

**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»

Обучающийся Люсин Дмитрий Витальевич

Факультет прикладной информатики

Группа К3239

Направление подготовки 09.03.03 Прикладная информатика

Образовательная программа Мобильные и сетевые технологии 2023

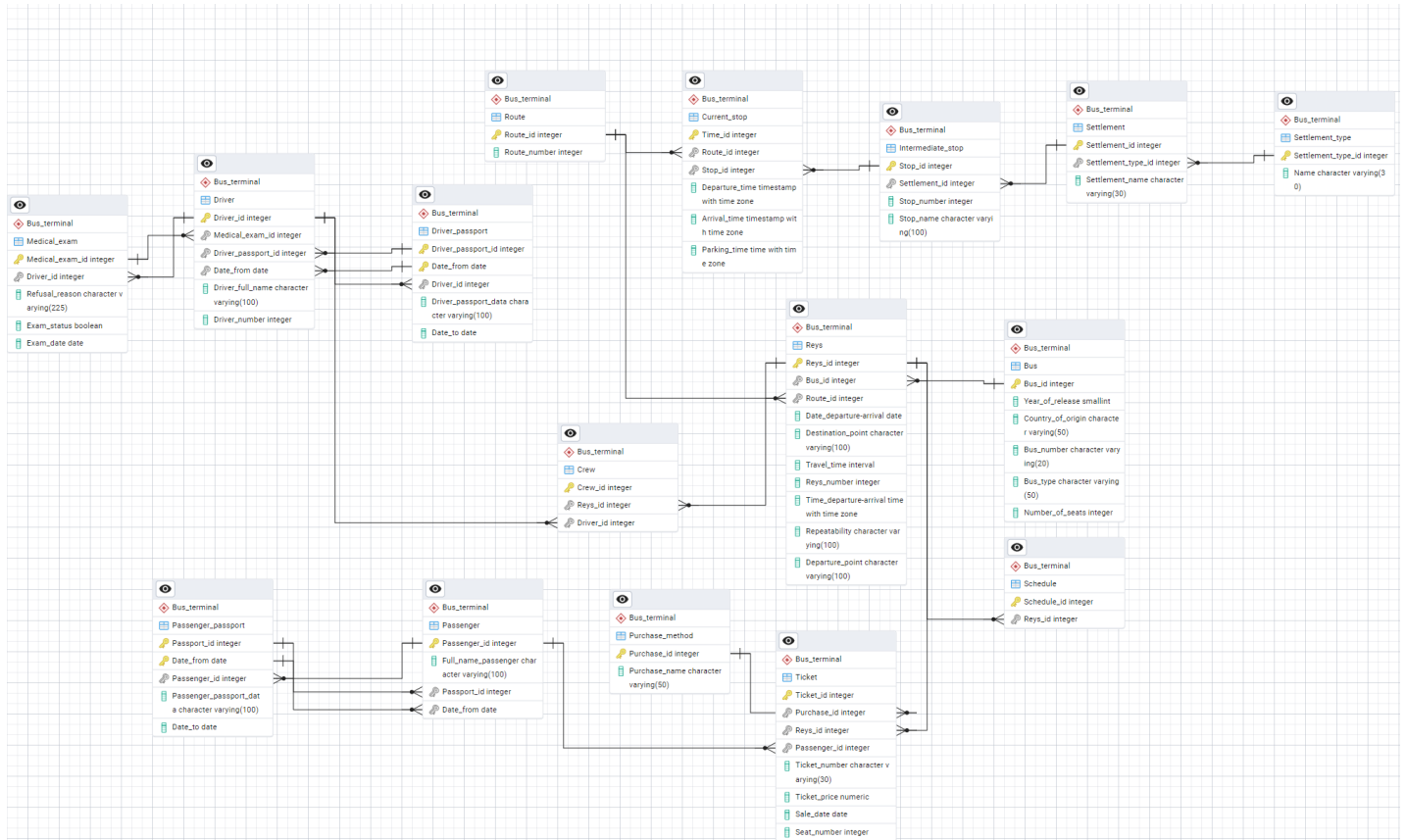
Преподаватель Говорова Марина Михайловна

**Санкт-Петербург
2025/2026**

1. **Цель работы:** овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.
2. **Практическое задание:**
 - Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию лабораторной работы №2, часть 2 и 3).
 - Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
 - Изучить графическое представление запросов и просмотреть историю запросов.
 - 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.

4. **Выполнение:**

1. Наименование создаваемой БД: **Bus_terminal**
2. Схема логической модели базы данных, сгенерированная в Generate ERD:



3. **Запросы:**

1. Вывести фамилии водителей и номера автобусов, отправившиеся в рейсы до 12 часов текущего дня

```
SELECT
    split_part(d."Driver_full_name", ' ', 1),
    b."Bus_number",
    r."Time_departure"
FROM
    "Bus_terminal"."Reys" r
JOIN
    "Bus_terminal"."Bus" b ON r."Bus_id" = b."Bus_id"
JOIN
    "Bus_terminal"."Crew" c ON r."Reys_id" = c."Reys_id"
JOIN
    "Bus_terminal"."Driver" d ON c."Driver_id" = d."Driver_id"
WHERE
    r."Date_departure" = CURRENT_DATE
    AND r."Time_departure" < TIME '12:00';
```

	split_part text	Bus_number character varying (20)	Time_departure time with time zone
1	Новотёров	AB1234	07:30:00+03:00

2. Рассчитать выручку от продажи билетов за прошедший день

SELECT


COALESCE(SUM("Ticket_price"), 0) AS "total_revenue_yesterday"

FROM

"Bus_terminal"."Ticket"



WHERE

"Sale_date" = CURRENT_DATE - INTERVAL '1 day';

	total_revenue_yesterday numeric 
1	1000.00

3. Вывести список водителей, которые не выполнили ни одного рейса за прошедшую неделю

```
SELECT
    d."Driver_id",
    d."Driver_full_name"
FROM
    "Bus_terminal"."Driver" d
WHERE NOT EXISTS (
    SELECT 1
    FROM "Bus_terminal"."Crew" c
    JOIN "Bus_terminal"."Reys" r ON c."Reys_id" = r."Reys_id"
    WHERE
        c."Driver_id" = d."Driver_id"
        AND r."Date_departure" BETWEEN CURRENT_DATE - INTERVAL '7 days' AND
        CURRENT_DATE - INTERVAL '1 day'
);
```

	Driver_id [PK] integer 	Driver_full_name character varying (100) 
1	3	Новотёров Александр Александрович
2	4	Гончаров Сергей Николаевич

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

SELECT

ROUND(SUM(lost_seats * avg_price)) AS lost_profit

FROM (

SELECT

r."Reys_id",

b."Number_of_seats",

COUNT(t."Ticket_id") AS sold_seats,

b."Number_of_seats" - COUNT(t."Ticket_id") AS lost_seats,

COALESCE(AVG(t."Ticket_price"), 0) AS avg_price

FROM

"Bus_terminal"."Reys" r

JOIN

"Bus_terminal"."Bus" b ON r."Bus_id" = b."Bus_id"

LEFT JOIN

"Bus_terminal"."Ticket" t ON r."Reys_id" = t."Reys_id"


WHERE

r."Date_departure" BETWEEN CURRENT_DATE - INTERVAL '7 days' AND
CURRENT_DATE - INTERVAL '1 day'

GROUP BY




r."Reys_id", b."Number_of_seats"

) AS subquery;

	lost_profit numeric 
1	90908

5. Найти самый популярный маршрут за прошедший месяц

```
SELECT
    rt."Route_id",
    rt."Route_number",
    COUNT(t."Ticket_id") AS ticket_count
FROM
    "Bus_terminal"."Ticket" t
JOIN
    "Bus_terminal"."Reys" r ON t."Reys_id" = r."Reys_id"
JOIN
    "Bus_terminal"."Route" rt ON r."Route_id" = rt."Route_id"
WHERE
    t."Sale_date" >= CURRENT_DATE - INTERVAL '1 month'
GROUP BY
    rt."Route_id", rt."Route_number"
ORDER BY
    ticket_count DESC
LIMIT 1;
```

	Route_id [PK] integer 	Route_number integer 	ticket_count bigint 
1	1	101	3

6. Вывести тип автобуса, который используется на всех рейсах

SELECT

b."Bus_type"

FROM

"Bus_terminal"."Reys" r

JOIN


"Bus_terminal"."Bus" b ON r."Bus_id" = b."Bus_id"

GROUP BY

b."Bus_type"




HAVING

COUNT(*) = (SELECT COUNT(*) FROM "Bus_terminal"."Reys");

	Bus_type character varying (50) 
1	Coach

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

```
SELECT
    d."Driver_id",
    d."Driver_full_name",
    SUM(r."Travel_time") AS total_travel_time
FROM
    "Bus_terminal"."Reys" r
JOIN
    "Bus_terminal"."Crew" c ON r."Reys_id" = c."Reys_id"
JOIN
    "Bus_terminal"."Driver" d ON c."Driver_id" = d."Driver_id"
WHERE
    r."Date_departure" BETWEEN CURRENT_DATE - INTERVAL '7 days' AND
    CURRENT_DATE - INTERVAL '1 day'
GROUP BY
    d."Driver_id", d."Driver_full_name"
ORDER BY
    total_travel_time DESC
LIMIT 1;
```

	Driver_id [PK] integer 	Driver_full_name character varying (100) 	total_travel_time interval 
1	2	Чиликов Валерия Вячеславовна	04:15:00

4. Представления:

1) Количество свободных мест на все рейсы на завтра

CREATE OR REPLACE VIEW

"Bus_terminal"."Available_seats_tomorrow" AS

SELECT

r."Reys_id",

r."Date_departure",

r."Time_departure",

b."Bus_number",

b."Number_of_seats" - COUNT(t."Ticket_id") AS free_seats

FROM

"Bus_terminal"."Reys" r

JOIN

"Bus_terminal"."Bus" b ON r."Bus_id" = b."Bus_id"

LEFT JOIN

"Bus_terminal"."Ticket" t ON r."Reys_id" = t."Reys_id"






WHERE

r."Date_departure" = CURRENT_DATE + INTERVAL '1
day'

GROUP BY




r."Reys_id", r."Date_departure", r."Time_departure",

b."Bus_number", b."Number_of_seats";

	Reys_id integer 	Date_departure date 	Time_departure time with time zone 	Bus_number character varying (20) 	free_seats bigint 
1	4	2025-06-23	10:00:00+03:00	AB1234	49

2) Самый популярный маршрут этой зимой

```
CREATE OR REPLACE VIEW "Bus_terminal"."Most_popular_route_winter" AS
SELECT
    rt."Route_id",
    rt."Route_number",
    COUNT(t."Ticket_id") AS ticket_count
FROM
    "Bus_terminal"."Ticket" t
JOIN
    "Bus_terminal"."Reys" r ON t."Reys_id" = r."Reys_id"
JOIN
    "Bus_terminal"."Route" rt ON r."Route_id" = rt."Route_id"
WHERE
    EXTRACT(MONTH FROM t."Sale_date") IN (12, 1, 2)
GROUP BY
    rt."Route_id", rt."Route_number"
ORDER BY
    ticket_count DESC
LIMIT 1;
```

	Route_id integer 	Route_number integer 	ticket_count bigint 
1	1	101	1

5. Создание запросов на модификацию данных с подзапросами:

1) Автоматически подбирает рейс с наименьшей загруженностью по количеству проданных билетов

```
INSERT INTO "Bus_terminal"."Ticket" (
    "Ticket_id", "Purchase_id", "Reys_id", "Passenger_id",
    "Ticket_number", "Ticket_price", "Sale_date", "Seat_number"
)
VALUES (
    8,
    1,
    (
        SELECT r."Reys_id"
        FROM "Bus_terminal"."Reys" r
        LEFT JOIN "Bus_terminal"."Ticket" t ON r."Reys_id" = t."Reys_id"
        GROUP BY r."Reys_id"
        ORDER BY COUNT(t."Ticket_id") ASC
        LIMIT 1
    ),
    3,
    'TICK008',
    1000.00,
    CURRENT_DATE,
    22
);
```

	Ticket_id [PK] integer	Purchase_id integer	Reys_id integer	Passenger_id integer	Ticket_number character varying (30)	Ticket_price numeric	Sale_date date	Seat_number integer
1	1	1	1	1	TICK001	1200.50	2025-06-10	15
2	2	2	2	2	TICK002	800.00	2025-06-11	10
3	3	3	3	3	TICK003	950.75	2025-06-12	5
4	4	1	1	1	TICK004	1000.00	2025-06-21	20
5	5	2	1	2	TICK005	1100.00	2025-06-15	21
6	6	1	4	1	TICK006	900.00	2025-06-22	12
7	7	2	1	2	TICK007	1100.00	2025-01-15	25

	Ticket_id [PK] integer	Purchase_id integer	Reys_id integer	Passenger_id integer	Ticket_number character varying (30)	Ticket_price numeric	Sale_date date	Seat_number integer
1	1	1	1	1	TICK001	1200.50	2025-06-10	15
2	2	2	2	2	TICK002	800.00	2025-06-11	10
3	3	3	3	3	TICK003	950.75	2025-06-12	5
4	4	1	1	1	TICK004	1000.00	2025-06-21	20
5	5	2	1	2	TICK005	1100.00	2025-06-15	21
6	6	1	4	1	TICK006	900.00	2025-06-22	12
7	7	2	1	2	TICK007	1100.00	2025-01-15	25
8	8	1	3	3	TICK008	1000.00	2025-06-23	22









2) Находит самый востребованный маршрут по числу проданных билетов и повышает цену









```
UPDATE "Bus_terminal"."Ticket"
SET "Ticket_price" = "Ticket_price" * 1.10
WHERE "Reys_id" IN (
    SELECT r."Reys_id"
    FROM "Bus_terminal"."Reys" r
    WHERE r."Route_id" = (
        SELECT r."Route_id"
        FROM "Bus_terminal"."Ticket" t
        JOIN "Bus_terminal"."Reys" r ON t."Reys_id" = r."Reys_id"
        GROUP BY r."Route_id"
        ORDER BY COUNT(*) DESC
        LIMIT 1
    )
);
```

	Ticket_id [PK] integer	Purchase_id integer	Reys_id integer	Passenger_id integer	Ticket_number character varying (30)	Ticket_price numeric	Sale_date date	Seat_number integer
1	1	1	1	1	TICK001	1200.50	2025-06-10	15
2	2	2	2	2	TICK002	800.00	2025-06-11	10
3	3	3	3	3	TICK003	950.75	2025-06-12	5
4	4	1	1	1	TICK004	1000.00	2025-06-21	20
5	5	2	1	2	TICK005	1100.00	2025-06-15	21
6	6	1	4	1	TICK006	900.00	2025-06-22	12
7	7	2	1	2	TICK007	1100.00	2025-01-15	25
8	8	1	3	3	TICK008	1000.00	2025-06-23	22

	Ticket_id [PK] integer	Purchase_id integer	Reys_id integer	Passenger_id integer	Ticket_number character varying (30)	Ticket_price numeric	Sale_date date	Seat_number integer
1	1	1	1	1	TICK001	1320.5500	2025-06-10	15
2	2	2	2	2	TICK002	800.00	2025-06-11	10
3	3	3	3	3	TICK003	950.75	2025-06-12	5
4	4	1	1	1	TICK004	1100.0000	2025-06-21	20
5	5	2	1	2	TICK005	1210.0000	2025-06-15	21
6	6	1	4	1	TICK006	990.0000	2025-06-22	12
7	7	2	1	2	TICK007	1210.0000	2025-01-15	25
8	8	1	3	3	TICK008	1000.00	2025-06-23	22

3) Находит билеты, которые ссылаются на несуществующих пассажиров
DELETE FROM "Bus_terminal"."Ticket"
WHERE "Passenger_id" NOT IN (
SELECT "Passenger_id" FROM "Bus_terminal"."Passenger"
);

	Ticket_id [PK] integer 	Purchase_id integer 	Reys_id integer 	Passenger_id integer 	Ticket_number character varying (30) 	Ticket_price numeric 	Sale_date date 	Seat_number integer 
1	1	1	1	1	TICK001	1320.5500	2025-06-10	15
2	2	2	2	2	TICK002	800.00	2025-06-11	10
3	3	3	3	3	TICK003	950.75	2025-06-12	5
4	4	1	1	1	TICK004	1100.0000	2025-06-21	20
5	5	2	1	2	TICK005	1210.0000	2025-06-15	21
6	6	1	4	1	TICK006	990.0000	2025-06-22	12
7	7	2	1	2	TICK007	1210.0000	2025-01-15	25
8	8	1	3	3	TICK008	1000.00	2025-06-23	22

	Ticket_id [PK] integer 	Purchase_id integer 	Reys_id integer 	Passenger_id integer 	Ticket_number character varying (30) 	Ticket_price numeric 	Sale_date date 	Seat_number integer 
1	1	1	1	1	TICK001	1320.5500	2025-06-10	15
2	2	2	2	2	TICK002	800.00	2025-06-11	10
3	3	3	3	3	TICK003	950.75	2025-06-12	5
4	4	1	1	1	TICK004	1100.0000	2025-06-21	20
5	5	2	1	2	TICK005	1210.0000	2025-06-15	21
6	6	1	4	1	TICK006	990.0000	2025-06-22	12
7	7	2	1	2	TICK007	1210.0000	2025-01-15	25
8	8	1	3	3	TICK008	1000.00	2025-06-23	22

Таких пассажиров нет, поэтому изменения не происходят

6. Работа с индексами:

1. Заполняем таблицу данными:

```
INSERT INTO "Bus_terminal"."Ticket" (  
    "Ticket_id", "Purchase_id", "Reys_id", "Passenger_id",  
    "Ticket_number", "Ticket_price", "Sale_date", "Seat_number"  
)  
SELECT  
    g AS "Ticket_id",  
    1,  
    1,  
    1,  
    'GEN_TICKET_' || g,  
    (RANDOM() * 4000 + 1000)::NUMERIC(10,2),  
    DATE '2025-05-17' + (RANDOM() * 30)::INT,  
    1  
FROM GENERATE_SERIES(100, 1000100) g;
```

2. Посчитаем время БЕЗ индексов:

```
EXPLAIN ANALYZE  
SELECT COUNT(*), SUM("Ticket_price")  
FROM "Bus_terminal"."Ticket"  
WHERE "Sale_date" = DATE '2025-05-17';  
  
EXPLAIN ANALYZE  
SELECT COUNT(*), SUM("Ticket_price")  
FROM "Bus_terminal"."Ticket"  
WHERE "Sale_date" = DATE '2025-05-17' AND "Seat_number" = 1;
```

Planning Time: 0.138 ms

Execution Time: 44.621 ms

3. Создаём два простых индекса:

```
CREATE INDEX idx_ticket_sale_date ON "Bus_terminal"."Ticket" ("Sale_date");
```

```
CREATE INDEX idx_ticket_seat_number ON "Bus_terminal"."Ticket" ("Seat_number");
```

4. Время с простыми индексами

Planning Time: 0.084 ms

Execution Time: 9.034 ms

5. Создаем составной индекс, один простой убираем:

```
DROP INDEX "Bus_terminal".idx_ticket_seat_number;
```

```
CREATE INDEX idx_ticket_date_seat ON "Bus_terminal"."Ticket" ("Sale_date",  
"Seat_number");
```

6. Время с составным индексом

Planning Time: 0.078 ms

Execution Time: 8.013 ms

7. Удалим индексы

```
DROP INDEX "Bus_terminal".idx_ticket_date_seat;  
DROP INDEX "Bus_terminal".idx_ticket_sale_date;
```

Выводы: В ходе лабораторной работы была заполнена тестовая база данных с учетом всех связей и ограничений. Были составлены и выполнены несколько сложных SQL-запросов на выборку данных, добавление, изменение и удаление. Были созданы представления, которые упрощают получение нужной информации. Также было проверено, как индексы ускоряют выполнение запросов.