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: Русинов В.А.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

Цель работы и практическое задание	9
1. Запросы к БД	4
2. Создание представлений	9
3. Запросы на модификацию данных	. 12
Вывод	. 16

Цель работы и практическое задание

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.

1. Запросы к БД

• Определить расчетное время полета по всем маршрутам. Листинг:

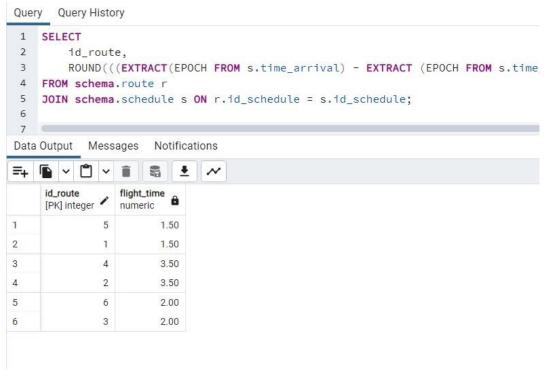
SELECT

id route,

ROUND(((EXTRACT(EPOCH FROM s.time_arrival) - EXTRACT (EPOCH FROM s.time_departure))/3600),2) AS flight time

FROM schema.route r

JOIN schema.schedule s ON r.id schedule = s.id schedule;



• Определить расход топлива по всем маршрутам. Листинг:

SELECT r.id route,

p.fuel rate AS fuel rate per hour,

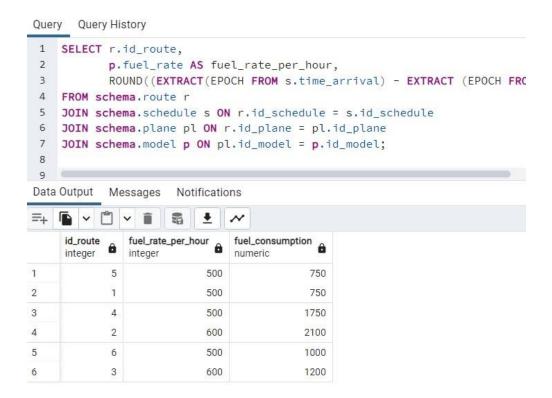
ROUND((EXTRACT(EPOCH FROM s.time_arrival) - EXTRACT (EPOCH FROM s.time_departure))/3600 * p.fuel rate,0) AS fuel consumption

FROM schema.route r

JOIN schema.schedule s ON r.id schedule = s.id schedule

JOIN schema.plane pl ON r.id plane = pl.id plane

JOIN schema.model p ON pl.id model = p.id model;



• Вывести данные о том, сколько свободных мест оставалось в самолетах, совершавших полет по заданному маршруте за вчерашний день. Листинг:

```
s.id_seat,
s.number,
s.row,
s.status
```

FROM schema.route r

JOIN schema.seat s ON r.id route = s.id route

WHERE DATE(r.date_departure) = CURRENT_DATE - INTERVAL '1 day' and s.status = 'Available' AND r.id_schedule = 2;



• Рассчитать убытки компании за счет непроданных билетов за вчерашний день.

SELECT COUNT(t.id ticket) AS total not sold tickets,

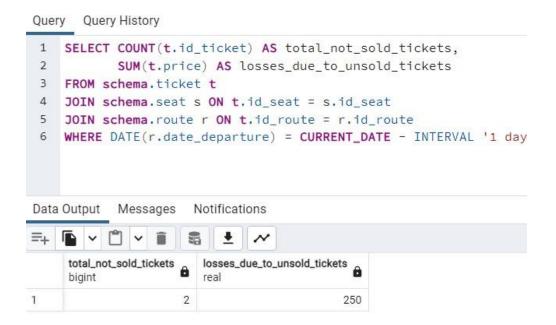
SUM(t.price) AS losses due to unsold tickets

FROM schema.ticket t

JOIN schema.seat s ON t.id seat = s.id seat

JOIN schema.route r ON t.id route = r.id route

WHERE DATE(r.date_departure) = CURRENT_DATE - INTERVAL '1 day' and t.status = 'Available';



• Определить, какой тип самолетов чаще всего летал в заданный аэропорт назначения. Листинг:

```
SELECT * FROM (
SELECT m.type of plane as type of plane,
   COUNT(*) AS flight count
FROM schema.route r
JOIN schema.schedule sc ON r.id schedule = sc.id schedule
JOIN schema.plane p ON r.id plane = p.id plane
JOIN schema.model m ON p.id model = m.id model
WHERE sc.id airport arrival = 1
GROUP BY type of plane
ORDER BY flight count DESC
)
WHERE flight count in (SELECT MAX(flight count) FROM (
SELECT m.type of plane as type of plane,
   COUNT(*) AS flight count
FROM schema.route r
JOIN schema.schedule sc ON r.id schedule = sc.id schedule
JOIN schema.plane p ON r.id plane = p.id plane
JOIN schema.model m ON p.id model = m.id model
WHERE sc.id airport arrival = 1
GROUP BY type of plane
ORDER BY flight count DESC
))
```

```
Query
      Query History
    SELECT * FROM (
 2
    SELECT m.type_of_plane as type_of_plane,
 3
           COUNT(*) AS flight_count
 4
    FROM schema.route r
 5
   JOIN schema.schedule sc ON r.id_schedule = sc.id_schedule
   JOIN schema.plane p ON r.id_plane = p.id_plane
 7
    JOIN schema.model m ON p.id_model = m.id_model
 8
    WHERE sc.id_airport_arrival = 1
 9
    GROUP BY type_of_plane
10
   ORDER BY flight_count DESC
11
   WHERE flight_count in (SELECT MAX(flight_count) FROM (
12
13
    SELECT m.type_of_plane as type_of_plane,
14
           COUNT(*) AS flight_count
   FROM schema.route r
15
    JOIN schema.schedule sc ON r.id_schedule = sc.id_schedule
16
17
    JOIN schema.plane p ON r.id_plane = p.id_plane
18
    JOIN schema.model m ON p.id_model = m.id_model
19
   WHERE sc.id_airport_arrival = 1
20
   GROUP BY type_of_plane
   ORDER BY flight_count DESC
21
Data Output Messages
                     Notifications
    □ ∨ □ ∨
     type_of_plane
                       flight_count
     character varying (20)
                       bigint
1
     Narrow-body
                                3
```

• Вывести список самолетов, "возраст" которых превышает средний "возраст" самолетов этого типа. Листинг:

```
SELECT p.id_plane,

p.tail_number,

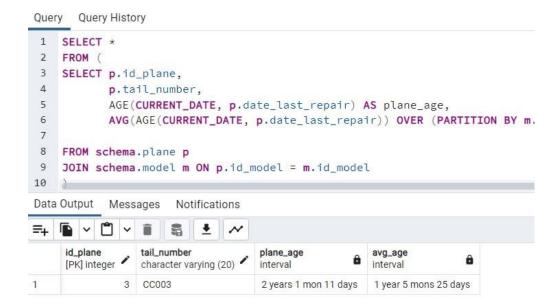
AGE(CURRENT_DATE, p.date_last_repair) AS plane_age,

AVG(AGE(CURRENT_DATE, p.date_last_repair)) OVER (PARTITION BY m.type_of_plane) AS avg_age

FROM schema.plane p

JOIN schema.model m ON p.id_model = m.id_model
)
```

WHERE plane_age > avg_age



2. Создание представлений

• для пассажиров авиакомпании о рейсах в Москву на ближайшую неделю; Листинг:

```
CREATE VIEW schema.passenger_flights_to_moscow AS SELECT

DISTINCT

r.id_route,

r.date_departure,

r.date_arrival

FROM

schema.route r JOIN schema.schedule s ON

r.id_schedule = s.id_schedule JOIN

schema.airport a ON s.id_airport_arrival = a.id_airport

WHERE

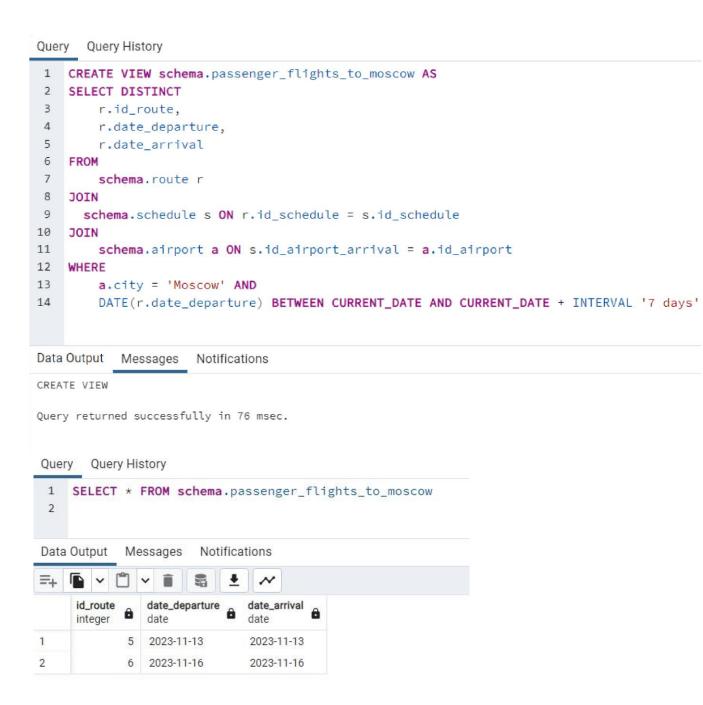
a.city = 'Moscow' AND
```

DATE(r.date departure)

INTERVAL '7 days';

9

BETWEEN CURRENT DATE AND CURRENT DATE



• количество самолетов каждого типа, летавшими за последний месяц. Листинг:

CREATE VIEW schema.aircraft count by type AS

SELECT

m.type_of_plane,

COUNT(r.id plane) AS airplane count

FROM

schema.route r

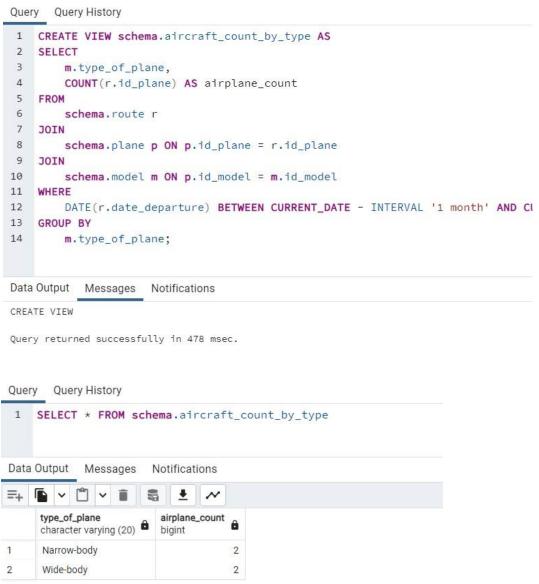
JOIN schema.plane p ON p.id_plane =
r.id_plane JOIN schema.model m ON
p.id_model = m.id_model

WHERE

DATE(r.date_departure) BETWEEN CURRENT_DATE - INTERVAL '1 month' AND CURRENT DATE

GROUP BY

m.type_of_plane;

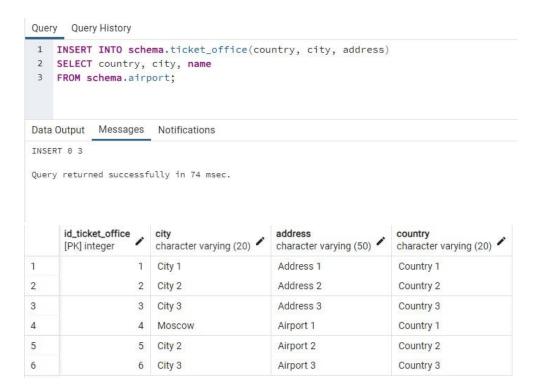


- 3. Запросы на модификацию данных
 - INSERT вставить в таблицу с кассами кассы, находящиеся в аэропортах Листинг:

INSERT INTO schema.ticket office(country, city, address)

SELECT country, city, name

FROM schema.airport;



 UPDATE – обновить дату последнего ремонта на сегодняшнюю для всех самолетов типа wide-body

Листинг:

UPDATE schema.plane

SET date last repair = CURRENT DATE

WHERE id_model IN (SELECT id_model FROM schema.model WHERE type_of_plane = 'Wide-body');

```
Query Query History

1    UPDATE schema.plane
2    SET date_last_repair = CURRENT_DATE
3    WHERE id_model IN (SELECT id_model FROM schema.model WHERE type_of_plane = 'Wide-body');

Data Output    Messages    Notifications

UPDATE 1

Query returned successfully in 165 msec.
```

	id_plane [PK] integer	tail_number character varying (20)	flight_hours integer	date_last_repair date	id_model integer	id_company integer
1	1	AA001	1000	2023-01-01	1	1
2	2	BB002	2000	2023-11-10	2	2
3	3	CC003	500	2021-09-29	1	2

• DELETE – удалить все самолеты, принадлежащие компаниям определенной страны Листинг:

DELETE FROM schema.plane

WHERE id_company IN (SELECT id_company FROM schema.company WHERE country = 'Country 3');



4. Создание индексов Листинг:

CREATE INDEX idx plane ON schema.route (id plane);

CREATE INDEX idx fuel ON schema.plane (flight hours, id model);

Без индексов:

```
Query Query History

1    SELECT (EXTRACT(EPOCH FROM s.time_arrival) - EXTRACT (EPOCH FROM 2 FROM schema.route r 3    JOIN schema.schedule s ON r.id_schedule = s.id_schedule;

Data Output    Messages    Notifications

Successfully run. Total query runtime: 98 msec. 6 rows affected.
```

Query Query History

```
1
    SELECT r.id_route,
 2
           p.fuel_rate AS fuel_rate_per_hour,
 3
           (EXTRACT(EPOCH FROM s.time_arrival) - EXTRA
   FROM schema.route r
 4
 5
   JOIN schema.schedule s ON r.id_schedule = s.id_sch
   JOIN schema.plane pl ON r.id_plane = pl.id_plane
    JOIN schema.model p ON pl.id_model = p.id_model;
Data Output
           Messages
                      Notifications
Successfully run. Total query runtime: 116 msec.
6 rows affected.
```

С индексами:

```
Query Query History

1    SELECT (EXTRACT(EPOCH FROM s.time_arrival) - EXTRACT (EPOCH F
2    FROM schema.route r
3    JOIN schema.schedule s ON r.id_schedule = s.id_schedule;

Data Output Messages Notifications
```

Successfully run. Total query runtime: 80 msec. 6 rows affected.

Query Query History

```
1
    SELECT r.id_route,
2
           p.fuel_rate AS fuel_rate_per_hour,
3
           (EXTRACT(EPOCH FROM s.time_arrival) - EXTRACT (EPC
4
   FROM schema.route r
5
   JOIN schema.schedule s ON r.id_schedule = s.id_schedule
   JOIN schema.plane pl ON r.id_plane = pl.id_plane
6
7
   JOIN schema.model p ON pl.id_model = p.id_model;
8
           Messages
Data Output
                     Notifications
```

Successfully run. Total query runtime: 79 msec. 6 rows affected.

Вывод

В ходе лабораторной работы я освоил работу с различными SQL-запросами к базе данных, также создание представлений и индексов. Также сравнил время работы SELECT запросов с индексами и без. Разумеется, с индексами время меньше.