name, route.train_type,
carriage.id, carriage_type.place_count)) as carriage
GROUP BY (schedule_id, train_name, train_type)
HAVING SUM(ticket_count) = SUM(place_count);

```

```

1 SELECT SUM(ticket_count) as ticket_count, SUM(place_count) as place_count, train_name, train_type, schedule_id FROM
2   (SELECT COUNT(schedule.id) as ticket_count, schedule.id as schedule_id, route.train_name, route.train_type, carriage.id
3    FROM schedule
4    INNER JOIN route ON route.id=schedule.route_id
5    RIGHT JOIN carriage ON carriage.schedule_id=schedule.id
6    RIGHT JOIN carriage_type ON carriage.carriage_type_id=carriage_type.id
7    INNER JOIN seat ON carriage.id=seat.carriage_id
8    INNER JOIN ticket ON ticket.seat_code=seat.code
9    WHERE schedule.planned_departure BETWEEN NOW() and NOW() + INTERVAL '24 HOURS'
10   GROUP BY (schedule.id, route.train_name, route.train_type, carriage.id, carriage_type.place_count)) as carriage
11 GROUP BY (schedule_id, train_name, train_type)
12 HAVING SUM(ticket_count) = SUM(place_count);

```

Data Output Messages Notifications

	ticket_count numeric	place_count bigint	train_name character varying (255)	train_type character varying (255)	schedule_id bigint
1	7	7	Анжерпо-2	Сапсан	6

- Свободные места в купейные вагоны всех рейсов до Москвы на текущие сутки.

```

SELECT DISTINCT schedule.planned_departure, carriage.id, seat.code FROM station
INNER JOIN route_point ON station.name = 'Москва Октябрьская'
INNER JOIN schedule ON route_point.route_id = schedule.route_id
INNER JOIN carriage ON carriage.schedule_id = schedule.id
INNER JOIN carriage_type ON carriage.carriage_type_id = carriage_type.id
INNER JOIN seat ON seat.carriage_id = carriage.id
WHERE (carriage_type.name = 'Kyne') AND (schedule.planned_departure BETWEEN
NOW() and NOW() + INTERVAL '24 HOURS');

```

```

1 SELECT DISTINCT schedule.planned_departure, carriage.id, seat.code FROM station
2 INNER JOIN route_point ON station.name = 'Москва Октябрьская'
3 INNER JOIN schedule ON route_point.route_id = schedule.route_id
4 INNER JOIN carriage ON carriage.schedule_id = schedule.id
5 INNER JOIN carriage_type ON carriage.carriage_type_id = carriage_type.id
6 INNER JOIN seat ON seat.carriage_id = carriage.id
7 WHERE (carriage_type.name = 'Kyne') AND (schedule.planned_departure BETWEEN NOW() and

```

Data Output Messages Notifications

	planned_departure timestamp with time zone	id bigint	code character varying
1	2023-11-13 20:13:00+03	4	058-2

- Выручка от продажи билетов на все поезда за прошедшие сутки.

```

SELECT SUM(ticket.price) FROM ticket
INNER JOIN seat on seat.code = ticket.seat_code
INNER JOIN carriage on carriage.id = seat.carriage_id
INNER JOIN schedule on schedule.id = carriage.schedule_id
WHERE schedule.planned_departure BETWEEN NOW() - INTERVAL '24 HOURS' AND
NOW() AND ticket.pay_status = 1;
1 SELECT SUM(ticket.price) FROM ticket
2 INNER JOIN seat on seat.code = ticket.seat_code
3 INNER JOIN carriage on carriage.id = seat.carriage_id
4 INNER JOIN schedule on schedule.id = carriage.schedule_id
5 WHERE schedule.planned_departure BETWEEN NOW() - INTERVAL '24 HOURS' AND NOW() AND ti

```

Data Output	Messages	Notifications
<div> <div>+</div> <div>📄</div> <div>▼</div> <div>📄</div> <div>▼</div> <div>🗑️</div> <div>🔄</div> <div>📶</div> <div>⬇️</div> <div>📈</div> </div>		
	<div>sum</div> <div>double precision 🔒</div>	
1	119998.8	

- Общее количество билетов, проданных по всем направлениям в вагоны типа “СВ”.

```

SELECT COUNT(*) FROM ticket
INNER JOIN seat on seat.code = ticket.seat_code
INNER JOIN carriage on carriage.id = seat.carriage_id
INNER JOIN carriage_type on carriage_type.id = carriage.carriage_type_id
INNER JOIN schedule on schedule.id = carriage.schedule_id
WHERE (carriage_type.name = 'Барон "СВ"') AND (schedule.planned_departure
BETWEEN NOW() and NOW() + INTERVAL '24 HOURS');

```



```

1 SELECT COUNT(*) FROM ticket
2 INNER JOIN seat on seat.code = ticket.seat_code
3 INNER JOIN carriage on carriage.id = seat.carriage_id
4 INNER JOIN carriage_type on carriage_type.id = carriage.carriage_type_id
5 INNER JOIN schedule on schedule.id = carriage.schedule_id
6 WHERE (carriage_type.name = 'Вагон "CB"') AND (schedule.planned_departure BETWEEN NOW

```

Data Output Messages Notifications



	count	
	bigint	
1		2

- Номера и названия поездов, все вагоны которых были заполнены менее чем наполовину за прошедшие сутки.

```

SELECT COUNT(schedule.id) as ticket_count, schedule.id as schedule_id,
route.train_name, route.train_type, carriage.id, carriage_type.place_count
FROM schedule
INNER JOIN route ON route.id=schedule.route_id
RIGHT JOIN carriage ON carriage.schedule_id=schedule.id
RIGHT JOIN carriage_type ON carriage.carriage_type_id=carriage_type.id
INNER JOIN seat ON carriage.id=seat.carriage_id
INNER JOIN ticket ON ticket.seat_code=seat.code
WHERE schedule.planned_departure BETWEEN NOW() and NOW() + INTERVAL '24
HOURS'
GROUP BY (schedule.id, route.train_name, route.train_type, carriage.id,
carriage_type.place_count)
HAVING COUNT(schedule.id) < place_count/2

```

```

1 SELECT COUNT(schedule.id) as ticket_count, schedule.id as schedule_id, route.train_name, route.train_type, carriage.id, carri
2 FROM schedule
3 INNER JOIN route ON route.id=schedule.route_id
4 RIGHT JOIN carriage ON carriage.schedule_id=schedule.id
5 RIGHT JOIN carriage_type ON carriage.carriage_type_id=carriage_type.id
6 INNER JOIN seat ON carriage.id=seat.carriage_id
7 INNER JOIN ticket ON ticket.seat_code=seat.code
8 WHERE schedule.planned_departure BETWEEN NOW() and NOW() + INTERVAL '24 HOURS'
9 GROUP BY (schedule.id, route.train_name, route.train_type, carriage.id, carriage_type.place_count)
10 HAVING COUNT(schedule.id) < place_count/2

```

Data Output Messages Notifications



	ticket_count bigint	schedule_id bigint	train_name character varying (255)	train_type character varying (255)	id bigint	place_count smallint
1	2	1	Аллерго	Скоростной	3	6

## 2. Создать представления

- для пассажиров о наличии свободных мест на заданный рейс

```

CREATE VIEW places AS
SELECT schedule.id, seat.code, carriage.id as carriage_id,
schedule.planned_departure FROM schedule
INNER JOIN carriage ON carriage.schedule_id = schedule.id
INNER JOIN seat ON seat.carriage_id = carriage.id
LEFT OUTER JOIN ticket on ticket.seat_code = seat.code
WHERE ticket.seat_code is NULL;

```

```

1 SELECT * FROM places

```

Data Output Messages Notifications



	id bigint	code character varying	carriage_id bigint	planned_departure timestamp with time zone
1	1	03A-1	3	2023-11-13 20:13:00+03
2	1	058-2	4	2023-11-13 20:13:00+03

- количество непроданных билетов на все поезда, формирующиеся за прошедшие сутки (номер поезда, тип вагона, количество).

```
CREATE VIEW free_tickets_count as
SELECT COUNT(*) FROM seat
LEFT OUTER JOIN ticket on ticket.seat_code = seat.code
INNER JOIN carriage ON seat.carriage_id = carriage.id
INNER JOIN schedule ON carriage.schedule_id = schedule.id
WHERE ticket.seat_code is NULL AND (schedule.planned_departure BETWEEN NOW()
- INTERVAL '24 HOURS' AND NOW());
```

```
1 SELECT * FROM free_tickets_count;
```

Data Output		Messages	Notifications
<div> <div>+</div> <div>📄</div> <div>▼</div> <div>📄</div> <div>▼</div> <div>🗑️</div> <div>📦</div> <div>⬇️</div> <div>📈</div> </div>			
	count bigint		
1	2		

### 3. Создать хранимые процедуры:

- Для повышения цен в пригородные поезда на 20%.

```
CREATE PROCEDURE increase_ticket_prices_by20percent()
LANGUAGE SQL
AS $$
UPDATE ticket SET price = price*1.2 WHERE ticket.id IN (
SELECT ticket.id FROM schedule
INNER JOIN route ON schedule.route_id = route.id
INNER JOIN carriage ON carriage.schedule_id = schedule.id
INNER JOIN seat ON carriage.id = seat.carriage_id
INNER JOIN ticket ON seat.code = ticket.seat_code
```

```
WHERE schedule.planned_departure > NOW() AND route.train_type =
'Пригородный')
$$;
```

1 SELECT \* FROM ticket;

	id [PK] bigint	price double precision	passenger_id bigint	cashbox_id bigint	seat_code character varying (255)	is_online boolean	pay_status smallint	route_point_departure_id bigint	route_point_arrival_id bigint
1	2	5670	1	1	05A-1	true	1	1	14
2	3	5800	2	1	03B-1	false	1	1	14
3	4	9999	4	1	СБУ-31	true	1	1	21
4	5	9999	3	1	СБУ-30	true	1	1	21
5	7	15000	6	1	КЕК-5	true	1	1	21
6	6	18000	5	1	КЕК-6	false	1	1	21
7	9	18000	6	1	КЕК-7	true	1	1	21
8	11	18000	6	1	КЕК-8	true	1	1	21
9	12	18000	6	1	КЕК-9	true	1	1	21
10	13	18000	6	1	КЕК-10	true	1	1	21
11	14	18000	6	1	КЕК-11	true	1	1	21
12	15	11998.8	6	1	АБУ-2	true	1	1	21

- Для создания нового рейса на поезд.

```
CREATE PROCEDURE create_new_route(train_name varchar, train_type varchar,
route_point_departure_id integer, route_point_arrival_id integer)
LANGUAGE SQL
AS $$
INSERT INTO route("train_name", "train_type", "route_point_departure_id",
"route_point_arrival_id") VALUES(train_name, train_type,
route_point_departure_id, route_point_arrival_id)
$$;
```

```
CALL create_new_route('Ураган', 'Скоростной', 1, 14);
SELECT * FROM route;
```

Output Messages Notifications

id [PK] bigint	train_name character varying (255)	train_type character varying (255)	route_point_departure_id bigint	route_point_arrival_id bigint	departure_time time with time zone	arrival_time time with time zone
1	Аллерго	Скоростной	1	14	[null]	[null]
2	Аллерго-2	Сапсан	1	21	[null]	[null]
5	Ураган	Скоростной	1	14	[null]	[null]

- Для формирования общей выручки по продаже билетов за сутки

```
CREATE OR REPLACE PROCEDURE get_last24hr_revenue(
  INOUT summ DOUBLE PRECISION DEFAULT 0)
AS $$
SELECT SUM(price) FROM ticket
  INNER JOIN seat ON ticket.seat_code = seat.code
  INNER JOIN carriage ON carriage.id = seat.carriage_id
  INNER JOIN schedule ON carriage.schedule_id = schedule.id
WHERE schedule.planned_departure BETWEEN NOW() - INTERVAL '24 HOURS' and
NOW()
$$ LANGUAGE SQL;
```

```

1 CREATE OR REPLACE PROCEDURE get_last24hr_revenue(
2 INOUT summ DOUBLE PRECISION DEFAULT 0)
3 AS $$
4 SELECT SUM(price) FROM ticket
5 INNER JOIN seat ON ticket.seat_code = seat.code
6 INNER JOIN carriage ON carriage.id = seat.carriage_id
7 INNER JOIN schedule ON carriage.schedule_id = schedule.id
8 WHERE schedule.planned_departure BETWEEN NOW() - INTERVAL '24 HOURS' and NOW()
9 $$ LANGUAGE SQL;
10 CALL get_last24hr_revenue();

```

Data Output Messages Notifications



	summ double precision	
1	19998	

1. UPDATE – (перенесем рейсы, поезда которых заполнены менее чем на 50% на 12 часов)

```

UPDATE schedule
SET planned_departure = planned_departure + INTERVAL '12 HOURS'
WHERE id IN (
SELECT schedule.id
FROM schedule
INNER JOIN route ON route.id=schedule.route_id
RIGHT JOIN carriage ON carriage.schedule_id=schedule.id
RIGHT JOIN carriage_type ON
carriage.carriage_type_id=carriage_type.id
INNER JOIN seat ON carriage.id=seat.carriage_id
INNER JOIN ticket ON ticket.seat_code=seat.code
WHERE schedule.planned_departure BETWEEN NOW() and NOW() +
INTERVAL '24 HOURS'
GROUP BY (schedule.id, route.train_name, route.train_type,
carriage.id, carriage_type.place_count)
HAVING COUNT(schedule.id) < place_count/2);

```

до:

	id [PK] bigint	planned_departure timestamp with time zone	factual_departure timestamp with time zone	factual_arrival timestamp with time zone	route_id bigint
1	1	2023-11-13 20:13:00+03	[null]	[null]	1
2	2	2023-11-11 20:13:00+03	[null]	[null]	1
3	3	2023-11-13 20:13:00+03	[null]	[null]	1
4	4	2023-11-12 09:10:00+03	2023-11-11 09:10:00+03	2023-11-11 13:09:00+03	2
5	5	2023-11-11 09:10:00+03	[null]	[null]	2
6	6	2023-11-13 09:10:00+03	[null]	[null]	2

после:

	id [PK] bigint	planned_departure timestamp with time zone	factual_departure timestamp with time zone	factual_arrival timestamp with time zone	route_id bigint
1	1	2023-11-14 08:13:00+03	[null]	[null]	1
2	2	2023-11-11 20:13:00+03	[null]	[null]	1
3	3	2023-11-13 20:13:00+03	[null]	[null]	1
4	4	2023-11-12 09:10:00+03	2023-11-11 09:10:00+03	2023-11-11 13:09:00+03	2
5	5	2023-11-11 09:10:00+03	[null]	[null]	2
6	6	2023-11-13 09:10:00+03	[null]	[null]	2

2. DELETE — (удалить рейсы в течение следующих суток, на которых не были продано ни одного билета)

```
DELETE FROM schedule WHERE id IN (
SELECT schedule.id FROM schedule
LEFT OUTER JOIN carriage ON carriage.schedule_id = schedule.id
LEFT OUTER JOIN seat ON seat.carriage_id = carriage.id
LEFT OUTER JOIN ticket ON ticket.seat_code = seat.code
GROUP BY (schedule.id)
HAVING COUNT(ticket.id)=0 AND schedule.planned_departure BETWEEN
NOW() and NOW() + INTERVAL '24 HOURS');
```

до:

	id [PK] bigint	planned_departure timestamp with time zone	factual_departure timestamp with time zone	factual_arrival timestamp with time zone	route_id bigint
1	1	2023-11-14 08:13:00+03	[null]	[null]	1
2	2	2023-11-11 20:13:00+03	[null]	[null]	1
3	3	2023-11-13 20:13:00+03	[null]	[null]	1
4	4	2023-11-12 09:10:00+03	2023-11-11 09:10:00+03	2023-11-11 13:09:00+03	2
5	5	2023-11-11 09:10:00+03	[null]	[null]	2
6	6	2023-11-13 09:10:00+03	[null]	[null]	2
7	7	2023-11-14 20:13:00+03	[null]	[null]	1

после:

	id [PK] bigint	planned_departure timestamp with time zone	factual_departure timestamp with time zone	factual_arrival timestamp with time zone	route_id bigint
1	1	2023-11-14 08:13:00+03	[null]	[null]	1
2	2	2023-11-11 20:13:00+03	[null]	[null]	1
3	4	2023-11-12 09:10:00+03	2023-11-11 09:10:00+03	2023-11-11 13:09:00+03	2
4	5	2023-11-11 09:10:00+03	[null]	[null]	2
5	6	2023-11-13 09:10:00+03	[null]	[null]	2
6	7	2023-11-14 20:13:00+03	[null]	[null]	1

3. INSERT (добавить резервное место для кондуктора в вагон, который отправляется в течение 24 часов)

```
INSERT INTO seat("code", "number", "type", "carriage_id", "manufacturer") VALUES
('РЕЗЕРВ', 99,
999,
(SELECT MAX(t.id) FROM (SELECT carriage.id FROM schedule
INNER JOIN carriage ON schedule.id = carriage.schedule_id
GROUP BY (carriage.id, schedule.planned_departure)
HAVING schedule.planned_departure BETWEEN NOW() AND NOW() + INTERVAL '24
HOURS') as t),
'Росстрой'
)
```

	code [PK] character varying	number integer	type smallint	carriage_id bigint	manufacturer character varying
5	КЕК-10	14	2	8	ООО Рога и Копыта
6	КЕК-11	15	2	8	ООО Рога и Копыта
7	КЕК-7	11	2	8	ООО Рога и Копыта
8	КЕК-8	12	2	8	ООО Рога и Копыта
9	КЕК-9	13	2	8	ООО Рога и Копыта
10	АБУ-2	19	3	9	АБУ
11	КЕК-5	8	2	5	ООО Рога и Копыта
12	КЕК-6	9	2	8	ООО Рога и Копыта
13	РЕЗЕРВ	99	999	4	Росстрой
14	СБУ-30	6	1	6	Укрмашстрой
15	СБУ-31	7	1	6	Укрмашстрой

#### 4. Индексы

Создадим индекс на даты в контракте и на айди машины:

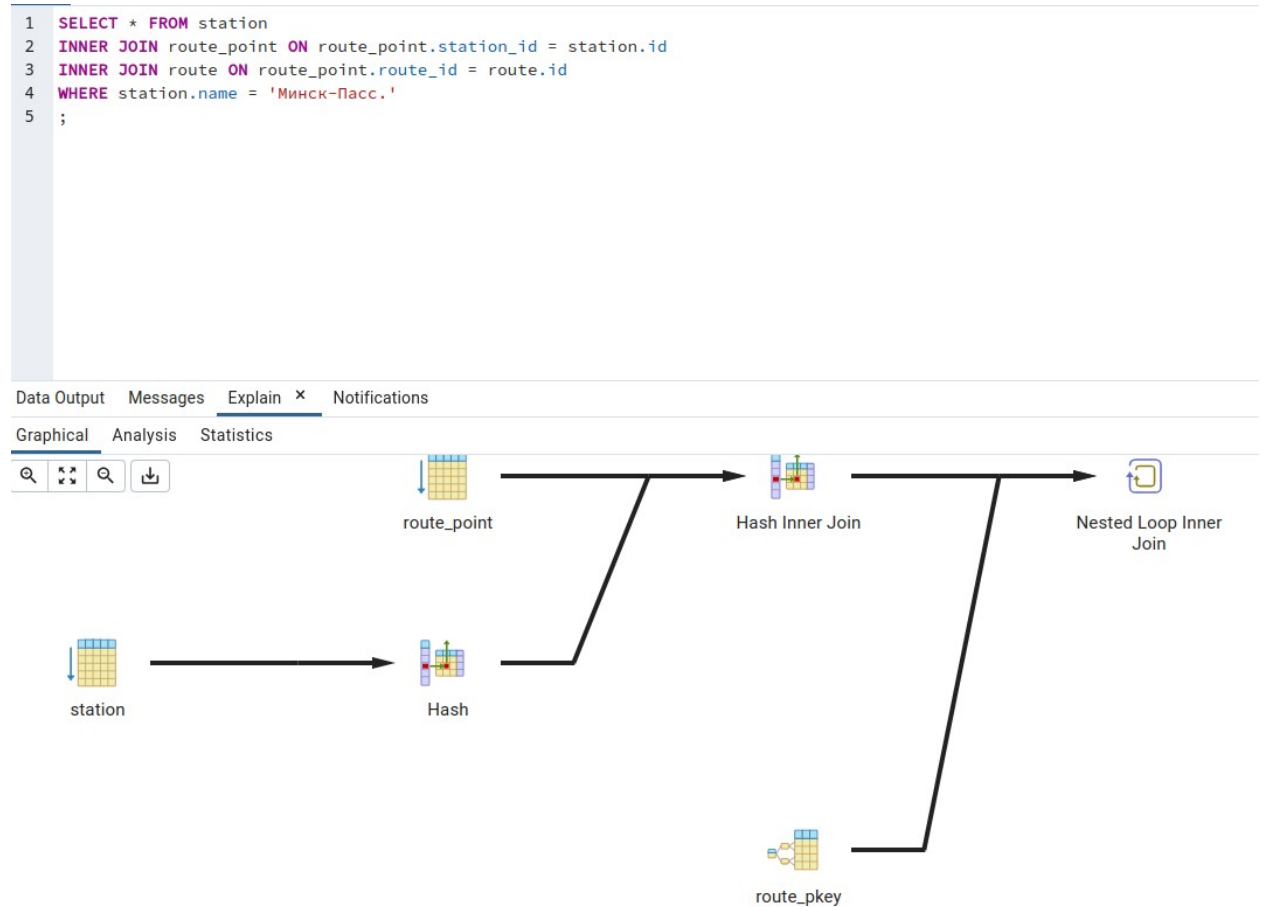
```
CREATE INDEX ON passenger (fio,id);
CREATE INDEX ON station USING HASH(name);
```



Выполнение :

```
EXPLAIN SELECT * FROM station
INNER JOIN route_point ON route_point.station_id = station.id
INNER JOIN route ON route_point.route_id = route.id
WHERE station.name = 'Минск-Пасс.'
;
```

С индексом:



```
EXPLAIN SELECT * FROM passenger WHERE passenger.fio = 'Гоев
Александр Прогреевич';
```

Без индекса:

1 EXPLAIN SELECT \* FROM station WHERE station.name = 'Минск-Пасс.';

Data Output Messages Notifications



QUERY PLAN	
text	
Seq Scan on station	(cost=0.00..11.75 rows=1 width=52..)
Filter: ((name)::text = 'Минск-Пасс. '::text)	

✓ Successfully run. Total query runtime: 122 msec. 2 rows affected. ✕

### С индексом:

1 EXPLAIN SELECT \* FROM station WHERE station.name = 'Минск-Пасс.';

Data Output Messages Notifications



QUERY PLAN	
text	
1 Seq Scan on station	(cost=0.00..1.24 rows=1 width=52..)
2 Filter: ((name)::text = 'Минск-Пасс. '::text)	

✓ Successfully run. Total query runtime: 55 msec. 2 rows affected. ✕

```
1 SELECT * FROM passenger WHERE passenger.fio = 'Гоев Александр Прогреевич';
```

Data Output

Messages





Explain ×


Notifications

Graphical

Analysis

Statistics



  
passenger

## Вывод

В результате выполнения лабораторной работы были созданы различные SELECT запросы для извлечения данных из бд. Кроме того, были созданы представления, хранимые процедуры. В результате выполнения лабораторной работы было установлено, что создание индексов привело к улучшению производительности запросов, сократив время выполнения в среднем на 50% и уменьшив нагрузку на базу данных. Таким образом, использование индексов является эффективным способом оптимизации базы данных и улучшения производительности запросов.