Lab_4_Nemkov_KMBO_05_23

Task 1

Выбрать любой рейс (flight_id из таблицы flights). Дан запрос для получения всех данных по перелётам (ticket_flights) по выбранному рейсу:

```
SELECT * FROM ticket_flights WHERE flight_id=выбранный_рейс;
```

Проанализировать план запроса и оптимизировать его.

```
SELECT * FROM flights LIMIT 1;
```

	flight_id [PK] integer	flight_no character (6)	scheduled_departure timestamp with time zone	scheduled_arrival timestamp with time zone	departure_airport character (3)	arrival_airport character (3)	status character varying (20)	airc cha
1	2880	PG0216	2017-09-14 14:10:00+03	2017-09-14 15:15:00+03	DME	KUF	Scheduled	763

image-223.png

```
EXPLAIN ANALYZE SELECT * FROM ticket_flights WHERE flight_id = 2880;
```

	QUERY PLAN text
1	Gather (cost=1000.00..114677.77 rows=102 width=32) (actual time=222.001..226.443 rows=3 loops=1)
2	Workers Planned: 2
3	Workers Launched: 2
4	-> Parallel Seq Scan on ticket_flights (cost=0.00..113667.57 rows=42 width=32) (actual time=148.470..195.492 rows=1 loop…
5	Filter: (flight_id = 2880)
6	Rows Removed by Filter: 2797284
7	Planning Time: 0.697 ms
8	Execution Time: 226.471 ms

image-228.png

```
CREATE INDEX IF NOT EXISTS idx_ticket_flights_flight_id ON
ticket_flights (flight_id);
EXPLAIN ANALYZE SELECT * FROM ticket_flights WHERE flight_id = 2880;
```

	QUERY PLAN text
1	Index Scan using idx_ticket_flights_flight_id on ticket_flights (cost=0.43..394.78 rows=102 width=32) (actual time=0.023..0.026 rows=3 loop…
2	Index Cond: (flight_id = 2880)
3	Planning Time: 0.201 ms
4	Execution Time: 0.046 ms

image-227.png

Task 2

Для отображения данных в табло задержанных рейсов в каждом аэропорту прибытия (arrival_airport) используются данные из таблицы полётов (flights):

```
SELECT * FROM flights WHERE status='Delayed' AND
arrival_airport=аэропорт;
```

Проанализировать запрос для любого из аэропортов и оптимизировать его (подсказка: нужно использовать частичные индексы). Проанализировать запрос после добавления индекса. Проанализировать запросы для отображения всех рейсов данного аэропорта, а также для отображения всех задержанных рейсов по всем аэропортам.
Использует ли планировщик созданный индекс в этих запросах? Предположить причины такого поведения.

```
EXPLAIN ANALYZE SELECT * FROM flights WHERE arrival_airport='SVO';
```

	QUERY PLAN text
1	Seq Scan on flights (cost=0.00..5309.84 rows=19553 width=63) (actual time=0.015..14.876 rows=19348 loops…
2	Filter: (arrival_airport = 'SVO'::bpchar)
3	Rows Removed by Filter: 195519
4	Planning Time: 0.095 ms
5	Execution Time: 15.254 ms

image-230.png

```
EXPLAIN ANALYZE SELECT * FROM flights WHERE status='Delayed' AND
arrival_airport='SVO';
```

	QUERY PLAN text
1	Gather (cost=1000.00..5520.19 rows=3 width=63) (actual time=10.040..36.928 rows=2 loops=1)
2	Workers Planned: 1
3	Workers Launched: 1
4	-> Parallel Seq Scan on flights (cost=0.00..4519.89 rows=2 width=63) (actual time=4.860..9.955 rows=1 loops...
5	Filter: (((status)::text = 'Delayed'::text) AND (arrival_airport = 'SVO'::bpchar))
6	Rows Removed by Filter: 107432
7	Planning Time: 0.633 ms
8	Execution Time: 36.959 ms

image-229.png

```
CREATE INDEX IF NOT EXISTS idx_flights_arrival_airport_delayed ON
flights (arrival_airport) WHERE status = 'Delayed';
EXPLAIN ANALYZE SELECT * FROM flights WHERE arrival_airport='SVO';
EXPLAIN ANALYZE SELECT * FROM flights WHERE status='Delayed' AND
arrival_airport='SVO';
```

	QUERY PLAN text
1	Seq Scan on flights (cost=0.00..5309.84 rows=19553 width=63) (actual time=0.193..102.397 rows=19348 loops...
2	Filter: (arrival_airport = 'SVO'::bpchar)
3	Rows Removed by Filter: 195519
4	Planning Time: 4.538 ms
5	Execution Time: 103.763 ms

image-234.png

	QUERY PLAN text
1	Bitmap Heap Scan on flights (cost=4.16..15.90 rows=3 width=63) (actual time=0.074..0.076 rows=2 loops=1)
2	Recheck Cond: ((arrival_airport = 'SVO'::bpchar) AND ((status)::text = 'Delayed'::text))
3	Heap Blocks: exact=2
4	-> Bitmap Index Scan on idx_flights_arrival_airport_delayed (cost=0.00..4.16 rows=3 width=0) (actual time=0.058..0.058 rows=2 loop...
5	Index Cond: (arrival_airport = 'SVO'::bpchar)
6	Planning Time: 0.097 ms
7	Execution Time: 0.092 ms

image-232.png

Task 3

В системе продажи билетов необходимо часто просматривать рейсы за диапазон дат отправления (scheduled_departure) из аэропорта отправления (departure_airport) в аэропорт прибытия (arrival_airport).
Написать запрос для получения этих данных.
Рассмотреть несколько вариантов оптимизации: с одним индексом, с составным индексом (из нескольких столбцов) и с несколькими индексами.
Проверить на нескольких диапазонах дат и разными аэропортами. Какой из вариантов оказался самым оптимальным? Предположить, почему.

```
-- неделя
EXPLAIN ANALYZE SELECT * FROM flights
WHERE scheduled_departure BETWEEN '2016-01-31' AND '2016-02-07'
AND departure_airport = 'SVO'
AND arrival_airport = 'LED';

-- месяц
EXPLAIN ANALYZE SELECT * FROM flights
WHERE scheduled_departure BETWEEN '2016-01-1' AND '2016-01-31'
AND departure_airport = 'SVO'
AND arrival_airport = 'CSY';

-- 3 месяца
EXPLAIN ANALYZE SELECT * FROM flights
WHERE scheduled_departure BETWEEN '2016-01-01' AND '2016-04-01'
AND departure_airport = 'DME'
AND arrival_airport = 'CSY';

-- одиночный индекс
CREATE INDEX IF NOT EXISTS idx_flights_scheduled_departure ON
flights (scheduled_departure);
-- составной индекс
CREATE INDEX IF NOT EXISTS idx_flights_composite ON flights
(departure_airport,arrival_airport);
-- множественный индекс
CREATE INDEX IF NOT EXISTS idx_flights_departure ON flights
(departure_airport);
```

Task 4

В системе продажи билетов иногда нужно найти билеты пассажира (tickets) по номеру телефона. Телефон хранится в json-поле contact_data. Написать запрос для получения всей информации по номеру телефона. Оптимизировать запрос.

```
EXPLAIN ANALYZE SELECT * FROM tickets WHERE contact_data->>'phone' =
'+70110137563';
CREATE INDEX IF NOT EXISTS idx_tickets_phone ON tickets
((contact_data->>'phone'));
```

Task 5

Посмотреть размер ранее созданных индексов. Предположить, какие следует оставить, а какие удалить.
Размер индексов можно посмотреть с помощью запроса Index size/usage statistics (второй запрос по ссылке):
https://wiki.postgresql.org/wiki/Index_Maintenance

```
SELECT pg_size_pretty(pg_relation_size(indexrelid)) AS size,
       relname AS index_name
FROM pg_index
JOIN pg_class ON pg_index.indexrelid = pg_class.oid
ORDER BY pg_relation_size(indexrelid) DESC;
```

	size text	index_name name
1	325 MB	ticket_flights_pkey
2	307 MB	boarding_passes_pkey
3	170 MB	boarding_passes_flight_id_boarding_no_key
4	170 MB	boarding_passes_flight_id_seat_no_key
5	89 MB	idx_tickets_phone
6	89 MB	tickets_pkey
7	58 MB	idx_ticket_flights_flight_id
8	45 MB	bookings_pkey
9	6648 kB	flights_flight_no_scheduled_departure_key
10	4744 kB	flights_pkey
11	256 kB	pg_proc_proname_args_nsp_index
12	216 kB	pg_description_o_c_o_index
13	136 kB	pg_collation_name_enc_nsp_index
14	128 kB	pg_attribute_relid_attnam_index
15	96 kB	pg_proc_oid_index
16	88 kB	pg_attribute_relid_attnum_index
17	88 kB	pg_depend_depender_index
18	72 kB	pg_depend_reference_index
19	72 kB	pg_collation_oid_index
20	48 kB	seats_pkey

Total rows: 181 of 181 Query complete 00:00:00.106 Ln 45, Col 44

image-233.png

Task 6

(From lab_2, task_1)
Напишите запрос, который выводит статистику по количеству заказов (total_orders) и выручке (total_revenue) для всех возможных комбинаций города и региона, но только для тех пунктов выдачи, где выручка превышает среднюю выручку по их региону. Добавьте общие итоги по каждому городу и региону.

```
SELECT
    COALESCE(p.region, 'Все регионы') AS region,
    COALESCE(p.city, 'Все города') AS city,
    SUM(p.total_orders) AS total_orders,
    SUM(p.total_revenue) AS total_revenue
FROM pickup_points p
JOIN (
    SELECT
```

```
        region,
        AVG(total_revenue) AS avg_revenue
    FROM pickup_points
    GROUP BY region
) r ON p.region = r.region
WHERE p.total_revenue > r.avg_revenue
GROUP BY ROLLUP (p.region, p.city)
ORDER BY p.region, p.city;
```

Без индексов мы сталкиваемся с проблемой:
БД полностью перебирает все записи в таблице `pickup_points` дважды:

- Сначала для расчета средней выручки по регионам
- Потом для поиска точек с выручкой выше средней
  Это работает медленно, если записей много

Если мы добавим индекс для быстрого расчета средних по регионам

- наша бд быстро сгруппирует данные по регионам
- не нужно будет перебирать всю таблицу для расчета средних значений
  Ещё мы можем добавить индекс для поиска по выручке
- будет быстрый поиск точек с выручкой выше средней
- фильтрация будет работать эффективнее