Pontifícia Universidade Católica do Rio Grande do Sul

Infraestrutura para Gestão de Dados

Engenharia de Software

Carolina Ferreira e Mateus Caçabuena

Trabalho Prático 1

Porto Alegre

2025

Sumário

1. Introdução

Este relatório tem como objetivo apresentar o desenvolvimento e a otimização de consultas SQL no ambiente Oracle, com foco na melhoria do desempenho por meio da criação de estruturas de acesso, como índices e clusters hash. A atividade inclui tabelas do domínio da aviação comercial, como passageiros, reservas, voos, companhias aéreas e aeroportos.

Inicialmente, foram elaboradas consultas SQL atendendo a diferentes critérios de seleção, junção e ordenação de dados. Em seguida, foi realizada a análise dos planos de execução dessas consultas, identificando possíveis gargalos de desempenho. Com base nessa análise, foram criadas estruturas otimizadas, como índices B-Tree, clusters com acesso por hash e redefinição de consultas, com o intuito de melhorar o tempo de resposta e reduzir o custo computacional.

A criação de um cluster hash foi explorada utilizando tabelas relacionadas por chaves primárias, com o intuito de obter acessos mais eficientes por meio do uso direto da chave de hash. Essa técnica foi aplicada em conjunto com outras estratégias de tuning para maximizar os ganhos de desempenho.

Por fim, são apresentados os resultados obtidos antes e depois da otimização, seguidos de uma reflexão crítica sobre o impacto das técnicas aplicadas e os aprendizados adquiridos durante a atividade.

2. Desenvolvimento do Experimento

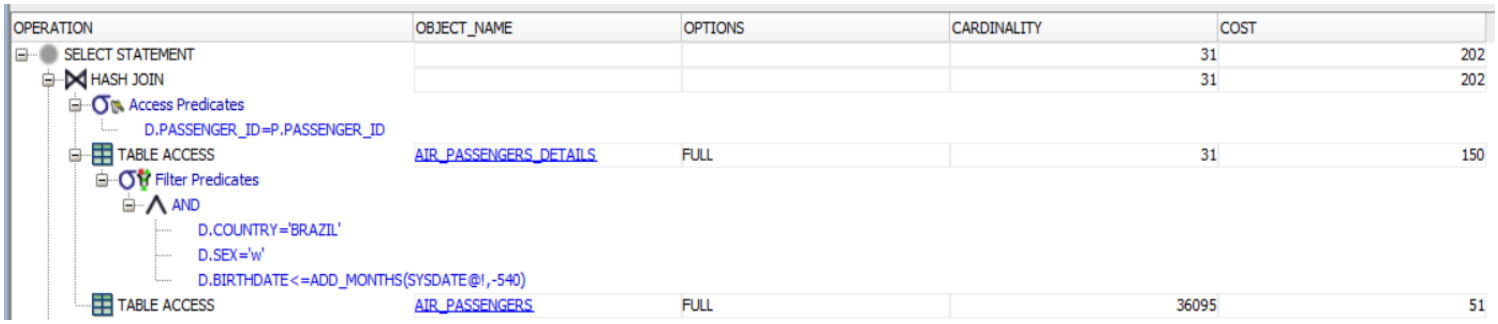
2.1 Consulta A

Listar último nome, a idade e a cidade de todos os passageiros do sexo feminino com mais de 45 anos, residentes no Brasil.

2.1.1 Comando SQL

```
SELECT
    p.LASTNAME,
    TRUNC(MONTHS_BETWEEN(SYSDATE, d.BIRTHDATE) / 12) AS IDADE,
    d.CITY
FROM
    AIR_PASSENGERS_DETAILS d
JOIN
    AIR_PASSENGERS p ON d.PASSENGER_ID = p.PASSENGER_ID
WHERE
    d.sex = 'w'
    AND birthdate <= ADD_MONTHS(SYSDATE, -45*12)
    AND country = 'BRAZIL';
```

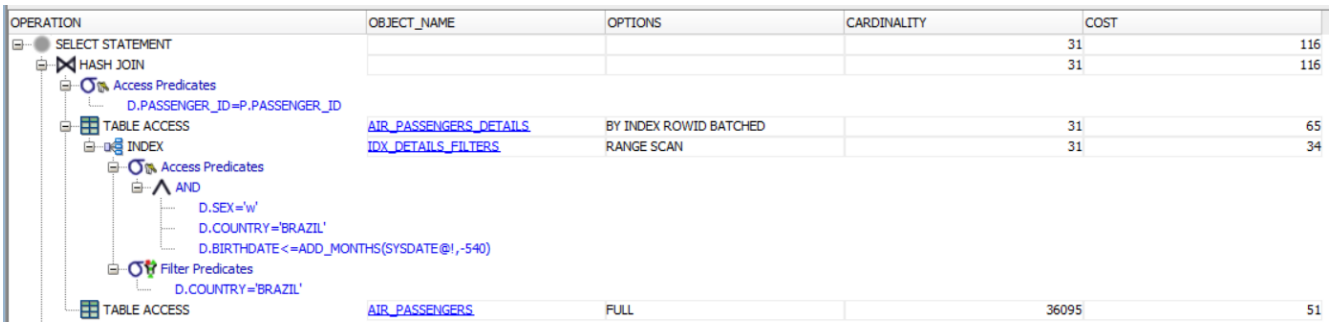
2.1.2 Plano de Execução Antes da Otimização



2.1.3 Estruturas Criadas (Índices/Clusters)

```
CREATE INDEX IDX_DETAILS_FILTERS
ON AIR_PASSENGERS_DETAILS(SEX, BIRTHDATE, COUNTRY);
```

2.1.4 Plano de Execução Após a Otimização

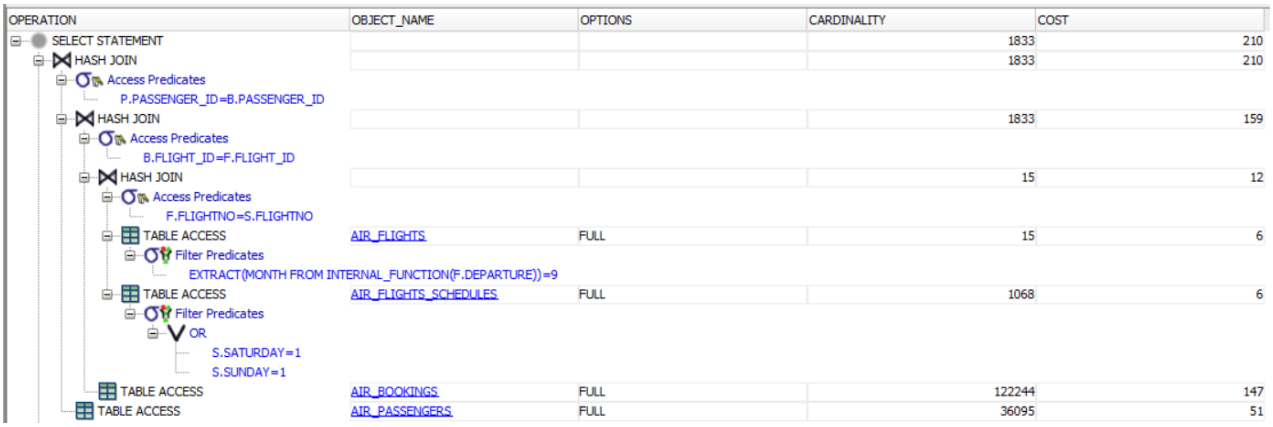


2.2 Consulta B

Listar o nome e sobrenome dos passageiros que fizeram reservas no último mês cujo voos são em finais de semana (sábados e domingos).

2.2.1 Comando SQL

2.2.2 Plano de Execução Antes da Otimização

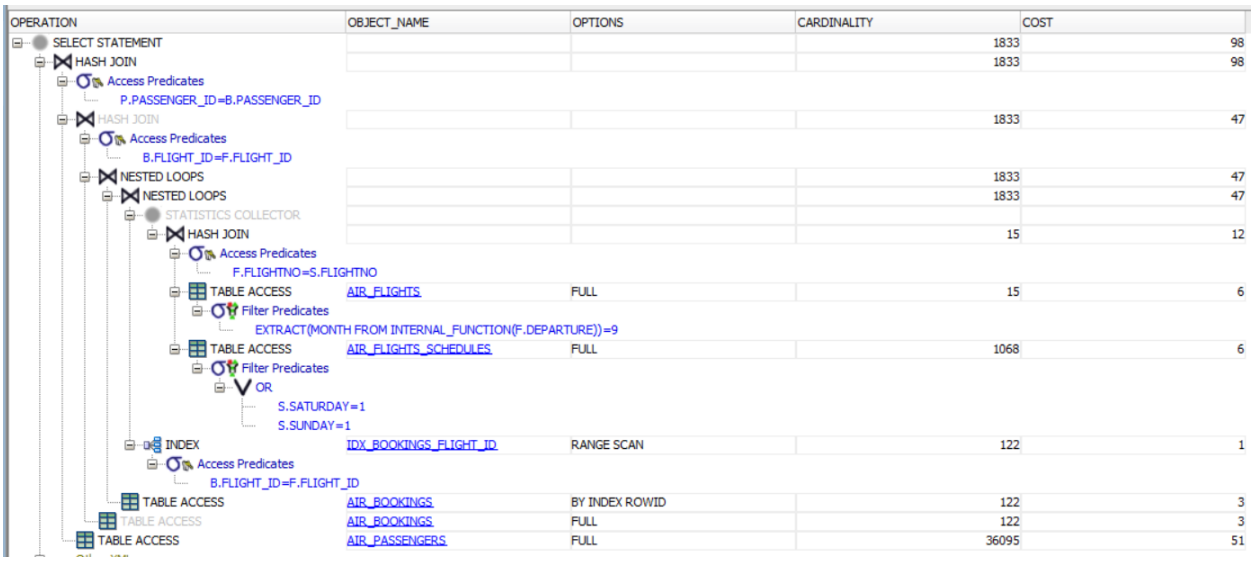


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1833	210
HASH JOIN			1833	210
Access Predicates				
P.PASSENGER_ID=B.PASSENGER_ID				
HASH JOIN			1833	159
Access Predicates				
B.FLIGHT_ID=F.FLIGHT_ID				
HASH JOIN			15	12
Access Predicates				
F.FLIGHTNO=S.FLIGHTNO				
TABLE ACCESS	AIR_FLIGHTS	FULL	15	6
Filter Predicates				
EXTRACT(MONTH FROM INTERNAL_FUNCTION(F.DEPARTURE))=9				
TABLE ACCESS	AIR_FLIGHTS_SCHEDULES	FULL	1068	6
Filter Predicates				
OR				
S.SATURDAY=1				
S.SUNDAY=1				
TABLE ACCESS	AIR_BOOKINGS	FULL	122244	147
TABLE ACCESS	AIR_PASSENGERS	FULL	36095	51

2.2.3 Estruturas Criadas (Índices/Clusters)

```
CREATE INDEX IDX_BOOKINGS_FLIGHT_ID ON AIR_BOOKINGS(FLIGHT_ID);
```

2.2.4 Plano de Execução Após a Otimização



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1833	98
HASH JOIN			1833	98
Access Predicates				
P.PASSENGER_ID=B.PASSENGER_ID				
HASH JOIN			1833	47
Access Predicates				
B.FLIGHT_ID=F.FLIGHT_ID				
NESTED LOOPS			1833	47
NESTED LOOPS			1833	47
STATISTICS COLLECTOR				
HASH JOIN			15	12
Access Predicates				
F.FLIGHTNO=S.FLIGHTNO				
TABLE ACCESS	AIR_FLIGHTS	FULL	15	6
Filter Predicates				
EXTRACT(MONTH FROM INTERNAL_FUNCTION(F.DEPARTURE))=9				
TABLE ACCESS	AIR_FLIGHTS_SCHEDULES	FULL	1068	6
Filter Predicates				
OR				
S.SATURDAY=1				
S.SUNDAY=1				
INDEX	IDX_BOOKINGS_FLIGHT_ID	RANGE SCAN	122	1
Access Predicates				
B.FLIGHT_ID=F.FLIGHT_ID				
TABLE ACCESS	AIR_BOOKINGS	BY INDEX ROWID	122	3
TABLE ACCESS	AIR_BOOKINGS	FULL	122	3
TABLE ACCESS	AIR_PASSENGERS	FULL	36095	51

2.3 Consulta C

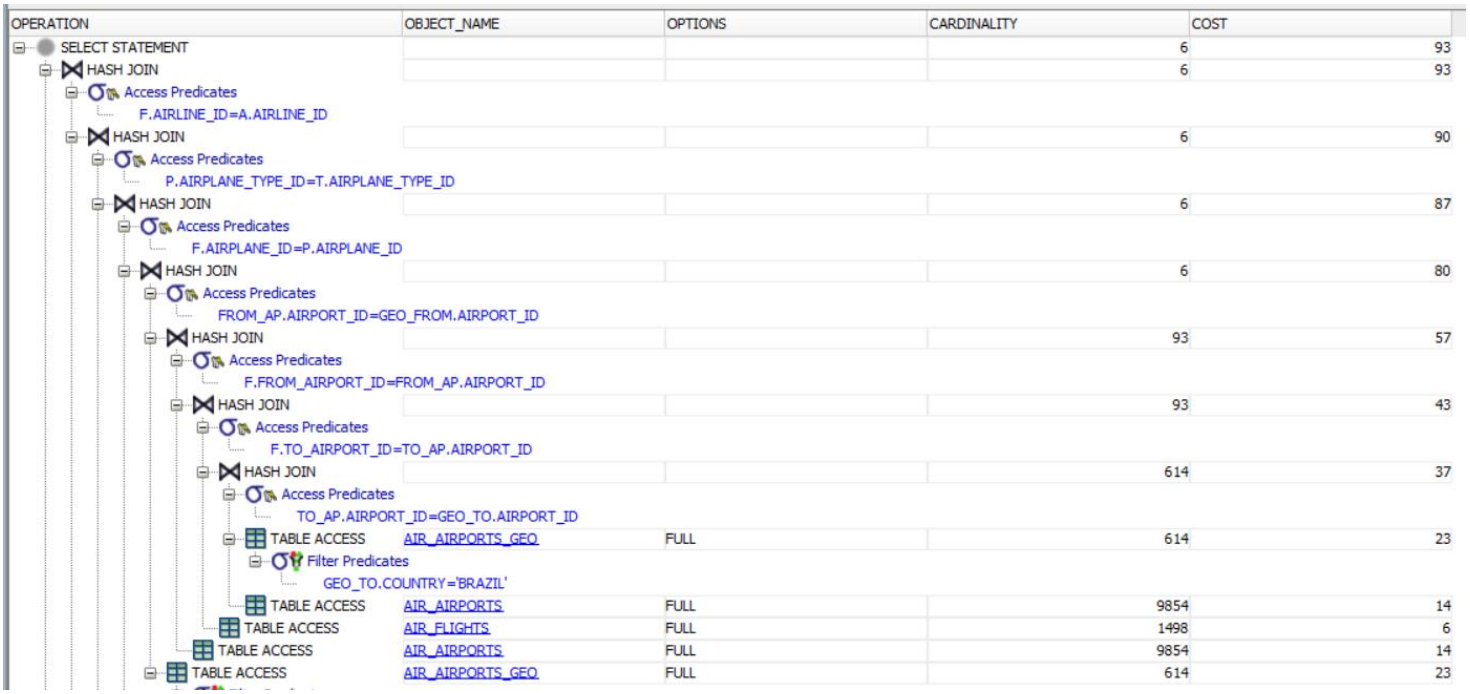
Listar o nome da companhia aérea, o identificador da aeronave, o nome do tipo de aeronave e o número de todos os voos operados por essa companhia aérea (independentemente de a aeronave ser de sua propriedade) que saem E chegam em aeroportos localizados no país 'BRAZIL'.

2.3.1 Comando SQL

```
SELECT
    a.AIRLINE_NAME,
    f.AIRPLANE_ID,
    t.NAME AS AIRPLANE_TYPE,
    f.FLIGHTNO
FROM
    AIR_FLIGHTS f
JOIN
    AIR_AIRLINES a ON f.AIRLINE_ID = a.AIRLINE_ID
JOIN
    AIR_AIRPLANES p ON f.AIRPLANE_ID = p.AIRPLANE_ID
JOIN
    AIR_AIRPLANE_TYPES t ON p.AIRPLANE_TYPE_ID =
t.AIRPLANE_TYPE_ID
JOIN
    AIR_AIRPORTS from_ap ON f.FROM_AIRPORT_ID =
from_ap.AIRPORT_ID
JOIN
    AIR_AIRPORTS to_ap ON f.TO_AIRPORT_ID = to_ap.AIRPORT_ID
JOIN
    AIR_AIRPORTS_GEO geo_from ON from_ap.AIRPORT_ID =
geo_from.AIRPORT_ID
JOIN
```

```
    AIR_AIRPORTS_GEO geo_to ON to_ap.AIRPORT_ID =
geo_to.AIRPORT_ID
WHERE
    geo_from.COUNTRY = 'BRAZIL'
    AND geo_to.COUNTRY = 'BRAZIL';
```

2.3.2 Plano de Execução Antes da Otimização

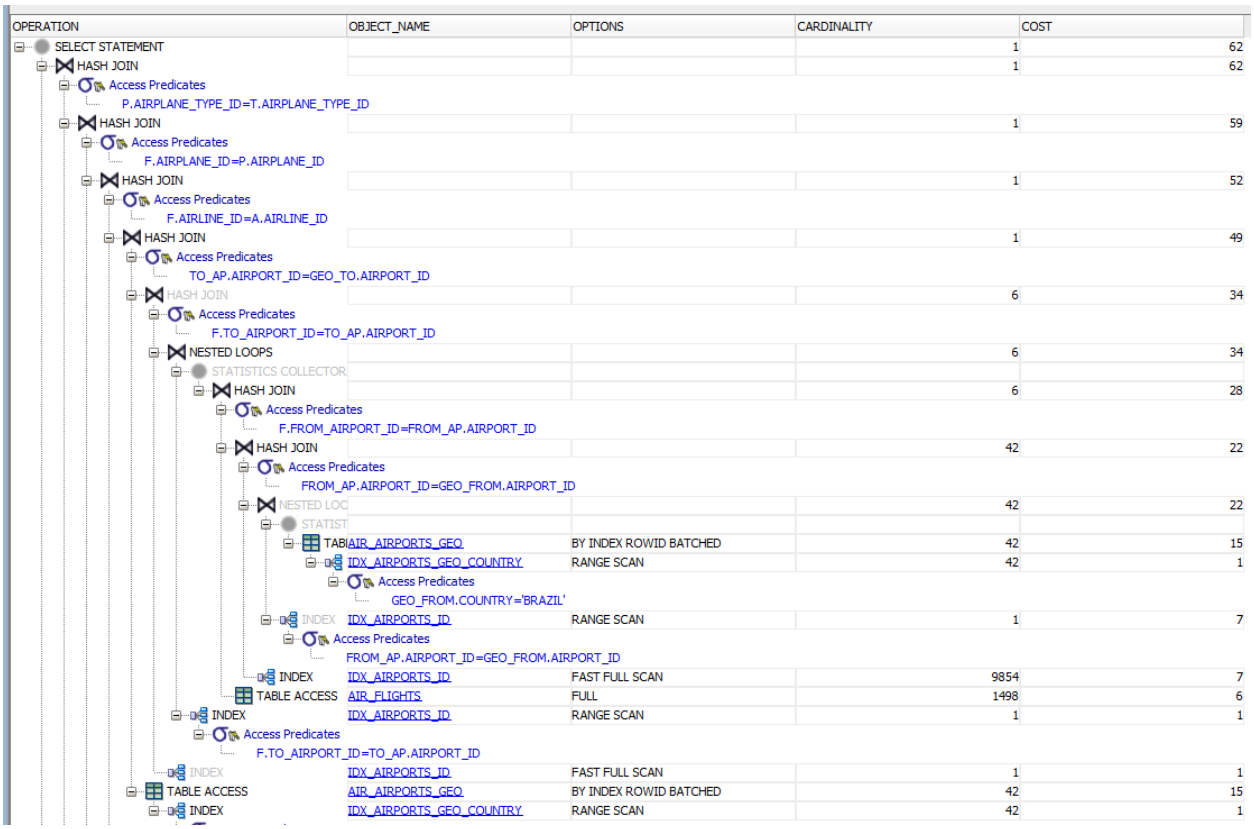


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			6	93
HASH JOIN			6	93
Access Predicates				
F.AIRLINE_ID=A.AIRLINE_ID				
HASH JOIN			6	90
Access Predicates				
P.AIRPLANE_TYPE_ID=T.AIRPLANE_TYPE_ID				
HASH JOIN			6	87
Access Predicates				
F.AIRPLANE_ID=P.AIRPLANE_ID				
HASH JOIN			6	80
Access Predicates				
FROM_AP.AIRPORT_ID=GEO_FROM.AIRPORT_ID				
HASH JOIN			93	57
Access Predicates				
F.FROM_AIRPORT_ID=FROM_AP.AIRPORT_ID				
HASH JOIN			93	43
Access Predicates				
F.TO_AIRPORT_ID=TO_AP.AIRPORT_ID				
HASH JOIN			614	37
Access Predicates				
TO_AP.AIRPORT_ID=GEO_TO.AIRPORT_ID				
TABLE ACCESS	AIR_AIRPORTS_GEO	FULL	614	23
Filter Predicates				
GEO_TO.COUNTRY='BRAZIL'				
TABLE ACCESS	AIR_AIRPORTS	FULL	9854	14
TABLE ACCESS	AIR_FLIGHTS	FULL	1498	6
TABLE ACCESS	AIR_AIRPORTS	FULL	9854	14
TABLE ACCESS	AIR_AIRPORTS_GEO	FULL	614	23

2.3.3 Estruturas Criadas (Índices/Clusters)

```
CREATE INDEX IDX_FLIGHTS_AIRPLANE_ID ON AIR_FLIGHTS(AIRPLANE_ID);
CREATE INDEX IDX_AIRPLANES_AIRLINE_ID ON
AIR_AIRPLANES(AIRLINE_ID);
CREATE INDEX IDX_AIRPORTS_ID ON AIR_AIRPORTS(AIRPORT_ID);
CREATE INDEX IDX_AIRPORTS_GEO_COUNTRY ON
AIR_AIRPORTS_GEO(COUNTRY);
```

2.3.4 Plano de Execução Após a Otimização



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	62
HASH JOIN			1	62
Access Predicates				
P.AIRPLANE_TYPE_ID=T.AIRPLANE_TYPE_ID				
HASH JOIN			1	59
Access Predicates				
F.AIRPLANE_ID=P.AIRPLANE_ID				
HASH JOIN			1	52
Access Predicates				
F.AIRLINE_ID=A.AIRLINE_ID				
HASH JOIN			1	49
Access Predicates				
TO_AP.AIRPORT_ID=GEO_TO.AIRPORT_ID				
HASH JOIN			6	34
Access Predicates				
F.TO_AIRPORT_ID=TO_AP.AIRPORT_ID				
NESTED LOOPS			6	34
STATISTICS COLLECTOR				
HASH JOIN			6	28
Access Predicates				
F.FROM_AIRPORT_ID=FROM_AP.AIRPORT_ID				
HASH JOIN			42	22
Access Predicates				
FROM_AP.AIRPORT_ID=GEO_FROM.AIRPORT_ID				
NESTED LOC			42	22
STATIST				
TABL	AIR_AIRPORTS_GEO	BY INDEX ROWID BATCHED	42	15
	IDX_AIRPORTS_GEO_COUNTRY	RANGE SCAN	42	1
Access Predicates				
GEO_FROM.COUNTRY='BRAZIL'				
INDEX	IDX_AIRPORTS_ID	RANGE SCAN	1	7
Access Predicates				
FROM_AP.AIRPORT_ID=GEO_FROM.AIRPORT_ID				
INDEX	IDX_AIRPORTS_ID	FAST FULL SCAN	9854	7
TABLE ACCESS	AIR_FLIGHTS	FULL	1498	6
INDEX	IDX_AIRPORTS_ID	RANGE SCAN	1	1
Access Predicates				
F.TO_AIRPORT_ID=TO_AP.AIRPORT_ID				
INDEX	IDX_AIRPORTS_ID	FAST FULL SCAN	1	1
TABLE ACCESS	AIR_AIRPORTS_GEO	BY INDEX ROWID BATCHED	42	15
INDEX	IDX_AIRPORTS_GEO_COUNTRY	RANGE SCAN	42	1

2.4 Consulta D

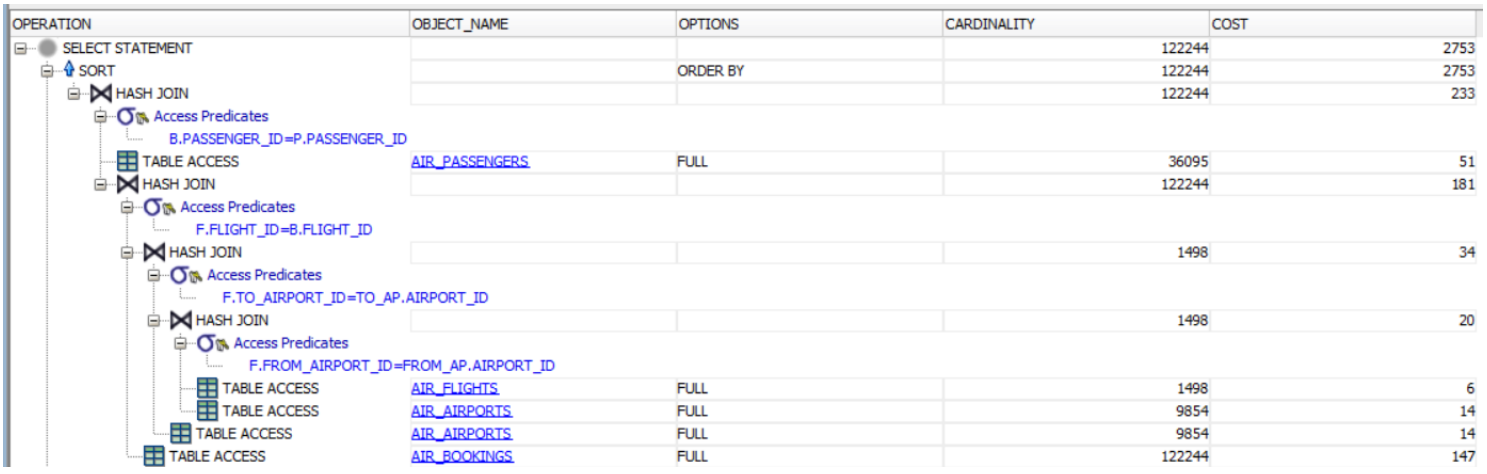
Listar o número do voo, o nome do aeroporto de saída e o nome do aeroporto de destino, o nome completo e o assento de cada passageiro ordenado por voo e nome do passageiro.

2.4.1 Comando SQL

```
SELECT
    f.FLIGHTNO,
    from_ap.NAME AS DEPARTURE_AIRPORT,
    to_ap.NAME AS DESTINATION_AIRPORT,
```

```
    p.FIRSTNAME,
    p.LASTNAME,
    b.SEAT
FROM
    AIR_FLIGHTS f
JOIN
    AIR_AIRPORTS from_ap ON f.FROM_AIRPORT_ID =
from_ap.AIRPORT_ID
JOIN
    AIR_AIRPORTS to_ap ON f.TO_AIRPORT_ID = to_ap.AIRPORT_ID
JOIN
    AIR_BOOKINGS b ON f.FLIGHT_ID = b.FLIGHT_ID
JOIN
    AIR_PASSENGERS p ON b.PASSENGER_ID = p.PASSENGER_ID
ORDER BY
    f.FLIGHTNO, p.FIRSTNAME;
```

2.4.2 Plano de Execução Antes da Otimização



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			122244	2753
SORT		ORDER BY	122244	2753
HASH JOIN			122244	233
Access Predicates				
B.PASSENGER_ID=P.PASSENGER_ID				
TABLE ACCESS	AIR_PASSENGERS	FULL	36095	51
HASH JOIN			122244	181
Access Predicates				
F.FLIGHT_ID=B.FLIGHT_ID				
HASH JOIN			1498	34
Access Predicates				
F.TO_AIRPORT_ID=TO_AP.AIRPORT_ID				
HASH JOIN			1498	20
Access Predicates				
F.FROM_AIRPORT_ID=FROM_AP.AIRPORT_ID				
TABLE ACCESS	AIR_FLIGHTS	FULL	1498	6
TABLE ACCESS	AIR_AIRPORTS	FULL	9854	14
TABLE ACCESS	AIR_AIRPORTS	FULL	9854	14
TABLE ACCESS	AIR_BOOKINGS	FULL	122244	147

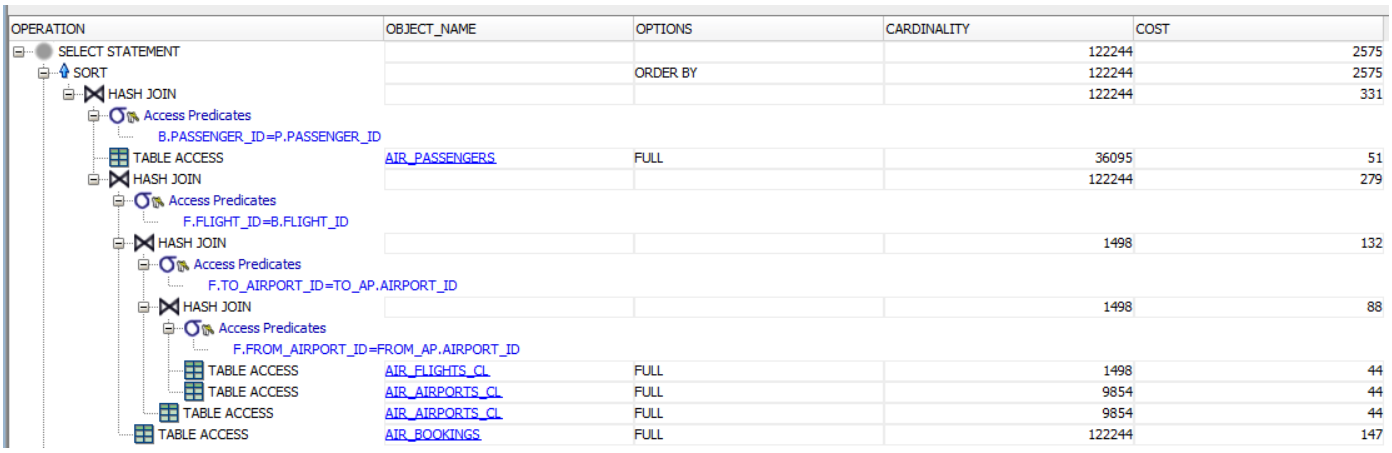
2.4.3 Estruturas Criadas (Índices/Clusters)

```
CREATE CLUSTER CL_FLIGHT_AIRPORTS (AIRPORT_ID NUMBER(5))
HASHKEYS 1000
SIZE 1024;

CREATE TABLE AIR_AIRPORTS_CL
  CLUSTER CL_FLIGHT_AIRPORTS (AIRPORT_ID)
  AS SELECT * FROM AIR_AIRPORTS;

CREATE TABLE AIR_FLIGHTS_CL
  CLUSTER CL_FLIGHT_AIRPORTS (FROM_AIRPORT_ID)
  AS SELECT * FROM AIR_FLIGHTS;
```

2.4.4 Plano de Execução Após a Otimização



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			122244	2575
SORT		ORDER BY	122244	2575
HASH JOIN			122244	331
Access Predicates				
B.PASSENGER_ID=P.PASSENGER_ID				
TABLE ACCESS	AIR_PASSENGERS	FULL	36095	51
HASH JOIN			122244	279
Access Predicates				
F.FLIGHT_ID=B.FLIGHT_ID				
HASH JOIN			1498	132
Access Predicates				
F.TO_AIRPORT_ID=TO_AP.AIRPORT_ID				
HASH JOIN			1498	88
Access Predicates				
F.FROM_AIRPORT_ID=FROM_AP.AIRPORT_ID				
TABLE ACCESS	AIR_FLIGHTS_CL	FULL	1498	44
TABLE ACCESS	AIR_AIRPORTS_CL	FULL	9854	44
TABLE ACCESS	AIR_AIRPORTS_CL	FULL	9854	44
TABLE ACCESS	AIR_BOOKINGS	FULL	122244	147

2.5 Consulta E

Crie uma consulta que seja resolvida adequadamente com um acesso *hash* em um *cluster* com pelo menos duas tabelas. A consulta deve utilizar todas as tabelas do *cluster* e pelo menos outra tabela fora dele.

2.5.1 Comando SQL

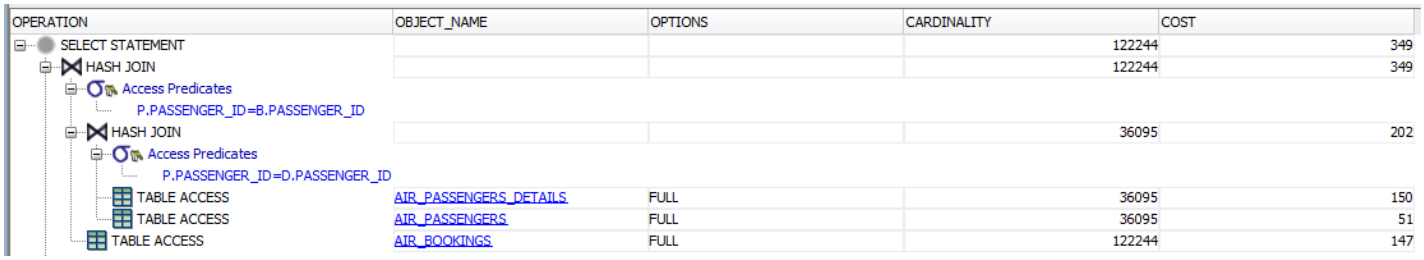
```
CREATE CLUSTER CL_PASSENGERS (PASSENGER_ID NUMBER(12))
HASHKEYS 1000
SIZE 1024;

CREATE TABLE AIR_PASSENGERS_CL CLUSTER CL_PASSENGERS(PASSENGER_ID)
AS SELECT * FROM ACAMPOS.AIR_PASSENGERS;

CREATE TABLE AIR_PASSENGERS_DETAILS_CL CLUSTER
CL_PASSENGERS(PASSENGER_ID)
AS SELECT * FROM ACAMPOS.AIR_PASSENGERS_DETAILS;

SELECT
    p.FIRSTNAME,
    p.LASTNAME,
    d.BIRTHDATE,
    b.BOOKING_ID
FROM
    AIR_PASSENGERS_CL p
JOIN
    AIR_PASSENGERS_DETAILS_CL d ON p.PASSENGER_ID = d.PASSENGER_ID
JOIN
    AIR_BOOKINGS b ON p.PASSENGER_ID = b.PASSENGER_ID;
```

2.5.2 Plano de Execução Antes da Otimização



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			122244	349
HASH JOIN			122244	349
Access Predicates				
P.PASSENGER_ID=B.PASSENGER_ID				
HASH JOIN			36095	202
Access Predicates				
P.PASSENGER_ID=D.PASSENGER_ID				
TABLE ACCESS	AIR_PASSENGERS_DETAILS	FULL	36095	150
TABLE ACCESS	AIR_PASSENGERS	FULL	36095	51
TABLE ACCESS	AIR_BOOKINGS	FULL	122244	147

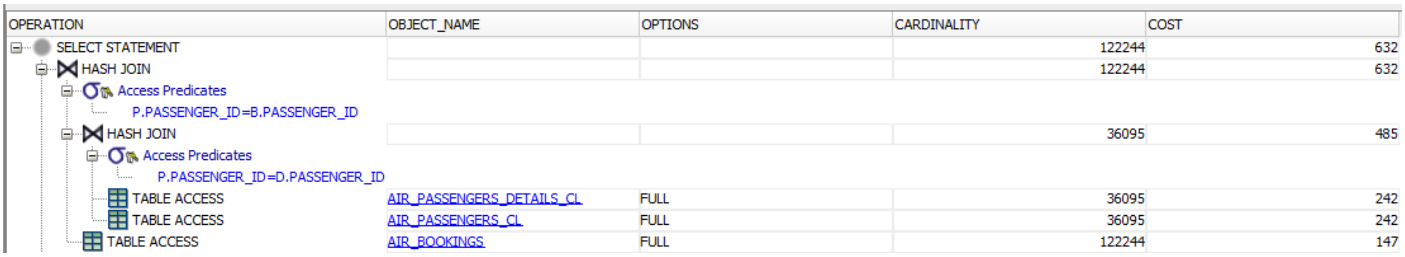
2.5.3 Estruturas Criadas (Índices/Clusters)

```
CREATE CLUSTER CL_PASSENGERS (PASSENGER_ID NUMBER(12))
HASHKEYS 1000
SIZE 1024;

CREATE TABLE AIR_PASSENGERS_CL CLUSTER
CL_PASSENGERS(PASSENGER_ID)
AS SELECT * FROM ACAMPOS.AIR_PASSENGERS;

CREATE TABLE AIR_PASSENGERS_DETAILS_CL CLUSTER
CL_PASSENGERS(PASSENGER_ID)
AS SELECT * FROM ACAMPOS.AIR_PASSENGERS_DETAILS;
```

2.5.4 Plano de Execução Após a Otimização



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			122244	632
HASH JOIN			122244	632
Access Predicates				
P.PASSENGER_ID=B.PASSENGER_ID				
HASH JOIN			36095	485
Access Predicates				
P.PASSENGER_ID=D.PASSENGER_ID				
TABLE ACCESS	AIR_PASSENGERS_DETAILS_CL	FULL	36095	242
TABLE ACCESS	AIR_PASSENGERS_CL	FULL	36095	242
TABLE ACCESS	AIR_BOOKINGS	FULL	122244	147

3. Análise do Experimento

A realização deste experimento permitiu compreender, na prática, como diferentes estratégias de otimização impactam o desempenho de consultas em bancos de dados relacionais. Ao longo da análise das consultas A até E, observou-se que a criação de índices compostos e específicos para os filtros mais seletivos, como nas consultas A e B, foi extremamente eficaz na redução de custos, evidenciando o papel dos índices B-Tree na melhoria do desempenho em operações de seleção.

Nas consultas mais complexas, como a C e a D, envolvendo múltiplas junções, a adoção de índices direcionados às colunas de junção, bem como o uso de clusters, demonstrou um ganho considerável na leitura dos dados. Ficou evidente que a eficácia da otimização depende fortemente do padrão de acesso à informação: a simples criação de um cluster não garante melhora de desempenho, como observado na consulta E. Apesar do uso de um cluster hash entre as tabelas AIR_PASSENGERS_CL e AIR_PASSENGERS_DETAILS_CL, o custo total da consulta aumentou. Isso mostra que a estratégia de clusterização deve ser cuidadosamente avaliada conforme a frequência e a forma de acesso às tabelas envolvidas.

Outra reflexão importante é a escolha das colunas para indexação ou clusterização. Em algumas situações, poderiam ser criados outros índices, por exemplo, individuais para COUNTRY em AIR_PASSENGERS_DETAILS ou para AIRLINE_ID em AIR_FLIGHTS, mas após executá-los, foi concluído que o custo seria maior em comparação com as otimizações já aplicadas.

Por fim, o experimento reforçou a importância de analisar os planos de execução antes de tomar decisões de tuning. A otimização deve ser sempre guiada por evidências e pelo contexto específico da carga de trabalho. A experiência prática consolidou o entendimento de que otimizar um banco de dados é tanto um exercício técnico quanto estratégico, e que cada escolha tem efeitos distintos, muitas vezes não óbvios, sobre o desempenho final.

4. Conclusão

O trabalho permitiu entender melhor como otimizar consultas em bancos de dados usando índices e clusters. Foi possível ver, na prática, que essas estruturas ajudam a melhorar o desempenho das consultas, mas que cada caso precisa ser analisado com cuidado. Nem sempre criar um índice ou cluster vai trazer melhorias, como aconteceu na Consulta E. A atividade foi importante para aprender a interpretar planos de execução e fazer escolhas mais eficientes ao lidar com grandes volumes de dados.