**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: Даньшин С. А.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

**Оглавление**

[1. Запросы к базе данных 4](#_Toc150470282)

[2. Представления 8](#_Toc150470283)

[3. Кастом запросы 9](#_Toc150470284)

[4. Индексы 12](#_Toc150470285)

[Вывод 13](#_Toc150470286)

**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.

# Запросы к базе данных

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

SELECT drivers.last\_name, buses.registration\_number

FROM trips

JOIN crews ON trips.id = crews.trip\_id

JOIN drivers ON crews.driver\_id = drivers.id

JOIN buses ON trips.bus\_id = buses.id

WHERE cast(trips.start\_time as time) < '12:00:00' and cast(trips.start\_time as date) = current\_date;

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

* Рассчитать выручку от продажи билетов за прошедший день.

SELECT SUM(tr.price) AS total\_revenue

FROM tickets t

JOIN trips tr ON t.trip\_id = tr.id

where cast(t.sold\_at as date) = current\_date - interval 'day 1'

and t.status=’PAID’;

Изображение выглядит как текст, снимок экрана, Шрифт, Мультимедийное программное обеспечение

Автоматически созданное описание

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

select \* from drivers

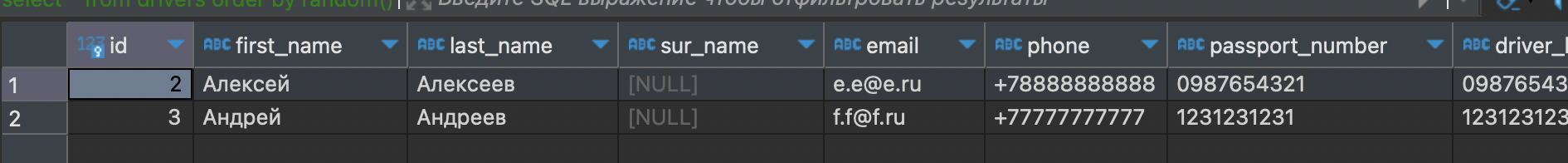
where drivers.id not in (

select d.id from trips t

join crews c on c.trip\_id = t.id

join drivers d on c.driver\_id = d.id

where cast(t.start\_time as date) = current\_date - interval '1 day';



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

select

tr.id,

(tr.number\_of\_seats - COUNT(t.id)) \* tr.price as lost\_profit

from tickets t

join (

select

trips.id as id,

trips.start\_time,

bm.number\_of\_seats as number\_of\_seats,

trips.price

from trips

join buses b on trips.bus\_id = b.id

join bus\_models bm on bm.id = b.model\_id

) tr on t.trip\_id = tr.id

where cast(tr.start\_time as date) > current\_date - interval '7 day'

group by tr.id, tr.number\_of\_seats, tr.price;

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

* Сколько рейсов выполнил каждый водитель за последний месяц.

**select**

d.last\_name,

d.first\_name,

d.sur\_name,

tmp.trip\_count **as** trip\_count

**from** drivers d

**inner** **join** (**select** c.driver\_id **as** driver\_id, **count**(tr.id) **as** trip\_count

**from** trips tr

**join** crews c **on** tr.id = c.trip\_id

**group** **by** c.driver\_id

) tmp **on** d.id=tmp.driver\_id;

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

* Вывести тип автобуса, который используется на всех рейсах.

**select**

bus\_model

**from** (

**select**

bm.id **as** bus\_model\_id,

bm.brand || ' ' || bm.model\_name **AS** bus\_model,

**count**(**distinct** r.id) **as** rout\_count

**from** routes r

**join** trips t **on** r.id=t.route\_id

**JOIN** buses b **ON** t.bus\_id=b.id

**JOIN** bus\_models bm **ON** bm.id=b.model\_id

**group** **by** bm.id, bus\_model

) **as** br

**where** rout\_count=(**select** **count**(id) **from** routes r2);

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

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

S **select**

d.last\_name,

d.first\_name,

d.sur\_name

**from** drivers d

**join** (

**SELECT** driver\_id

**FROM** (

**SELECT** ddt.driver\_id, **sum**(ddt.driving\_time) **as** total\_driving\_time

**FROM** (

**SELECT** d.id **as** driver\_id, tr.id **as** trip\_id, tr.end\_time - tr.start\_time **as** driving\_time

**FROM** trips tr

**JOIN** crews c **ON** tr.id = c.trip\_id

**JOIN** drivers d **ON** c.driver\_id = d.id

**WHERE** **cast**(tr.start\_time **as** **date**) > **current\_date** - **interval** '7 day'

) **as** ddt

**GROUP** **BY** ddt.driver\_id

) mt

**WHERE** total\_driving\_time = (

**SELECT** **MAX**(total\_driving\_time)

**FROM** (

**SELECT** ddt.driver\_id, **sum**(ddt.driving\_time) **as** total\_driving\_time

**FROM** (

**SELECT** d.id **as** driver\_id, tr.id **as** trip\_id, tr.end\_time - tr.start\_time **as** driving\_time

**FROM** trips tr

**JOIN** crews c **ON** tr.id = c.trip\_id

**JOIN** drivers d **ON** c.driver\_id = d.id

**WHERE** **cast**(tr.start\_time **as** **date**) > **current\_date** - **interval** '7 day'

) **as** ddt

**GROUP** **BY** ddt.driver\_id

) sub

)

) tmp **on** tmp.driver\_id=d.id;

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

# Представления

* Количество свободных мест на все рейсы на завтра:

CREATE VIEW available\_seats\_tomorrow AS

SELECT

t.id AS trip\_id,

r.name AS route\_name,

bm.brand || '' || bm.model\_name AS bus\_model,

b.registration\_number AS bus\_number,

COUNT(tickets.id) AS occupied\_seats, bm.number\_of\_seats AS total\_seats,

bm.number\_of\_seats - COUNT(tickets.id) AS available\_seats

FROM trips t

JOIN buses b ON t.bus\_id = b.id

JOIN bus\_models bm ON b.model\_id = bm.id

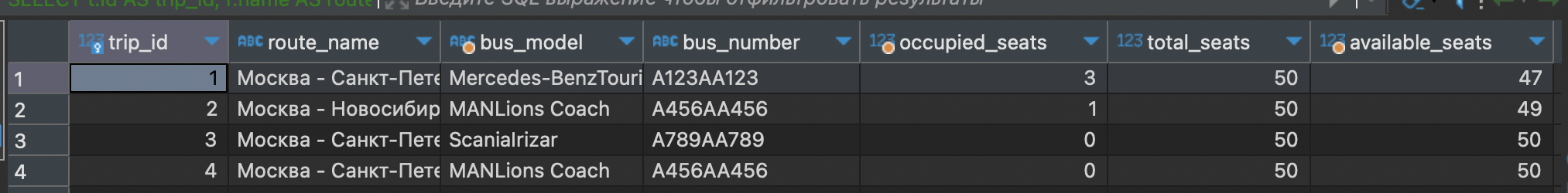
JOIN routes r ON t.route\_id = r.id

LEFT JOIN tickets ON t.id = tickets.trip\_id

WHERE t.start\_time >= CURRENT\_DATE + INTERVAL '1 day'

AND t.start\_time < CURRENT\_DATE + INTERVAL '2 days'

GROUP BY t.id, r.name, bm.brand, bm.model\_name, b.registration\_number, bm.number\_of\_seats;



* Самый популярный маршрут этой зимой:

CREATE VIEW most\_popular\_routes\_winter AS

SELECT route\_id, route\_name, total\_tickets\_sold

FROM (

SELECT r.id AS route\_id, r.name AS route\_name,

COUNT(tickets.id) AS total\_tickets\_sold

FROM routes r

JOIN trips t ON r.id = t.route\_id

JOIN tickets ON t.id = tickets.trip\_id

WHERE EXTRACT(MONTH FROM t.start\_time) IN (12, 1, 2)

GROUP BY r.id, r.name

) as route\_sales

WHERE total\_tickets\_sold = (

SELECT MAX(total\_tickets\_sold)

FROM (

SELECT COUNT(tickets.id) AS total\_tickets\_sold

FROM routes r

JOIN trips t ON r.id = t.route\_id

JOIN tickets ON t.id = tickets.trip\_id

WHERE EXTRACT(MONTH FROM t.start\_time) IN (12, 1, 2)

GROUP BY r.id, r.name

) as max\_sales

);

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

# Запросы на модификацию (с подзапросами)

* Исправить опечатку в названии модели автобуса

UPDATE bus\_models

SET model\_name = 'Lions Coach'

WHERE id = (SELECT id FROM bus\_models WHERE brand = ‘MAN’ AND model\_name = ‘Lionscroach’');

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

* Удалить пассажиров которые не ездили у нас больше 2х лет

DELETE FROM passengers

WHERE id IN (

SELECT p.id

FROM passengers p

JOIN tickets t ON p.id = t.passenger\_id

JOIN trips tr ON t.trip\_id = tr.id

WHERE tr.end\_time < NOW() - INTERVAL '2 years'

GROUP BY p.id

HAVING MAX(tr.end\_time) < NOW() - INTERVAL '2 years'

);

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описаниеИзображение выглядит как снимок экрана, Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

* Добавление билета:

INSERT INTO tickets (passenger\_id, trip\_id, seat\_number, status, start\_station\_id, end\_station\_id, is\_online\_sale)

VALUES (

(SELECT id FROM passengers WHERE email = 'a.a@a.ru'),

(SELECT id FROM trips WHERE route\_id = 1 AND status=”PLANNED” ORDER BY start\_time),

1,

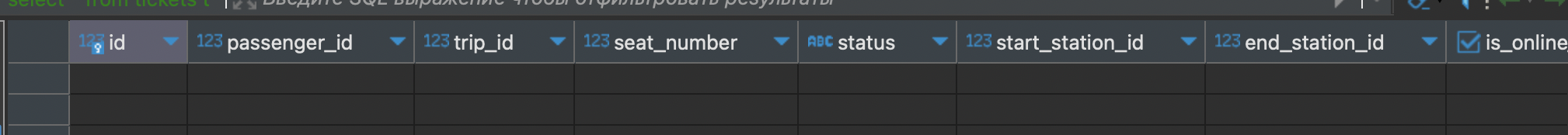
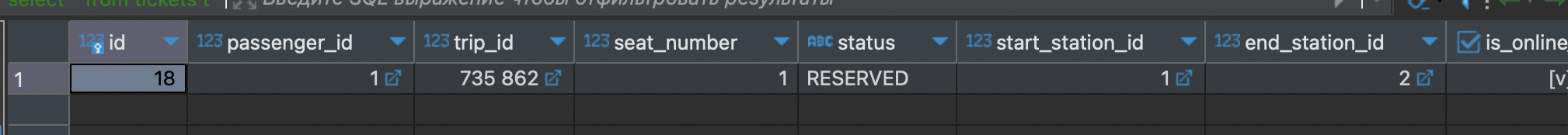
'RESERVED',

1,

2,

true

);

# Индексы

Создадим индекс на время отправления завершения поездки как отдельно, так и вместе:

CREATE INDEX idx\_trip\_start\_time on trips(start\_time);

CREATE INDEX idx\_trip\_end\_time on trips(end\_time);

CREATE INDEX idx\_trip\_start\_and\_end\_time on trips(start\_date, end\_time);

Тестовый запрос:

select

t.id as trip\_id,

r.name as route\_name,

bm.brand || ' ' || bm.model\_name as bus\_model,

b.registration\_number as bus\_number,

bm.number\_of\_seats - COUNT(t2.id) as available\_seats

from

trips t

join buses b on

t.bus\_id = b.id

join bus\_models bm on

b.model\_id = bm.id

join routes r on

t.route\_id = r.id

left join tickets t2 on

t.id = t2.trip\_id

where

cast(t.start\_time as date) = '2021-01-01'

and cast(t.end\_time as date) = '2021-01-01'

group by

t.id,

r.name,

bm.brand,

bm.model\_name,

b.registration\_number,

bm.number\_of\_seats;

Без индекса:

Planning Time: 4.879 ms

Execution Time: 128.666 ms

Индекс стартового времени:

Planning Time: 3.667 ms

Execution Time: 56.838 ms

Индекс конечного времени:

Planning Time: 1.479 ms

Execution Time: 42.334 ms

Индекс на оба поля:

Planning Time: 0.729 ms

Execution Time: 44.153 ms

# Вывод

В ходе лабораторной работы я освоил практические навыки по выполнению select, insert, delete и update запросов. Также, научился делать представления и индексы. В ходе анализа в первом запросе индексы позволили сократить время выполнения запроса на ~40%.