

Databases [Spring 25]

Assignment 1

March 2025
Innopolis University

Topic	Database optimization using indexes
Weight	15.00 points (15.00% of your course grade)
Deadline	Sunday, 30th of March 2025, 23:59
PostgreSQL version	17.4
Database	Flights Database demo-medium-en-20170815 Download link: https://edu.postgrespro.com/demo-medium-en-20170815.zip

1. Description

For this assignment, you are required to analyze and enhance the performance of a database, focusing on optimizing **four READ queries** and **one WRITE query** (five queries in total) that are frequently executed in your system. Your task is to improve query performance by creating appropriate indexes. Note that you are required to create **only one set of indexes** that will be applied for all five queries.

The database management system to be used is **PostgreSQL version 17.4**, and you will work with the **Flights Database**, which can be downloaded **[HERE](#)**.

2. Deliverables

For this assignment, you need to submit a **single .sql file** containing **only** indexes that will be applied for queries. Note that the same set of indexes will be applied for all five queries.

2.1. File name

Your file should be named as the first part of your Innopolis email address (the part before the '@' symbol), with the dot ('.') replaced by an underscore ('_'). Look at the examples below.

Your email	Your file name
i.ivanov@innopolis.university	i_ivanov.sql
i.ivanov@innopolis.ru	i_ivanov.sql
iv.ivanov@innopolis.university	iv_ivanov.sql
iva.ivanov@innopolis.university	iva_ivanov.sql

!!! An incorrectly named file will be penalized by a **50% reduction of the assignment grade**.

2.2. Rules & Constraints

- Your file must be an **executable** SQL script.
 - Your file should not contain indexes with the **duplicate names**.
 - Ensure that you have the **semicolon character** (';') at the end of each line.
- Your file should **NOT** contain more than **1000 indexes**.
- Your file should **NOT** contain more than **1000 lines of code**.
- Your file should contain **only** **CREATE INDEX** statements. Your file should **NOT** contain **DROP INDEX**, **CREATE TABLE** or any other SQL commands except index creations.

!!! Failing to follow any of these rules will result in a **50% reduction of the assignment grade**.

2.3. Submission File Example

```
CREATE INDEX index_1 ON some_table_1(attribute_1_1, attribute_1_2);
CREATE INDEX index_2 ON some_table_2(attribute_2_1, attribute_2_2);
...
```

We **strongly recommend** you to run your file on a **fresh database installation** and ensure it executes without errors before submitting it.

3. Plagiarism & Late Submissions

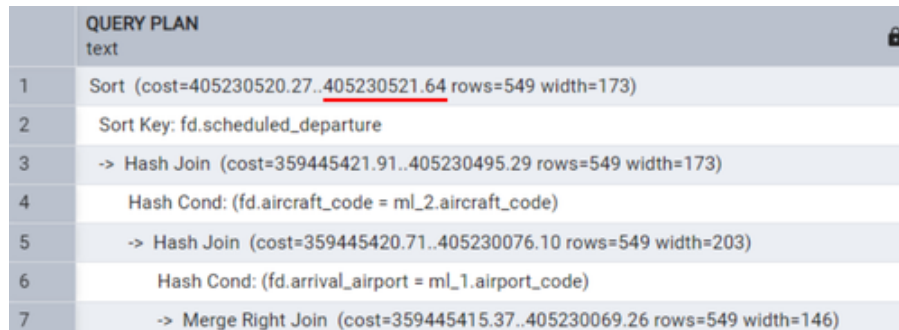
!!! This assignment is **individual**. All cases of **plagiarism** will result in a **ZERO** grade for this assignment and a report to the Department of Education.

!!! **Late submissions** without a legal excuse approved by the Department of Education will not be considered and will result in a **ZERO** grade for the assignment. In case you have a **legal excuse** that coincides with the duration of the assignment, you should inform your TA.

4. Grading

Your SQL script will be executed on a **freshly installed database**, after which each of the five queries will be run. For each student, we will compare the **cost** of the queries before creating indexes and the **cost** of the queries after creating indexes.

We will use the second value of the **query cost** (the one after '..**'**) in the first line of query **EXPLAIN** output. Please check the following image.



	QUERY PLAN text
1	Sort (cost=405230520.27.. <u>405230521.64</u> rows=549 width=173)
2	Sort Key: fd.scheduled_departure
3	-> Hash Join (cost=359445421.91..405230495.29 rows=549 width=173)
4	Hash Cond: (fd.aircraft_code = ml_2.aircraft_code)
5	-> Hash Join (cost=359445420.71..405230076.10 rows=549 width=203)
6	Hash Cond: (fd.arrival_airport = ml_1.airport_code)
7	-> Merge Right Join (cost=359445415.37..405230069.26 rows=549 width=146)

Students will be ranked based on the total optimization achieved for all five queries, and grades will be assigned as follows:

Rank	Grade
Top 30% of students	100.00% (15.00 points)
Next 40% of students	85.00% (12.75 points)
Next 25% of students	60.00% (9.00 points)
Last 5% of students	50.00% (7.50 points)

5. Queries

5.1. Query 1

```
SELECT p.passenger_id, p.passenger_name, fd.flight_no, fd.scheduled_departure,
fd.scheduled_arrival, da.airport_name AS dep_airport, aa.airport_name AS
arr_airport, a.model, fs.seat_no, li.next_flight_no, li.layover_duration
FROM
  (SELECT ticket_no, passenger_id, passenger_name
   FROM Tickets
   WHERE passenger_name = 'ADELINA IVANOVA') AS p
INNER JOIN Ticket_flights ON p.ticket_no = Ticket_flights.ticket_no
INNER JOIN Flights AS fd ON Ticket_flights.flight_id = fd.flight_id
INNER JOIN Airports AS da ON fd.departure_airport = da.airport_code
INNER JOIN Airports AS aa ON fd.arrival_airport = aa.airport_code
INNER JOIN Aircrafts AS a ON fd.aircraft_code = a.aircraft_code
INNER JOIN Boarding_passes AS fs ON Ticket_flights.ticket_no = fs.ticket_no AND
Ticket_flights.flight_id = fs.flight_id
LEFT JOIN
  (SELECT bp.ticket_no, f2.flight_no AS next_flight_no, MIN(f2.scheduled_departure)
   OVER (PARTITION BY bp.ticket_no) - MAX(f1.scheduled_arrival) OVER (PARTITION BY
   bp.ticket_no) AS layover_duration
   FROM Boarding_passes bp
   INNER JOIN Flights f1 ON bp.flight_id = f1.flight_id
   INNER JOIN Flights f2 ON f1.arrival_airport = f2.departure_airport AND
   f2.scheduled_departure > f1.scheduled_arrival) AS li ON Ticket_flights.ticket_no =
li.ticket_no
ORDER BY fd.scheduled_arrival DESC;
```

5.2. Query 2

```
WITH RECURSIVE recursive_routes AS (
  SELECT f.departure_airport AS origin, f.arrival_airport AS destination,
  ARRAY[f.flight_id] AS flight_path, ARRAY[f.departure_airport,
  f.arrival_airport]::bpchar[] AS airport_path, (f.scheduled_arrival -
  f.scheduled_departure) AS total_duration, 1 AS hops, f.flight_no,
  f.scheduled_departure, f.scheduled_arrival, tf.amount::numeric AS base_cost
  FROM
    flights f
  JOIN ticket_flights tf ON f.flight_id = tf.flight_id
  JOIN seats s ON f.aircraft_code = s.aircraft_code
  WHERE
    f.status NOT IN ('Cancelled')
    AND f.scheduled_departure > bookings.now()
    AND f.scheduled_departure < bookings.now() + INTERVAL '30 days'
    AND tf.fare_conditions = 'Economy'
    AND s.fare_conditions = 'Economy'
  UNION ALL
```

```

        SELECT r.origin, f.arrival_airport AS destination, r.flight_path || f.flight_id,
        r.airport_path || f.arrival_airport, r.total_duration + (f.scheduled_arrival -
        f.scheduled_departure), r.hops + 1, r.flight_no, r.scheduled_departure,
        f.scheduled_arrival, r.base_cost + tf.amount
        FROM
        recursive_routes r
        JOIN flights f ON r.destination = f.departure_airport
        JOIN ticket_flights tf ON f.flight_id = tf.flight_id
        JOIN seats s ON f.aircraft_code = s.aircraft_code
        WHERE
        f.status NOT IN ('Cancelled')
        AND f.scheduled_departure > r.scheduled_arrival + INTERVAL '1 hour'
        AND f.scheduled_departure < r.scheduled_arrival + INTERVAL '24 hours'
        AND NOT f.arrival_airport = ANY(r.airport_path)
        AND r.hops < 5
        AND tf.fare_conditions = 'Economy'
        AND s.fare_conditions = 'Economy'
    ),
    route_details AS (
        SELECT r.origin, r.destination, r.hops, r.flight_path, r.airport_path,
        r.total_duration, r.base_cost, r.scheduled_departure, r.scheduled_arrival,
        (SELECT COUNT(*)
         FROM flights f6
         WHERE f6.departure_airport = r.origin
         AND f6.arrival_airport = r.destination
         AND f6.status NOT IN ('Cancelled')
         AND f6.scheduled_departure > bookings.now()
         AND f6.scheduled_departure < bookings.now() + INTERVAL '30 days'
        ) AS direct_flight_count
        FROM
        recursive_routes r
    )

```

```

SELECT DISTINCT ON (rd.origin, rd.destination) rd.origin, dep.city AS
departure_city, rd.destination, arr.city AS arrival_city, rd.hops, rd.flight_path,
rd.airport_path, rd.total_duration, rd.base_cost,
(
    SELECT AVG(tf2.amount)
    FROM ticket_flights tf2
    JOIN flights f2 ON tf2.flight_id = f2.flight_id
    WHERE f2.departure_airport = rd.origin
    AND f2.arrival_airport = rd.destination
    AND tf2.fare_conditions = 'Economy'
    AND f2.scheduled_departure > bookings.now() - INTERVAL '180 days'
) AS avg_direct_cost,
(
    SELECT COUNT(*)
    FROM boarding_passes bp

```

```

JOIN flights f3 ON bp.flight_id = f3.flight_id
WHERE f3.departure_airport = rd.origin
AND f3.arrival_airport = rd.destination
AND f3.scheduled_departure > bookings.now() - INTERVAL '30 days'
) AS recent_passenger_count,
(
SELECT AVG(EXTRACT(EPOCH FROM (f5.actual_arrival - f5.scheduled_arrival)))
FROM flights f5
WHERE f5.departure_airport = rd.origin
AND f5.arrival_airport = rd.destination
AND f5.scheduled_departure > bookings.now() - INTERVAL '180 days'
AND f5.actual_arrival IS NOT NULL
) AS avg_delay_seconds,
jsonb_agg(DISTINCT jsonb_build_object(
'aircraft_code', ac.aircraft_code,
'model', ac.model,
'range', ac.range
)) AS aircraft_used,
(
SELECT json_agg(monthly_data)
FROM (
SELECT json_build_object(
'month', EXTRACT(MONTH FROM f4.scheduled_departure),
'passengers', COUNT(DISTINCT bp2.ticket_no),
'top_passengers', json_agg(
json_build_object(
'ticket_no', top_passengers.ticket_no,
'flight_count', top_passengers.flight_count
)
)
) AS monthly_data
FROM flights f4
LEFT JOIN boarding_passes bp2 ON f4.flight_id = bp2.flight_id
LEFT JOIN LATERAL (
SELECT bp2_inner.ticket_no, COUNT(*) AS flight_count
FROM boarding_passes bp2_inner
JOIN flights f_inner ON bp2_inner.flight_id = f_inner.flight_id
WHERE f_inner.departure_airport = rd.origin
AND f_inner.arrival_airport = rd.destination
AND EXTRACT(MONTH FROM f_inner.scheduled_departure) = EXTRACT(MONTH FROM
f4.scheduled_departure)
GROUP BY bp2_inner.ticket_no
ORDER BY flight_count DESC
LIMIT 3
) AS top_passengers ON true
WHERE f4.departure_airport = rd.origin
AND f4.arrival_airport = rd.destination
AND f4.scheduled_departure > bookings.now() - INTERVAL '730 days'

```

```

        GROUP BY EXTRACT(MONTH FROM f4.scheduled_departure)
    ) AS monthly_counts
    ) AS historical_monthly_data,
    RANK() OVER (PARTITION BY rd.origin, rd.destination ORDER BY rd.total_duration)
AS duration_rank
FROM
    route_details rd
JOIN airports_data dep ON rd.origin = dep.airport_code
JOIN airports_data arr ON rd.destination = arr.airport_code
JOIN LATERAL (
    SELECT DISTINCT ac.aircraft_code, ac.model, ac.range
    FROM unnest(rd.flight_path) AS flight_ids
    JOIN flights f5 ON f5.flight_id = flight_ids
    JOIN aircrafts_data ac ON f5.aircraft_code = ac.aircraft_code
    ) ac ON true
WHERE
    (rd.hops > 1 OR rd.direct_flight_count < 10)
    AND dep.city <> arr.city
GROUP BY
    rd.origin, dep.city, rd.destination, arr.city, rd.hops, rd.flight_path,
    rd.airport_path, rd.total_duration, rd.base_cost
ORDER BY
    rd.origin, rd.destination, rd.total_duration, rd.base_cost;

```

5.3. Query 3

```

SELECT
    t.passenger_name,
    t.passenger_id,
    COUNT(DISTINCT tf.ticket_no) AS total_tickets,
    SUM(tf.amount) AS total_amount_spent,
    COUNT(DISTINCT f.flight_id) AS total_flights,
    COUNT(DISTINCT bp.seat_no) AS total_seats_used,
    COUNT(DISTINCT a.aircraft_code) AS total_aircrafts_used
FROM
    tickets t
JOIN
    ticket_flights tf ON t.ticket_no = tf.ticket_no
JOIN
    flights f ON tf.flight_id = f.flight_id
JOIN
    boarding_passes bp ON tf.ticket_no = bp.ticket_no AND tf.flight_id =
bp.flight_id
JOIN
    aircrafts_data a ON f.aircraft_code = a.aircraft_code
WHERE
    f.actual_departure ≥ '2012-01-01 00:00:00'

```

```

        AND f.actual_departure < '2017-12-31 23:59:59'
GROUP BY
    t.passenger_name, t.passenger_id
HAVING
    COUNT(DISTINCT tf.ticket_no) > 0
ORDER BY
    total_amount_spent DESC;

```

5.4. Query 4

```

SELECT
    b.book_ref,
    COUNT(DISTINCT t.ticket_no) AS num_tickets,
    COUNT(DISTINCT tf.flight_id) AS num_flights,
    SUM(tf.amount) AS total_amount_for_flights,
    AVG(tf.amount) AS average_amount_per_flight,
    MAX(a.model→»'english') AS largest_aircraft_used,
    STRING_AGG(DISTINCT ap.city→»'english', ', ') AS cities_visited,
    (SELECT COUNT(*)
     FROM boarding_passes bp
     WHERE bp.ticket_no IN (SELECT ticket_no
                           FROM tickets
                           WHERE book_ref = b.book_ref)) AS
total_boarding_passes_issued,
    COUNT(DISTINCT f.departure_airport) AS unique_departure_airports,
    SUM(CASE WHEN tf.amount > 500 THEN 1 ELSE 0 END) AS flights_over_500
FROM
    bookings b
JOIN
    tickets t ON b.book_ref = t.book_ref
JOIN
    ticket_flights tf ON t.ticket_no = tf.ticket_no
JOIN
    flights f ON tf.flight_id = f.flight_id
JOIN
    aircrafts_data a ON f.aircraft_code = a.aircraft_code
JOIN
    airports_data ap ON f.arrival_airport = ap.airport_code OR f.departure_airport =
ap.airport_code
GROUP BY
    b.book_ref
HAVING
    COUNT(DISTINCT tf.flight_id) > 1 AND SUM(tf.amount) > 1000
ORDER BY
    total_amount_for_flights DESC, num_flights DESC
LIMIT 10;

```


5.5. Query 5 (Update Query)

```
WITH aircraft_avg_duration AS (  
    SELECT  
        f.aircraft_code,  
        AVG(EXTRACT(EPOCH FROM (f.actual_arrival - f.scheduled_departure)) / 60) AS  
avg_duration  
    FROM  
        bookings.flights f  
    GROUP BY  
        f.aircraft_code  
)  
UPDATE bookings.ticket_flights tf  
SET amount = amount * 1.15  
FROM  
    bookings.flights f  
JOIN aircraft_avg_duration aad ON f.aircraft_code = aad.aircraft_code  
WHERE  
    tf.flight_id = f.flight_id  
    AND aad.avg_duration > 180;
```