Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

По Лабораторной работе 2 Запросы на выборку и модификацию данных, представления и индексы в **POSTGRESQL**

по дисциплине «Проектирование и реализация баз данных»

Автор:Казанков И Факультет:ИКТ

Группа: К32402

Преподаватель: Говорова М. М.

Санкт-Петербург 2023

Цель работы: овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

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 посмотреть 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:

VALUES (1, 'CB', 1), (1, 'CB', 2),

(2, 'купейный', 1),

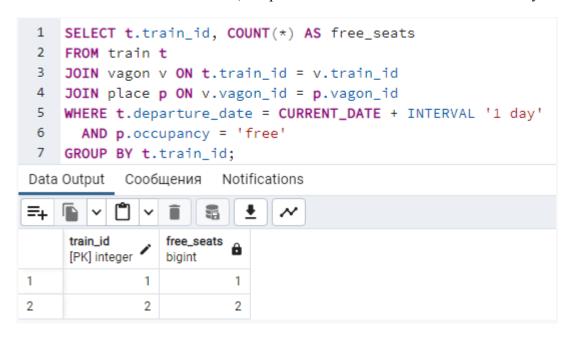
```
Заполнение бд:
INSERT INTO regular_schedule (arrival_time, departure_time, train_arrival_point, train_departure_point)
VALUES
('08:00', '08:30', 'Station A', 'Москва'),
('10:15', '10:45', 'Station C', 'Москва'),
('12:30', '13:00', 'Station E', 'Москва');
INSERT INTO train (train_name, train_number, fact_time_sent, fact_time_transformation, arrival_date,
departure_date, type_train)
VALUES
('Train A', 1, '07:45', '08:15', '2023-06-29', '2023-06-29', 'Express'),
('Train B', 2, '10:00', '10:30', '2023-06-28', '2023-06-29', 'Local'),
('Train C', 3, '12:15', '12:45', '2023-06-29', '2023-06-29', 'Express');
INSERT INTO stopover (train_number, type_locality, parking_time, type_stop)
VALUES
(1, 'Irkutsk', '10 minutes', 'Departure'),
(2, 'Krasnovarsk', '5 minutes', 'Arrival'),
(3, 'Omsk', '15 minutes', 'Departure');
INSERT INTO station (name_station)
VALUES
('Station A'),
('Station B'),
('Station C'),
('Москва');
```

INSERT INTO vagon (train_id, vagon_type, vagon_number_in_train)

```
(2, 'купейный', 2),
(3, 'CB', 1);
INSERT INTO place (vagon_id, number_of_seats, occupancy)
(1, 50, 'Occupied'),
(1, 50, 'free'),
(2, 40, 'Occupied'),
(2, 40, 'Occupied'),
(3, 30, 'free'),
(3, 20, 'free');
INSERT INTO passenger (passport, full_name)
VALUES
('ABC123', 'Ivan'),
('DEF456', 'Ilya'),
('GHI789', 'Sergey');
INSERT INTO cashbox (locality, address_cashbox)
VALUES
('City X', '123 Main Street'),
('City Y', '456 Park Avenue');
INSERT INTO ticket (place_id, stopover_id, cashbox_id, passenger_id, payment_method, date_of_purchase,
price_ticket, arrival_point, departure_point)
VALUES
(1, 1, 1, 1, 'Cash', '2023-06-28', 10, 'Station A', 'Москва'),
(2, 2, 2, 2, 'Credit Card', '2023-06-28', 15, 'Station A', 'Mocква'),
(3, 3, 1, 3, 'Cash', '2023-06-27', 20, 'Station A', 'Mocква');
```

1: Запросы:

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



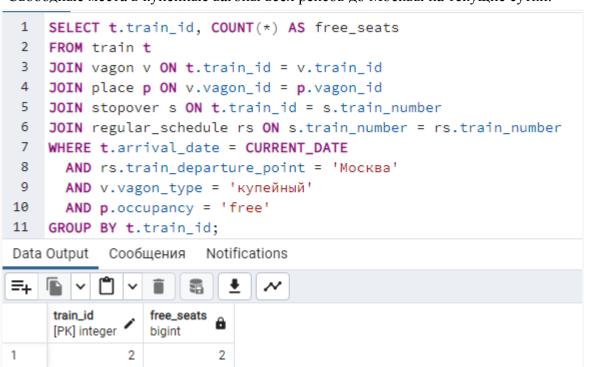
Список пассажиров, отправившихся в Москву всеми рейсами за прошедшие сутки.

```
1
    SELECT pa.full_name
 2
    FROM passenger pa
 3
    JOIN ticket ti ON pa.passenger_id = ti.passenger_id
    JOIN stopover s ON ti.stopover_id = s.stopover_id
 5
    JOIN regular_schedule rs ON s.train_number = rs.train_number
    WHERE ti.date_of_purchase >= CURRENT_DATE - INTERVAL '1 day'
 6
 7
      AND ti.date_of_purchase < CURRENT_DATE
      AND ti.departure_point = 'MockBa';
Data Output Сообщения Notifications
=+ |
     full_name
     character varying (40)
1
     Sergey
```

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

```
SELECT rs.train number
FROM regular_schedule rs
INNER JOIN train t ON rs.train_number = t.train_number
WHERE t.departure_date = CURRENT_DATE + INTERVAL '1 day'
GROUP BY rs.train_number
HAVING COUNT(DISTINCT t.train_id) = (
  SELECT COUNT(*)
  FROM vagon v
  JOIN place p ON v.vagon_id = p.vagon_id
  WHERE v.train_id = rs.train_number
);
       Сообщения
                   Notifications
Output
train_number
[PK] integer
```

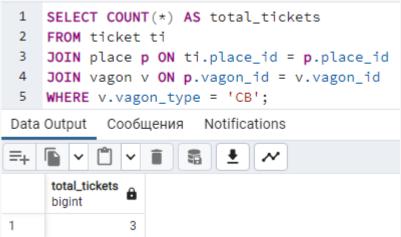
Свободные места в купейные вагоны всех рейсов до Москвы на текущие сутки.



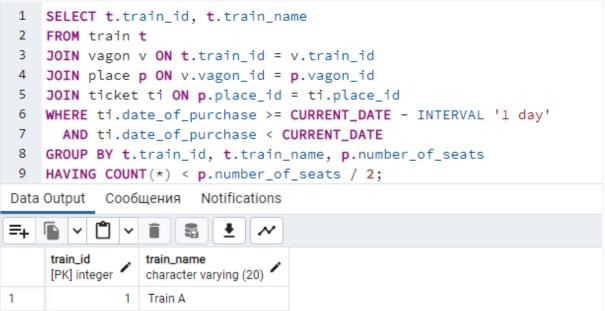
Выручка от продажи билетов на все поезда за прошедшие сутки.

```
SELECT t.train_id, SUM(ti.price_ticket) AS total_revenue
 1
 2
    FROM train t
    JOIN vagon v ON t.train_id = v.train_id
 3
    JOIN place p ON v.vagon_id = p.vagon_id
 4
 5
    JOIN ticket ti ON p.place_id = ti.place_id
   WHERE ti.date_of_purchase >= CURRENT_DATE - INTERVAL '1 day'
 6
 7
      AND ti.date_of_purchase < CURRENT_DATE
    GROUP BY t.train_id;
 8
Data Output
           Сообщения
                        Notifications
=+
     train_id
                 total_revenue
     [PK] integer
                 bigint
1
                           20
```

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.



представление:

для пассажиров о наличии свободных мест на заданный рейс:

```
1 CREATE VIEW free_seats AS
2 SELECT p.place_id, v.vagon_type, v.vagon_number_in_train, p.number_of_seats
3 FROM place p
4 INNER JOIN vagon v ON p.vagon_id = v.vagon_id
5 WHERE p.occupancy = 'free';

Data Output Сообщения Notifications

CREATE VIEW

Запрос завершён успешно, время выполнения: 418 msec.
```

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

```
CREATE INDEX idx_train_departure_date ON train (departure_date);
CREATE INDEX idx_vagon_train_id ON vagon (train_id);
CREATE INDEX idx_place_vagon_id_occupancy ON place (vagon_id, occupancy);
EXPLAIN SELECT t.train_id, COUNT(*) AS free_seats
FROM train t
JOIN vagon v ON t.train_id = v.train_id
JOIN place p ON v.vagon_id = p.vagon_id
WHERE t.departure_date = CURRENT_DATE + INTERVAL '1 day'
AND p.occupancy = 'free'
GROUP BY t.train_id;
```

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

INSERT:

```
1 INSERT INTO passenger (passport, full_name)
2 SELECT '1234567890', 'Ilya Kazankov'
3 WHERE NOT EXISTS (SELECT 1 FROM passenger WHERE passport = '1234567890');
Data Output Сообщения Notifications
```

INSERT 0 1

Запрос завершён успешно, время выполнения: 35 msec.

	passenger_id [PK] integer	passport character varying (10)	full_name character varying (40)
1	1	ABC123	Ivan
2	2	DEF456	Ilya
3	3	GHI789	Sergey
4	4	1234567890	Ilya Kazankov

UPDATE:

```
UPDATE place
 1
    SET occupancy = 'Occupied'
 2
    WHERE vagon_id = (
 3
      SELECT vagon_id
 4
      FROM vagon
 5
      WHERE vagon_type = 'купейный'
 6
      LIMIT 1
 7
    );
Data Output Сообщения Notifications
UPDATE 2
```

Запрос завершён успешно, время выполнения: 53 msec.

DELETE:

```
2 WHERE stopover_id IN (
3 SELECT stopover_id
4 FROM stopover
5 WHERE type_stop = 'отменен'
6 );
7

Data Output Сообщения Notifications

DELETE 0

Запрос завершён успешно, время выполнения: 131 msec.
```

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:

1 запрос:

```
SELECT t.train_id, COUNT(*) AS free_seats
    FROM train t
    JOIN vagon v ON t.train_id = v.train_id
    JOIN place p ON v.vagon_id = p.vagon_id
    WHERE t.departure_date = CURRENT_DATE + INTERVAL '1 day'
 5
      AND p.occupancy = 'free'
 7
    GROUP BY t.train_id;
Data Output
            Сообщения
                       Notifications
=₊ |
     train_id
                 free_seats
     [PK] integer
                 bigint
1
              1
                          1
2
              2
                          2
```

✓ Запрос выполнен успешно. Общее время выполнения: 124 msec. обработано строк: 1. X

```
CREATE INDEX idx_train_departure_date ON train (departure_date);
CREATE INDEX idx_vagon_train_id ON vagon (train_id);
CREATE INDEX idx_place_vagon_id_occupancy ON place (vagon_id, occupancy);
```

```
EXPLAIN SELECT t.train_id, COUNT(*) AS free_seats
FROM train t
JOIN vagon v ON t.train_id = v.train_id
JOIN place p ON v.vagon_id = p.vagon_id
WHERE t.departure_date = CURRENT_DATE + INTERVAL '1 day'
AND p.occupancy = 'free'
GROUP BY t.train_id;
 🗸 Запрос выполнен успешно. Общее время выполнения: 55 msec. обработано строк: 13. 🗶
2 запрос:
  1
     SELECT pa.full_name
     FROM passenger pa
     JOIN ticket ti ON pa.passenger_id = ti.passenger_id
     JOIN stopover s ON ti.stopover_id = s.stopover_id
     JOIN regular schedule rs ON s.train number = rs.train number
     WHERE ti.date_of_purchase >= CURRENT_DATE - INTERVAL '1 day'
  7
       AND ti.date_of_purchase < CURRENT_DATE
       AND ti.departure_point = 'MockBa';
 Data Output Сообщения Notifications
      full_name
      character varying (40)
 1
      Sergey
 ✓ Запрос выполнен успешно. Общее время выполнения: 62 msec. обработано строк: 1. Х
CREATE INDEX idx_passenger_passenger_id ON passenger (passenger_id);
CREATE INDEX idx_ticket_date_of_purchase ON ticket (date_of_purchase);
CREATE INDEX idx_stopover_train_number ON stopover (train_number);
CREATE INDEX idx_regular_schedule_train_departure_point ON regular_schedule (train_departure_point);
EXPLAIN SELECT pa.full_name
FROM passenger pa
JOIN ticket ti ON pa.passenger_id = ti.passenger_id
JOIN stopover s ON ti.stopover_id = s.stopover_id
JOIN regular_schedule rs ON s.train_number = rs.train_number
WHERE ti.date_of_purchase >= CURRENT_DATE - INTERVAL '1 day'
AND ti.date_of_purchase < CURRENT_DATE</pre>
AND rs.train_departure_point = 'MockBa';
```

Выводы:

В процессе работы с pgAdmin я научился делать запросы на выборку и модификацию данных в базе данных PostgreSQL. Я освоил создание представлений, которые помогают упростить работу с данными, а также настройку индексов, что повышает производительность работы с базой данных. Эти знания помогут мне более эффективно работать с PostgreSQL и создавать более сложные запросы и представления для обработки большого объема данных. В целом, работа с pgAdmin позволила мне более полно использовать возможности PostgreSQL и повысить эффективность работы с базой данных.