Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Факультет «Информационных технологий и анализа больших данных»

**ОТЧЕТ**

**по практической работе № 11**

**по теме «Способы соединения наборов строк. Подходы к оптимизации запросов.»**

**по дисциплине Б.1.1.3.4 Системы управления базами данных**

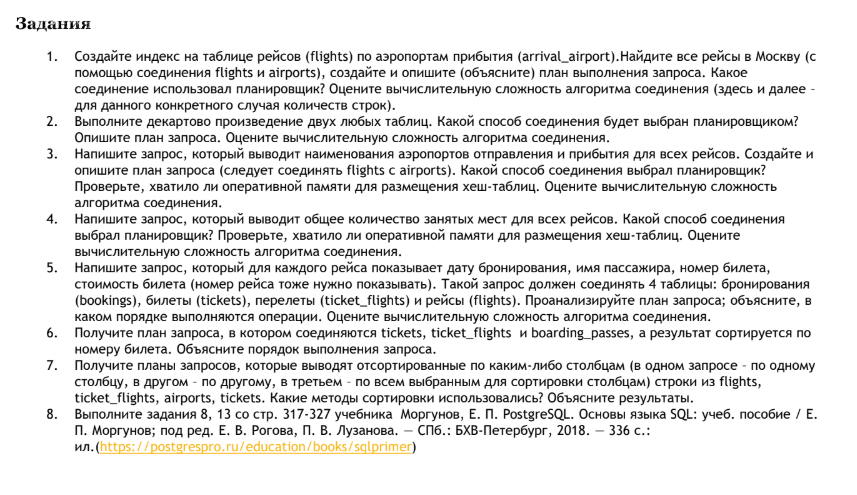
Выполнил: студент группы ИД23-1

Маслов Александр Николаевич

Проверил преподаватель: Цветков И.Е.

**Москва – 2025**

**Задания:**

****

**1.**

CREATE INDEX idx\_arrival\_airport ON flights(arrival\_airport);

EXPLAIN ANALYZE

SELECT f.flight\_id, a.city AS arrival\_city

FROM flights f

JOIN airports a ON f.arrival\_airport = a.airport\_code

WHERE a.city = 'Moscow';

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Используется Nested Loop – алгоритм соединения вложенными циклами.

Вычислительная сложность:

* Фильтрация аэропортов: O(n) — перебор аэропортов
* Соединение с flights:
  + Nested Loop: O(n × m)

**2.**

EXPLAIN ANALYZE

SELECT \*

FROM bookings, flights

LIMIT 1000

Изображение выглядит как текст, Шрифт, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

Планировщик использует Nested Loop.

Декартово произведение — это O(n × m).

Так как нет условия соединения — происходит полное перемножение строк.

**3.**

EXPLAIN ANALYZE

SELECT f.flight\_id, a1.airport\_name AS departure, a2.airport\_name AS arrival

FROM flights f

JOIN airports a1 ON f.departure\_airport = a1.airport\_code

JOIN airports a2 ON f.arrival\_airport = a2.airport\_code;

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Используется Hash Join, сложность O(n+m). Оперативной памяти хватило.

**4.**

EXPLAIN ANALYZE

SELECT f.flight\_id, COUNT(\*) AS seats\_taken

FROM flights f

JOIN ticket\_flights tf ON f.flight\_id = tf.flight\_id

GROUP BY f.flight\_id;

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.**

Используется Hash Aggregate, сложность O(n+m). Оперативной памяти хватило.

**5.**

EXPLAIN ANALYZE

SELECT b.book\_date, t.passenger\_name, t.ticket\_no, tf.amount, f.flight\_id

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;

**Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Контент, сгенерированный ИИ, может содержать ошибки.**

Планировщик использует несколько JOIN-ов:

Сначала соединяются bookings и tickets, затем с ticket\_flights, и наконец — flights.

В основном используется Hash Join.

Сложность: O (n + m + k + l)

**6.**

EXPLAIN ANALYZE

SELECT t.ticket\_no, tf.flight\_id, bp.boarding\_no

FROM tickets t

JOIN ticket\_flights tf ON t.ticket\_no = tf.ticket\_no

JOIN boarding\_passes bp ON t.ticket\_no = bp.ticket\_no

ORDER BY t.ticket\_no;

Изображение выглядит как текст, Шрифт, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

Планировщик использует Merge Join, а затем Index Scan.

Сложность: O(n + m + k)

**7.**

EXPLAIN ANALYZE

SELECT \* FROM flights ORDER BY flight\_id;

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

Контент, сгенерированный ИИ, может содержать ошибки.**

Используется Index Scan, так как создан индекс flights\_pkey.

EXPLAIN ANALYZE

SELECT \* FROM ticket\_flights ORDER BY amount DESC;

**Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.**

Используется Sort c параметром DESC. Метод выбран external merge, так как таблица большая и ему требуется дополнительная память.

EXPLAIN ANALYZE

SELECT \* FROM airports ORDER BY city, airport\_name;

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.**

Используется Sort. Метод выбран quicksort.

Так как таблица небольшая и ему не нужна дополнительная память.

EXPLAIN ANALYZE

SELECT \* FROM tickets ORDER BY passenger\_name;

**Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.**

Используется Sort. Метод выбран external merge, так как таблица большая и ему требуется дополнительная память.

**8.**

**Упражнение 8**

EXPLAIN ANALYZE

SELECT a.aircraft\_code,

a.model,

(SELECT COUNT(\*)

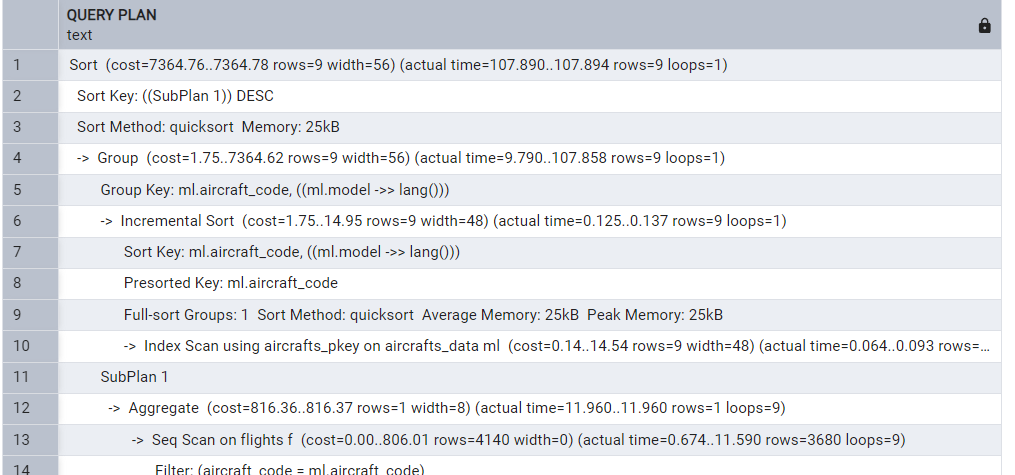
FROM flights f

WHERE f.aircraft\_code = a.aircraft\_code) AS flight\_count

FROM aircrafts a

GROUP BY 1, 2

ORDER BY 3 DESC;



EXPLAIN ANALYZE

SELECT a.aircraft\_code,

a.model,

COUNT(f.flight\_id) AS flight\_count

FROM aircrafts a

LEFT JOIN flights f ON a.aircraft\_code = f.aircraft\_code

GROUP BY 1, 2

ORDER BY 3 DESC;

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Коррелированные подзапросы проще читаются, но работают медленно на больших объемах, но при малых объемах коррелированные подзапросы работают быстрее (как пример в данных двух запросах, коррелированный подзапрос работает быстрее, так как в таблице всего 9 строчек).

Заменив их на JOIN, мы снижаем нагрузку и получаем линейную сложность.

**Упражнение 13**

CREATE INDEX bookings\_book\_date\_idx ON bookings (book\_date);

CREATE INDEX tickets\_book\_ref\_key ON tickets ( book\_ref );

EXPLAIN ANALYZE

SELECT num\_tickets, count( \* ) AS num\_bookings

FROM

( SELECT b.book\_ref, count( \* )

FROM bookings b, tickets t

WHERE date\_trunc( 'mon', b.book\_date ) = '2016-09-01'

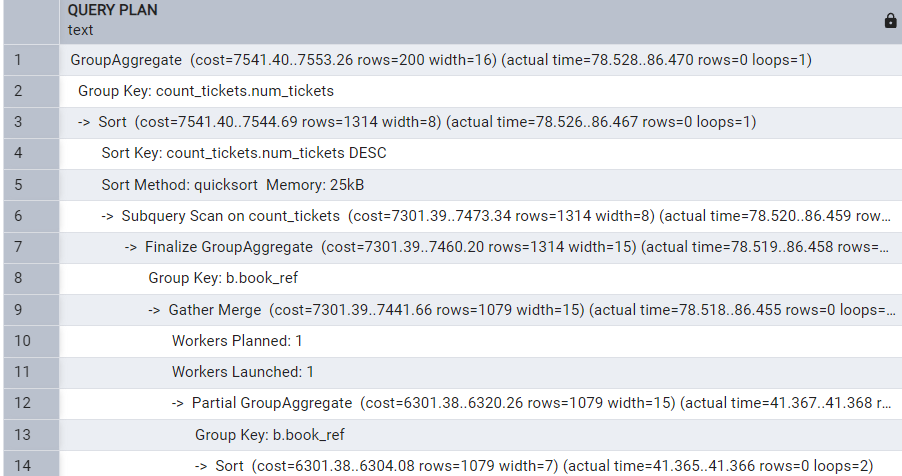
AND t.book\_ref = b.book\_ref

GROUP BY b.book\_ref

) AS count\_tickets( book\_ref, num\_tickets )

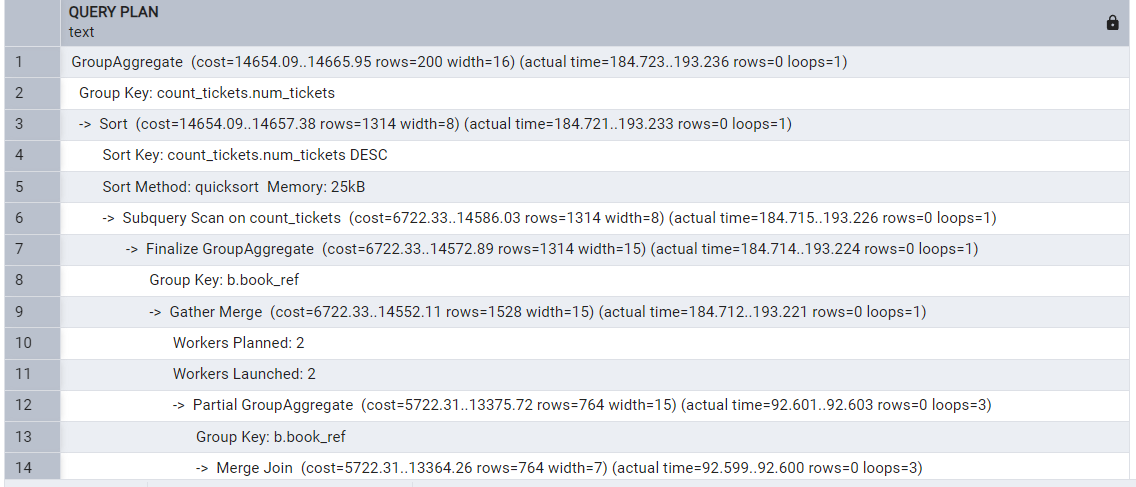
GROUP by num\_tickets

ORDER BY num\_tickets DESC;

****

SET enable\_hashjoin = off;

SET enable\_nestloop = off;

****

Селективность — это доля строк, попадающих в выборку.

При низкой селективности PostgresSQL будет вместо индекса использовать последовательное сканирование. Поэтому с данным запросом индекс по bookings не используется, так как селективность довольно мала, то есть много строк.

Даже при запрещенных Hash Join и Nest Loop, индекс по tickets будет использоваться, но с Merge Join.