Запросы на выборку и модификацию данных. Представления. Работа с индексами

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 лабораторной работы №2, 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.

Выполнил: Мищенко Максим K3239

Запросы
1 Определить расчетное время полета по всем маршрутам (в часах):

```
SELECT
   s.route_id,
   EXTRACT(EPOCH FROM (s.planned_time_arrival - s.planned_time_departure)) / 3600
AS flight_duration_hours
FROM public.schedule s;
```

2 Определить расход топлива по всем маршрутам.

```
SELECT
   f.flight_id,
   f.route_id,
   f.distance * pm.fuel_consumption AS estimated_fuel_consumption
FROM public.flights f
JOIN public.planes p      ON f.plane_id = p.plane_id
JOIN public.plane_models pm ON p.model_id = pm.model_id;
```

3 Вывести данные о том, сколько свободных мест оставалось в самолётах,

```
SELECT
   f.flight_id,
   pm.seat_count
     - COALESCE(tkt.sold_tickets, 0) AS free_seats
FROM public.flights f
JOIN public.planes p      ON f.plane_id = p.plane_id
JOIN public.plane_models pm ON p.model_id = pm.model_id
LEFT JOIN (
   SELECT flight_id, COUNT(*) AS sold_tickets
   FROM public.tickets
   WHERE flight_id = :flight_id
     AND DATE(sale_channel) IS NULL  -- если нужен фильтр по дате продажи, замените
на соответствующий столбец
   GROUP BY flight_id
) tkt ON f.flight_id = tkt.flight_id
WHERE f.flight_id = :flight_id
  AND DATE(f.departure_datetime_real) = current_date - 1;
```

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

```
WITH flights_yesterday AS (
   SELECT flight_id, plane_id
   FROM public.flights
   WHERE DATE(departure_datetime_real) = current_date - 1
),
```

```
sold AS (
   SELECT flight_id, COUNT(*) AS sold_cnt
   FROM public.tickets
   WHERE DATE(sale_channel) IS NULL  -- при необходимости заменить на дату
продажи
     AND flight_id IN (SELECT flight_id FROM flights_yesterday)
   GROUP BY flight_id
),
capacity AS (
   SELECT
      fy.flight_id,
      pm.seat_count,
      AVG(s.base_price) AS avg_price
   FROM flights_yesterday fy
   JOIN public.planes p      ON fy.plane_id = p.plane_id
   JOIN public.plane_models pm ON p.model_id = pm.model_id
   JOIN public.seats s       ON TRUE  -- если цены универсальны; иначе нужен маппинг
«какой самолёт — какие места»
   GROUP BY fy.flight_id, pm.seat_count
)
SELECT
   SUM((c.seat_count - COALESCE(s.sold_cnt, 0)) * c.avg_price) AS total_loss
FROM capacity c
LEFT JOIN sold s ON c.flight_id = s.flight_id;
```

5 Определить, какой тип самолётов чаще всего летал в заданный аэропорт назначения.

```
SELECT
   p.model_id,
   COUNT(*) AS flights_count
FROM public.flights f
JOIN public.schedule s ON f.route_id = s.route_id
JOIN public.planes p   ON f.plane_id = p.plane_id
WHERE s.arrival_airport_code = :dest_code
GROUP BY p.model_id
ORDER BY flights_count DESC
LIMIT 1;
```

6 Список самолётов, “возраст” которых превышает средний “возраст” этого типа.

```
WITH avg_age AS (
   SELECT
      model_id,
      AVG(EXTRACT(DAY FROM (current_date - last_maintenance_date))) AS avg_days
   FROM public.planes
```

```
    GROUP BY model_id
)
SELECT
    pl.plane_id,
    pl.model_id,
    current_date - pl.last_maintenance_date AS age_days
FROM public.planes pl
JOIN avg_age a ON pl.model_id = a.model_id
WHERE EXTRACT(DAY FROM (current_date - pl.last_maintenance_date)) > a.avg_days;
```

7 Определить тип самолётов, летающих во все аэропорты назначения.

```
SELECT DISTINCT pm.model_id
FROM public.plane_models pm
WHERE NOT EXISTS (
    SELECT airport_code
    FROM public.airports
    EXCEPT
    SELECT s.arrival_airport_code
    FROM public.flights f
    JOIN public.schedule s ON f.route_id = s.route_id
    JOIN public.planes p    ON f.plane_id = p.plane_id
    WHERE p.model_id = pm.model_id
);
```

Представления

1 Для пассажиров авиакомпании о рейсах в Москву на ближайшую неделю

```
CREATE OR REPLACE VIEW view_passenger_moscow_week AS
SELECT
    p.passenger_id,
    p.full_name,
    f.flight_id,
    s.planned_time_departure,
    s.planned_time_arrival,
    s.departure_airport_code,
    s.arrival_airport_code
FROM public.passengers p
JOIN public.tickets t ON p.passenger_id = t.passenger_id
JOIN public.flights f ON t.flight_id = f.flight_id
JOIN public.schedule s ON f.route_id = s.route_id
JOIN public.airports a ON s.arrival_airport_code = a.airport_code
WHERE a.city ILIKE 'Moscow'
  AND s.planned_time_departure BETWEEN current_date AND current_date + INTERVAL '7
days';
```

2 Количество самолётов каждого типа, летавшими за последний месяц

```
CREATE OR REPLACE VIEW view_plane_type_last_month AS
SELECT
   p.model_id,
   COUNT(DISTINCT p.plane_id) AS planes_flown
FROM public.flights f
JOIN public.planes p ON f.plane_id = p.plane_id
WHERE f.departure_datetime_real >= (current_date - INTERVAL '1 month')
  AND f.departure_datetime_real < current_date
GROUP BY p.model_id;
```

Хранимые процедуры
1 Поиск билетов в заданный пункт назначения

```
CREATE OR REPLACE FUNCTION sp_find_tickets(dest_code VARCHAR)
RETURNS TABLE(
   ticket_id    INTEGER,
   flight_id    INTEGER,
   passenger_id INTEGER,
   seat_id      INTEGER,
   sale_channel VARCHAR,
   status       VARCHAR
) AS $$
BEGIN
   RETURN QUERY
   SELECT
      t.ticket_id,
      t.flight_id,
      t.passenger_id,
      t.seat_id,
      t.sale_channel,
      t.status
   FROM public.tickets t
   JOIN public.flights f      ON t.flight_id = f.flight_id
   JOIN public.schedule s     ON f.route_id = s.route_id
   WHERE s.arrival_airport_code = dest_code;
END;
$$ LANGUAGE plpgsql;
```

2 Создание новой кассы продажи билетов

```
CREATE OR REPLACE PROCEDURE sp_create_cash_register(
   in_address VARCHAR,
   in_status  VARCHAR
```

```
)
LANGUAGE plpgsql
AS $$
BEGIN
   INSERT INTO public.cash_registers(address, status)
   VALUES (in_address, in_status);
END;
$$;
```

3 Определить расход топлива по всем маршрутам за истекший месяц

```
CREATE OR REPLACE FUNCTION sp_fuel_consumption_last_month()
RETURNS TABLE(
   route_id INTEGER,
   total_fuel DOUBLE PRECISION
) AS $$
BEGIN
   RETURN QUERY
   SELECT
      f.route_id,
      SUM(f.distance * pm.fuel_consumption) AS total_fuel
   FROM public.flights f
   JOIN public.planes p      ON f.plane_id = p.plane_id
   JOIN public.plane_models pm ON p.model_id = pm.model_id
   WHERE f.departure_datetime_real >= (current_date - INTERVAL '1 month')
     AND f.departure_datetime_real < current_date
   GROUP BY f.route_id;
END;
$$ LANGUAGE plpgsql;
```

-- 1.1 код, название и город всех аэропортов со статусом 'active'

```
SELECT airport_code,
      name      AS airport_name,
      city
FROM public.airports
WHERE status = 'active';
```

-- 1.2 идентификатор рейса, дату вылета и расстояние для рейсов авиакомпании 'Airline 1', где distance > 1000

```
SELECT f.flight_id,
      f.departure_datetime_real,
      f.distance
FROM public.flights AS f
JOIN public.planes  AS p ON f.plane_id = p.plane_id
JOIN public.flight_company AS c ON p.company_id = c.company_id
```

```
WHERE c.name = 'Airline 1'
  AND f.distance > 1000;

-- 1.3 номера и типы мест, которые еще свободны (status='available') на рейсе 1
SELECT s.seat_number,
       s.seat_type
FROM public.seats AS s
JOIN public.tickets AS t ON s.seat_id = t.seat_id
WHERE t.flight_id = 1
  AND s.status = 'available';
```

2.

2.1

```
CREATE OR REPLACE VIEW active_airports AS
SELECT airport_code,
       name      AS airport_name,
       city,
       country
FROM public.airports
WHERE status = 'active';

select * from active_airports
```

2.2

```
CREATE OR REPLACE VIEW long_distance_flights AS
SELECT f.flight_id,
       c.name    AS company_name,
       f.distance,
       f.departure_datetime_real,
       f.arrival_datetime_real
FROM public.flights AS f
JOIN public.planes  AS p ON f.plane_id = p.plane_id
JOIN public.flight_company AS c ON p.company_id = c.company_id
WHERE f.distance > 5000;
```

```
select * from long_distance_flights
```

2.3

```
CREATE OR REPLACE VIEW passenger_tickets AS
SELECT t.ticket_id,
       p.full_name   AS passenger_name,
       f.flight_id,
       s.seat_number,
       t.status      AS ticket_status
FROM public.tickets   AS t
JOIN public.passengers AS p ON t.passenger_id = p.passenger_id
JOIN public.seats     AS s ON t.seat_id = s.seat_id
JOIN public.flights   AS f ON t.flight_id = f.flight_id;

select * from passenger_tickets
```

3 Вложенные запросы

3.1

INSERT новый рейс для самолета с наибольшим пробегом, по маршруту с максимальным расстоянием

```
INSERT INTO public.flights (plane_id, route_id, status, distance, departure_datetime_real,
arrival_datetime_real)
VALUES (
  (SELECT plane_id FROM public.planes ORDER BY flight_hours DESC LIMIT 1),
  (SELECT route_id FROM public.schedule ORDER BY planned_time_departure DESC
LIMIT 1),
  'scheduled',
  (SELECT MAX(distance) FROM public.flights),
  now() + INTERVAL '1 day',
  now() + INTERVAL '1 day' + (SELECT MAX(distance)/500 || ' hours')::interval
```

```
);
```

3.2

UPDATE: Места, продаваемые на рейсах авиакомпании 'Airline 2', изменить статус на 'reserved'

```
UPDATE public.seats AS s
SET status = 'reserved'
WHERE seat_id IN (
  SELECT t.seat_id
  FROM public.tickets AS t
  JOIN public.flights AS f ON t.flight_id = f.flight_id
  JOIN public.planes  AS p ON f.plane_id = p.plane_id
  JOIN public.flight_company AS c ON p.company_id = c.company_id
  WHERE c.name = 'Airline 2'
);
```

3.3

DELETE Удаление билетов со статусом 'cancelled' старше 30 дней

```
DELETE FROM public.tickets
WHERE status = 'cancelled'
  AND ticket_id IN (
    SELECT ticket_id
    FROM public.tickets
    WHERE status = 'cancelled'
      AND now() - created_at::date > INTERVAL '30 days'
  );
```

4 создание индексов

4.1

```
EXPLAIN ANALYZE
SELECT flight_id, distance
FROM public.flights
WHERE distance > 1000;
```

"Execution Time: 0.020 ms"

```
CREATE INDEX idx_flights_distance ON public.flights(distance);

EXPLAIN ANALYZE
SELECT flight_id, distance
FROM public.flights
WHERE distance > 1000;
```

“Execution Time: 0.010 ms”

4.2

```
EXPLAIN ANALYZE
SELECT *
FROM public.tickets
WHERE flight_id = 1
  AND passenger_id = 10;
```

"Execution Time: 0.023 ms"

```
CREATE INDEX idx_tickets_flight_passenger ON public.tickets(flight_id, passenger_id);

EXPLAIN ANALYZE
SELECT *
FROM public.tickets
WHERE flight_id = 1
  AND passenger_id = 10;
```

"Execution Time: 0.012 ms"

Вывод

1. Представления упрощают повторное использование сложных запросов, скрывая детали соединений.
2. Подзапросы в модификациях позволяют динамически извлекать значения (например, по максимальному пробегу).
3. Создание индексов существенно ускоряет выборки по полям, участвующим в условиях WHERE, особенно для больших таблиц.