Otimização de Consultas

...

Pedro Henrique Klein

Consula 1 - Lenta -> C1L

```sql
SELECT
    bookings.*
FROM
    flights,
    ticket_flights,
    tickets,
    bookings,
    airports
WHERE
    flights.flight_id = ticket_flights.flight_id
    AND
ticket_flights.ticket_no = tickets.ticket_no
    AND
tickets.book_ref = bookings.book_ref
    AND
airports.airport_code = flights.departure_airport
    AND
airports.airport_code = 'BAX'
ORDER BY
    bookings.total_amount DESC
```

Plano de Consulta da C1L

```
QUERY PLAN
Sort  (cost=224126.38..224229.88 rows=41399 width=21) (actual time=2943.316..2945.563 rows=27666 loops=1)
  Sort Key: bookings.total_amount DESC
  Sort Method: quicksort  Memory: 2498kB
  ->  Nested Loop  (cost=5323.95..220951.64 rows=41399 width=21) (actual time=22.907..2922.089 rows=27666 loops=1)
        ->  Seq Scan on airports_data ml  (cost=0.00..4.30 rows=1 width=4) (actual time=0.037..0.039 rows=1 loops=1)
              Filter: (airport_code = 'BAX'::bpchar)
              Rows Removed by Filter: 103
        ->  Nested Loop  (cost=5323.95..220533.35 rows=41399 width=25) (actual time=22.867..2917.668 rows=27666 loops=1)
              ->  Nested Loop  (cost=5323.52..200927.20 rows=41399 width=11) (actual time=22.854..2491.177 rows=27666 loops=1)
                    ->  Hash Join  (cost=5323.09..181201.83 rows=41399 width=18) (actual time=22.822..2123.642 rows=27666 loops=1)
                          Hash Cond: (ticket_flights.flight_id = flights.flight_id)
                          ->  Seq Scan on ticket_flights  (cost=0.00..153850.08 rows=8391708 width=18) (actual time=0.045..1140.624 rows=8391852 loops=1)
                          ->  Hash  (cost=5309.84..5309.84 rows=1060 width=8) (actual time=22.699..22.701 rows=1017 loops=1)
                                Buckets: 2048  Batches: 1  Memory Usage: 56kB
                                ->  Seq Scan on flights  (cost=0.00..5309.84 rows=1060 width=8) (actual time=0.023..22.562 rows=1017 loops=1)
                                      Filter: (departure_airport = 'BAX'::bpchar)
                                      Rows Removed by Filter: 213850
                    ->  Index Scan using tickets_pkey on tickets  (cost=0.43..0.48 rows=1 width=21) (actual time=0.013..0.013 rows=1 loops=27666)
                          Index Cond: (ticket_no = ticket_flights.ticket_no)
              ->  Index Scan using bookings_pkey on bookings  (cost=0.43..0.47 rows=1 width=21) (actual time=0.015..0.015 rows=1 loops=27666)
                    Index Cond: (book_ref = tickets.book_ref)
Planning Time: 0.856 ms
Execution Time: 2947.116 ms
```

Consulta 1 - Otimizada -> C10

```sql
WITH airports_filter AS (
    SELECT
        *
    FROM
        BOOKINGS.AIRPORTS
    WHERE
        AIRPORT_CODE = 'BAX'
)
SELECT
    b.BOOK_REF,
    b.BOOK_DATE,
    b.TOTAL_AMOUNT
FROM
    BOOKINGS.FLIGHTS AS f
INNER JOIN BOOKINGS.TICKET_FLIGHTS AS tf ON
    f.FLIGHT_ID = tf.FLIGHT_ID
INNER JOIN TICKETS AS t ON
    tf.TICKET_NO = t.TICKET_NO
INNER JOIN BOOKINGS.BOOKINGS AS b ON
    t.BOOK_REF = b.BOOK_REF
INNER JOIN airports_filter AS af ON
    f.DEPARTURE_AIRPORT = af.AIRPORT_CODE
ORDER BY
    b.TOTAL_AMOUNT
DESC;
```

Plano de Consulta da C10

```
QUERY PLAN
Sort  (cost=224126.38..224229.88 rows=41399 width=21) (actual time=2619.911..2620.951 rows=27666 loops=1)
  Sort Key: b.total_amount DESC
  Sort Method: quicksort  Memory: 2498kB
  ->  Nested Loop  (cost=5323.95..220951.64 rows=41399 width=21) (actual time=29.169..2601.867 rows=27666 l
        ->  Seq Scan on airports_data ml  (cost=0.00..4.30 rows=1 width=4) (actual time=0.032..0.033 rows=1 loops
              Filter: (airport_code = 'BAX'::bpchar)
              Rows Removed by Filter: 103
        ->  Nested Loop  (cost=5323.95..220533.35 rows=41399 width=25) (actual time=29.135..2597.952 rows=276
              ->  Nested Loop  (cost=5323.52..200927.20 rows=41399 width=11) (actual time=29.114..2205.768 rows=
                    ->  Hash Join  (cost=5323.09..181201.83 rows=41399 width=18) (actual time=29.075..1850.995 rows=
                          Hash Cond: (tf.flight_id = f.flight_id)
                          ->  Seq Scan on ticket_flights tf  (cost=0.00..153850.08 rows=8391708 width=18) (actual time=0.0
                          ->  Hash  (cost=5309.84..5309.84 rows=1060 width=8) (actual time=29.011..29.012 rows=1017 loo
                                Buckets: 2048  Batches: 1  Memory Usage: 56kB
                                ->  Seq Scan on flights f  (cost=0.00..5309.84 rows=1060 width=8) (actual time=0.025..28.898
                                      Filter: (departure_airport = 'BAX'::bpchar)
                                      Rows Removed by Filter: 213850
                    ->  Index Scan using tickets_pkey on tickets t  (cost=0.43..0.48 rows=1 width=21) (actual time=0.012
                          Index Cond: (ticket_no = tf.ticket_no)
              ->  Index Scan using bookings_pkey on bookings b  (cost=0.43..0.47 rows=1 width=21) (actual time=0.
                    Index Cond: (book_ref = t.book_ref)
Planning Time: 0.589 ms
Execution Time: 2621.764 ms
```

Consulta 2 - Lenta -> C2L

```sql
SELECT
    flights.flight_id,
    count(*) total_flight_id
FROM
    flights
JOIN ticket_flights ON
    flights.flight_id = ticket_flights.flight_id
WHERE
    flights.arrival_airport IN (
    SELECT
        airport_code
    FROM
        airports
    WHERE
        airport_name =
'Barnaul Airport')
    OR
flights.departure_airport IN (
    SELECT
        airport_code
    FROM
        airports
    WHERE
        airport_name =
'Barnaul Airport')
GROUP BY
    flights.flight_id
ORDER BY
    flights.flight_id
```

Plano de Consulta da C2L

```
QUERY PLAN
Sort  (cost=672121.86..672524.73 rows=161150 width=12) (actual time=3220.849..3220.895 rows=1127 loops=1)
  Sort Key: flights.flight_id
  Sort Method: quicksort  Memory: 110kB
  ->  HashAggregate  (cost=604647.87..655429.46 rows=161150 width=12) (actual time=3219.855..3220.629 rows=1127 loops=1)
        Group Key: flights.flight_id
        Planned Partitions: 4  Batches: 1  Memory Usage: 3217kB
        ->  Hash Join  (cost=8552.50..250623.25 rows=6293771 width=4) (actual time=68.481..3196.100 rows=55357 loops=1)
              Hash Cond: (ticket_flights.flight_id = flights.flight_id)
              ->  Seq Scan on ticket_flights  (cost=0.00..153850.08 rows=8391708 width=4) (actual time=0.110..1230.594 rows=8391852 loops=1)
              ->  Hash  (cost=5908.13..5908.13 rows=161150 width=4) (actual time=67.328..67.330 rows=2035 loops=1)
                    Buckets: 262144  Batches: 2  Memory Usage: 2084kB
                    ->  Seq Scan on flights  (cost=61.13..5908.13 rows=161150 width=4) (actual time=0.862..66.323 rows=2035 loops=1)
                          Filter: ((hashed SubPlan 1) OR (hashed SubPlan 2))
                          Rows Removed by Filter: 212832
                          SubPlan 1
                            ->  Seq Scan on airports_data ml  (cost=0.00..30.56 rows=1 width=4) (actual time=0.514..0.519 rows=1 loops=1)
                                  Filter: ((airport_name ->> lang()) = 'Barnaul Airport'::text)
                                  Rows Removed by Filter: 103
                          SubPlan 2
                            ->  Seq Scan on airports_data ml_1  (cost=0.00..30.56 rows=1 width=4) (actual time=0.216..0.221 rows=1 loops=1)
                                  Filter: ((airport_name ->> lang()) = 'Barnaul Airport'::text)
                                  Rows Removed by Filter: 103
Planning Time: 0.459 ms
Execution Time: 3224.326 ms
```

Consulta 2 - Otimizada -> C2O

```sql
WITH airports_filter AS (
    SELECT
        AIRPORT_CODE
    FROM
        BOOKINGS.AIRPORTS
    WHERE
        AIRPORT_NAME = 'Barnaul Airport'
)
SELECT
    f.FLIGHT_ID,
    COUNT(f.FLIGHT_ID)
FROM
    BOOKINGS.FLIGHTS AS f
INNER JOIN BOOKINGS.TICKET_FLIGHTS AS tf ON
    f.FLIGHT_ID = tf.FLIGHT_ID
WHERE
    EXISTS (SELECT * FROM airports_filter WHERE AIRPORT_CODE = f.ARRIVAL_AIRPORT OR AIRPORT_CODE = f.DEPARTURE_AIRPORT)
GROUP BY
    1
ORDER BY
    1
ASC;
```

Plano de Consulta da C20

QUERY PLAN

```
GroupAggregate  (cost=211624.40..214434.95 rows=160603 width=12) (actual time=1840.985..1848.568 rows=1127 loops=1)
  Group Key: f.flight_id
  ->  Sort  (cost=211624.40..212025.91 rows=160603 width=4) (actual time=1840.925..1843.457 rows=55357 loops=1)
        Sort Key: f.flight_id
        Sort Method: quicksort  Memory: 1537kB
        ->  Hash Join  (cost=8614.80..195539.75 rows=160603 width=4) (actual time=96.258..1830.206 rows=55357 loops=1)
              Hash Cond: (tf.flight_id = f.flight_id)
              ->  Seq Scan on ticket_flights tf  (cost=0.00..153850.08 rows=8391708 width=4) (actual time=0.119..957.799 rows=8391852 loops=1)
              ->  Hash  (cost=8563.40..8563.40 rows=4112 width=4) (actual time=95.858..95.860 rows=2035 loops=1)
                    Buckets: 8192  Batches: 1  Memory Usage: 136kB
                    ->  Nested Loop Semi Join  (cost=0.00..8563.40 rows=4112 width=4) (actual time=0.316..95.332 rows=2035 loops=1)
                          Join Filter: ((ml.airport_code = f.arrival_airport) OR (ml.airport_code = f.departure_airport))
                          Rows Removed by Join Filter: 212832
                          ->  Seq Scan on flights f  (cost=0.00..4772.67 rows=214867 width=12) (actual time=0.009..18.710 rows=214867 loops=1)
                          ->  Materialize  (cost=0.00..30.57 rows=1 width=4) (actual time=0.000..0.000 rows=1 loops=214867)
                                ->  Seq Scan on airports_data ml  (cost=0.00..30.56 rows=1 width=4) (actual time=0.240..0.244 rows=1 loops=1)
                                      Filter: ((airport_name ->> lang()) = 'Barnaul Airport'::text)
                                      Rows Removed by Filter: 103
Planning Time: 0.422 ms
Execution Time: 1848.839 ms
```

Consulta 3 - Lenta -> C3L

```sql
SELECT
    bookings.book_ref,
    count(*)
FROM
    bookings,
    tickets,
    ticket_flights,
    flights
WHERE
    tickets.ticket_no = ticket_flights.ticket_no
    AND
flights.status = 'Scheduled'
    AND
ticket_flights.flight_id = flights.flight_id
    AND
bookings.book_ref = tickets.book_ref
GROUP BY
    bookings.book_ref
HAVING
    count(*) > 1
ORDER BY
    bookings.book_ref;
```

Plano de Consulta da C3L

```
Finalize GroupAggregate  (cost=270178.03..341190.06 rows=196161 width=15) (actual time=2318.961..2649.746 rows=111325 loops=1)
  Group Key: bookings.book_ref
  Filter: (count(*) > 1)
  Rows Removed by Filter: 19670
  -> Gather Merge  (cost=270178.03..331166.99 rows=491154 width=15) (actual time=2318.951..2589.386 rows=132995 loops=1)
    Workers Planned: 2
    Workers Launched: 2
    -> Partial GroupAggregate  (cost=269178.01..273675.60 rows=245577 width=15) (actual time=2259.493..2364.055 rows=44332 loops=3)
      Group Key: bookings.book_ref
      -> Sort  (cost=269178.01..269991.95 rows=245577 width=7) (actual time=2259.480..2335.481 rows=146766 loops=3)
        Sort Key: bookings.book_ref
        Sort Method: external merge  Disk: 191280
        Worker 0:  Sort Method: external merge  Disk: 261088
        Worker 1:  Sort Method: external merge  Disk: 271280
        -> Parallel Hash Join  (cost=209632.58..241031.72 rows=245577 width=7) (actual time=1604.937..1778.352 rows=146766 loops=3)
          Hash Cond: (bookings.book_ref = tickets.book_ref)
          -> Parallel Seq Scan on bookings  (cost=0.00..22213.29 rows=679629 width=7) (actual time=0.099..65.555 rows=701701 loops=3)
          -> Parallel Hash  (cost=205602.87..205602.87 rows=245577 width=7) (actual time=1633.721..1633.721 rows=146766 loops=3)
            Buckets: 262144  Batches: 4  Memory Usage: 6976kB
            -> Parallel Hash Join  (cost=122661.64..205602.87 rows=245577 width=7) (actual time=1158.400..1404.643 rows=146766 loops=3)
              Hash Cond: (tickets.ticket_no = ticket_flights.ticket_no)
              -> Parallel Seq Scan on tickets  (cost=0.00..61705.18 rows=1226018 width=21) (actual time=0.230..205.975 rows=983286 loops=3)
              -> Parallel Hash  (cost=118391.93..118391.93 rows=245577 width=14) (actual time=747.085..747.086 rows=146766 loops=3)
                Buckets: 262144  Batches: 8  Memory Usage: 4704kB
                -> Parallel Hash Join  (cost=41311.87..118391.93 rows=245577 width=14) (actual time=11.379..704.013 rows=146766 loops=3)
                  Hash Cond: (ticket_flights.flight_id = flights.flight_id)
                  -> Parallel Seq Scan on ticket_flights  (cost=0.00..104990.45 rows=3196545 width=18) (actual time=0.222..305.772 rows=2797284 loops=3)
                  -> Parallel Hash  (cost=4203.90..4203.90 rows=6877 width=4) (actual time=11.071..11.071 rows=5126 loops=3)
                    Buckets: 16384  Batches: 1  Memory Usage: 736kB
                    -> Parallel Seq Scan on flights  (cost=0.00..4203.90 rows=6877 width=4) (actual time=0.011..29.174 rows=15383 loops=3)
                      Filter: ((status)::text = 'Scheduled'::text)
                      Rows Removed by Filter: 199184
Planning Time: 0.554 ms
Execution Time: 2658.251 ms
```

Consulta 3 - Otimizada -> C3O

```sql
WITH flights_filter AS (
    SELECT
        FLIGHT_ID
    FROM
        BOOKINGS.FLIGHTS
    WHERE
        STATUS = 'Scheduled'
)
SELECT
    b.BOOK_REF,
    COUNT(b.BOOK_REF)
FROM
    BOOKINGS.BOOKINGS AS b
INNER JOIN BOOKINGS.TICKETS AS t ON
    b.BOOK_REF = t.BOOK_REF
INNER JOIN BOOKINGS.TICKET_FLIGHTS AS tf ON
    t.TICKET_NO = tf.TICKET_NO
INNER JOIN flights_filter AS ff ON
    tf.FLIGHT_ID = ff.FLIGHT_ID
GROUP BY
    1
HAVING
    COUNT(b.BOOK_REF) > 1
ORDER BY
    1
ASC;
```

Plano de Consulta da C3O

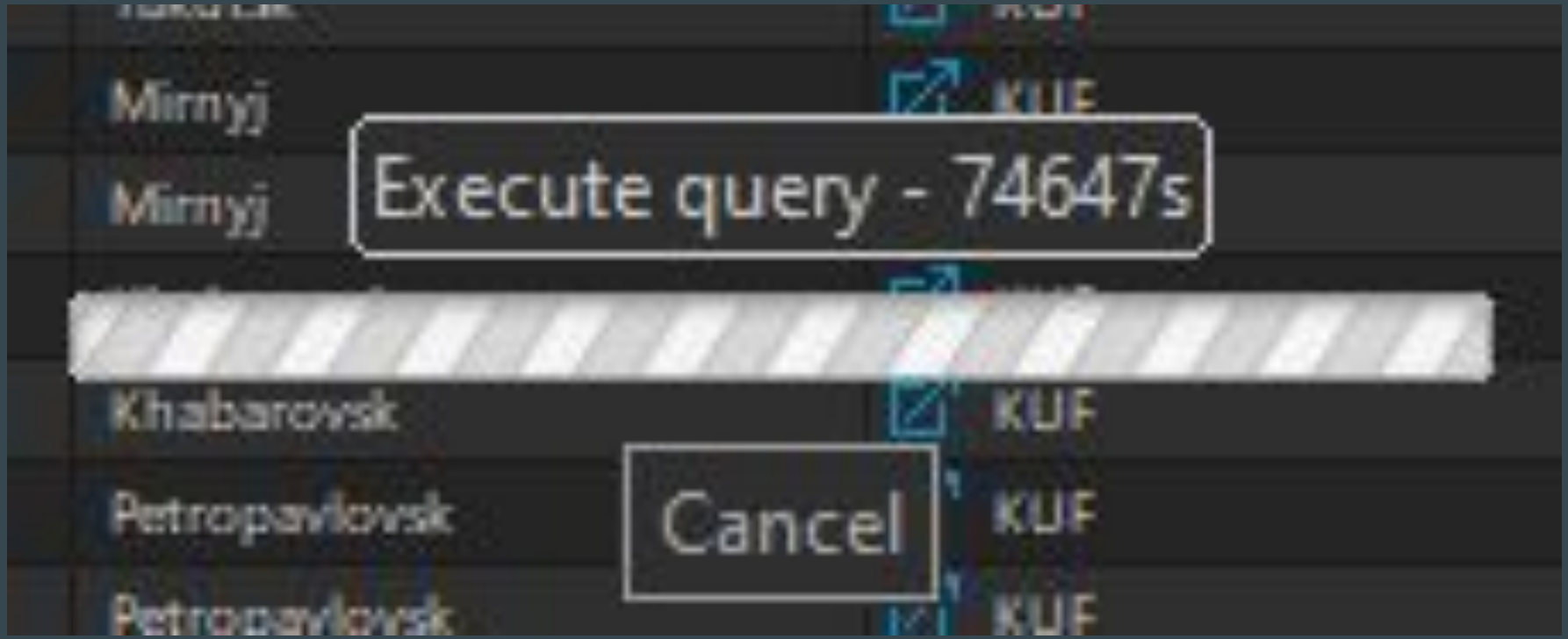
```
Finalize GroupAggregate  (cost=270378.03..341190.06 rows=196161 width=15) (actual time=2194.216..2556.259 rows=113325 loops=1)
  Group Key: b.book_ref
  Filter: (count(b.book_ref) > 1)
  Rows Removed by Filter: 19670
  -> Gather Merge  (cost=270378.03..331366.99 rows=491154 width=15) (actual time=2194.208..2494.692 rows=132995 loops=1)
    Workers Planned: 2
    Workers Launched: 2
    -> Partial GroupAggregate  (cost=269378.01..273675.60 rows=245577 width=15) (actual time=2137.109..2261.910 rows=44332 loops=3)
      Group Key: b.book_ref
      -> Sort  (cost=269378.01..269991.95 rows=245577 width=7) (actual time=2137.091..2213.295 rows=148788 loops=3)
        Sort Key: b.book_ref
        Sort Method: external merge  Disk: 1912KB
        Worker 0:  Sort Method: external merge  Disk: 2720KB
        Worker 1:  Sort Method: external merge  Disk: 2844KB
        -> Parallel Hash Join  (cost=209632.58..244031.72 rows=245577 width=7) (actual time=1526.857..1684.591 rows=148788 loops=3)
          Hash Cond: (b.book_ref = r.book_ref)
          -> Parallel Seq Scan on bookings.b  (cost=0.00..22213.29 rows=879629 width=7) (actual time=0.147..64.866 rows=703703 loops=3)
          -> Parallel Hash  (cost=205602.87..205602.87 rows=245577 width=7) (actual time=1355.088..1355.091 rows=148788 loops=3)
            Buckets: 262144  Batches: 4  Memory Usage: 6464kB
            -> Parallel Hash Join  (cost=122661.64..205602.87 rows=245577 width=7) (actual time=1062.833..1321.429 rows=148788 loops=3)
              Hash Cond: (r.rider_no = rf.rider_no)
              -> Parallel Seq Scan on riders r  (cost=0.00..61705.18 rows=1229018 width=21) (actual time=0.448..177.962 rows=983286 loops=3)
              -> Parallel Hash  (cost=118391.93..118391.93 rows=245577 width=14) (actual time=703.167..703.170 rows=148788 loops=3)
                Buckets: 262144  Batches: 8  Memory Usage: 4704kB
                -> Parallel Hash Join  (cost=41314.87..118391.93 rows=245577 width=14) (actual time=10.949..661.052 rows=148788 loops=3)
                  Hash Cond: (rf.flight_id = flights.flight_id)
                  -> Parallel Seq Scan on rider_flights rf  (cost=0.00..104866.15 rows=3196515 width=16) (actual time=0.237..286.646 rows=2797284 loops=3)
                  -> Parallel Hash  (cost=41203.90..41203.90 rows=8877 width=4) (actual time=10.624..10.624 rows=5128 loops=3)
                    Buckets: 16384  Batches: 1  Memory Usage: 736kB
                    -> Parallel Seq Scan on flights  (cost=0.00..41203.90 rows=8877 width=4) (actual time=0.012..27.586 rows=15361 loops=1)
                      Filter: ((status)::text = 'Scheduled'::text)
                      Rows Removed by Filter: 1996168
Planning Time: 0.614 ms
Execution Time: 2564.746 ms
```

Consulta 4 - Lenta -> C4L

```
SELECT
    f2.flight_no,
    f2.departure_airport,
    dep.airport_name AS departure_airport_name,
    dep.city AS departure_city,
    f2.arrival_airport,
    arr.airport_name AS arrival_airport_name,
    arr.city AS arrival_city,
    f2.aircraft_code,
    (SELECT avg(flights.scheduled_arrival - flights.scheduled_departure)
    FROM bookings.flights WHERE flight_no = f2.flight_no LIMIT 1) duration,
    (SELECT array_agg(DISTINCT to_char(flights.scheduled_departure, 'ID'::text)::integer)
    FROM bookings.flights WHERE flight_no = f2.flight_no LIMIT 1) days_of_week
FROM
    bookings.flights f2 ,
    bookings.airports dep,
    bookings.airports arr
WHERE
    f2.departure_airport = dep.airport_code AND f2.arrival_airport = arr.airport_code
    AND dep.city LIKE '%Petro%' OR arr.city LIKE '%Petro%'
```

Plano de Consulta da C4L

Acabou não finalizando o Explain Analyse



Plano de Consulta da C4L

Mostrando apenas com o explain

QUERY PLAN

```
Nested Loop  (cost=0.00..4562037211.06 rows=3718746 width=195)
  Join Filter: (((f2.departure_airport = ml.airport_code) AND (f2.arrival_airport = ml_1.airport_code) AND ((ml.city ->> lang()) ~~* '%Petro%'::text)) OR ((ml_1.city ->> lang()) ~~* '%Petro%'::text))
  ->  Seq Scan on flights f2  (cost=0.00..4772.67 rows=214867 width=19)
  ->  Materialize  (cost=0.00..197.62 rows=10816 width=228)
        ->  Nested Loop  (cost=0.00..143.54 rows=10816 width=228)
              ->  Seq Scan on airports_data ml  (cost=0.00..4.04 rows=104 width=114)
              ->  Materialize  (cost=0.00..4.56 rows=104 width=114)
                    ->  Seq Scan on airports_data ml_1  (cost=0.00..4.04 rows=104 width=114)
SubPlan 1
  ->  Limit  (cost=879.32..879.33 rows=1 width=16)
        ->  Aggregate  (cost=879.32..879.33 rows=1 width=16)
              ->  Bitmap Heap Scan on flights  (cost=10.77..877.81 rows=303 width=16)
                    Recheck Cond: (flight_no = f2.flight_no)
                    ->  Bitmap Index Scan on flights_flight_no_scheduled_departure_key  (cost=0.00..10.69 rows=303 width=0)
                          Index Cond: (flight_no = f2.flight_no)
SubPlan 2
  ->  Limit  (cost=16.76..16.77 rows=1 width=32)
        ->  Aggregate  (cost=16.76..16.77 rows=1 width=32)
              ->  Index Only Scan using flights_flight_no_scheduled_departure_key on flights flights_1  (cost=0.42..13.72 rows=303 width=8)
                    Index Cond: (flight_no = f2.flight_no)
```

Consulta 4 - Otimizada-> C4O

```sql
WITH hub AS (
    SELECT
        flight_no,
        ARRAY_AGG(DISTINCT to_char(f2.scheduled_departure, 'ID'::text)::integer) AS days_of_week
     FROM
        bookings.flights f2
     GROUP BY
        1
)
SELECT
    f.FLIGHT_ID,
    f.flight_no,
    f.departure_airport,
    a.airport_name AS departure_airport_name,
    a.city AS departure_city,
    f.arrival_airport,
    a.airport_name AS arrival_airport_name,
    a.city AS arrival_city,
    f.aircraft_code,
    f.scheduled_arrival - f.scheduled_departure AS duration,
    h.days_of_week
FROM
    bookings.flights AS f
INNER JOIN bookings.airports AS a ON
    f.DEPARTURE_AIRPORT = a.AIRPORT_CODE
    OR f.ARRIVAL_AIRPORT = a.AIRPORT_CODE
INNER JOIN hub AS h ON
    f.flight_no = h.flight_no
WHERE
    a.CITY LIKE '%Petro%';
```

Plano de Consulta da C4O

```
QUERY PLAN
Hash Join  (cost=8078.06..21542.93 rows=4112 width=199) (actual time=131.175..329.552 rows=4979 loops=1)
  Hash Cond: (f2.flight_no = f.flight_no)
  -> GroupAggregate  (cost=0.42..9242.14 rows=710 width=39) (actual time=1.276..175.071 rows=710 loops=1)
      Group Key: f2.flight_no
      -> Index Only Scan using flights_flight_no_scheduled_departure_key on flights f2  (cost=0.42..6547.43 rows=214867 width=15) (actual time=0.041..33.170 rows=214867 loops=1)
          Heap Fetches: 0
  -> Hash  (cost=8026.24..8026.24 rows=4112 width=149) (actual time=115.379..115.381 rows=4979 loops=1)
      Buckets: 8192  Batches: 1  Memory Usage: 1005kB
      -> Nested Loop  (cost=0.00..8026.24 rows=4112 width=149) (actual time=1.980..113.583 rows=4979 loops=1)
          Join Filter: ((f.departure_airport = ml.airport_code) OR (f.arrival_airport = ml.airport_code))
          Rows Removed by Join Filter: 424755
          -> Seq Scan on airports_data ml  (cost=0.00..30.56 rows=1 width=114) (actual time=0.218..0.477 rows=2 loops=1)
              Filter: ((city ->> lang()) ~~* '%Petro%'::text)
              Rows Removed by Filter: 102
          -> Seq Scan on flights f  (cost=0.00..4772.67 rows=214867 width=39) (actual time=0.004..22.310 rows=214867 loops=2)
Planning Time: 1.073 ms
Execution Time: 333.358 ms
```

Ganhos em Tempo de Execução (MS)

Consulta	Tempo Consulta Normal	Tempo Consulta Otimizada	Diferença
1	2945	2620	325
2	3220	1848	1372
3	2649	2556	93
4	–	329	–

Ganhos em Custo de Execução

Consulta	Custo Consulta Normal	Custo Consulta Otimizada	Diferença
1	224.229	224.229	0
2	672.524	214.434	458.090
3	241.190	241.190	0
4	4.562.037.211	21.542	4.562.015.669

Entendendo a pouca diferença entre as consultas 1 e 3

Para poder otimizar ainda mais as consulta 1 e 3, precisaríamos criar índices nas chaves onde ocorrem os JOINS, porém optei por não modificar a “Estrutura” do banco, mantendo assim sua forma original.

Tomei esta decisão pois, as consultas melhoraram significativamente em questão de tempo de execução.

Link para Consultas

- Consultas Normais: LINK

- Consultas Otimizadas: LINK