Banco de dados

Trabalho Prático 1

Alunos

Alana Oliveira - 2508818
Anna Hellen- 2508826
Rodrigo Fries - 2419190
Thiago Giebmeyer - 2534070

Introdução

- População do banco de dados;
  - https://sqldatagenerator.com/generator;
  - https://www.mockaroo.com;
- Código para execução das consultas 100 vezes;
- Criação das consultas;
- Tabelas para armazenar os dados de cada consulta;
- Índices utilizados:
  - hash
  - b-tree
  - brin
  - gin
  - multi-colunas

Função - EXPLAIN ANALYSE

```
CREATE OR REPLACE FUNCTION
get_explain_analyze_result_JSON(query_text TEXT)
RETURNS JSON AS $$
DECLARE
    result_text JSON;
BEGIN
--Executa o EXPLAIN ANALYZE
    EXECUTE 'EXPLAIN (ANALYZE, format json) ' ||
query_text INTO result_text;
--Retorna o resultado
    RETURN result_text;
END;
$$ LANGUAGE plpgsql;
```

```
DO $$
DECLARE
  V_SQL TEXT; --Variável que vai armazenar a SQL de consulta
  V_RESULT_JSON JSON; --Resultado do explain analyse
  V_TIME FLOAT; --Tempo de execução da consulta
  V_COST FLOAT; --Custo de recursos para executar a consulta
BEGIN
--Coloca a consulta na variável pra executar
  V_SQL := 'consulta'

--executar 100x
  for i in 1..100 loop
    SELECT get_explain_analyze_result_JSON(V_SQL) into
V_RESULT_JSON;
    V_TIME := CAST(((V_RESULT_JSON::jsonb)-> 0 -> 'Execution
Time') AS FLOAT);
    V_COST := CAST(((V_RESULT_JSON::jsonb)-> 0 -> 'Plan' ->
'Total Cost') AS FLOAT);

    INSERT INTO tbl_time_consult1 VALUES(i, V_TIME, V_COST);
--insere o tempo e o custo na tbl_time_consult1
  END LOOP;
END $$;
```

Índices

Índices: Hash

Os índices hash armazenam um código hash derivado do valor da coluna indexada. Portanto, esses índices só podem lidar com comparações de igualdade simples. O planejador de consultas considera o uso de um índice hash sempre que uma coluna indexada estiver envolvida em uma comparação usando o operador igual:

Índices: B-Tree

As árvores-B podem lidar com consultas de igualdade e de intervalo em dados que podem ser classificados por alguma ordenação. Em particular, o planejador de consultas do PostgreSQL vai considerar o uso de um índice árvore-B sempre que uma coluna indexada estiver envolvida em uma comparação usando um desses operadores:

Índices: Multi-Colunas

Apenas índices do tipo B-tree, GiST, GIN e BRIN possuem suporte para índices multi-colunas por exigirem classificação ordenada dos dados.

CREATE INDEX index_name ON table_name(a, b, c)

Considerara:
WHERE a = v1 and b = v2 and c = v3 || a = v1 and b = v2 || a = v1;

Não considerara:
WHERE c = v3 || b = v2 and c = v3;

Índices: GIN

Os índices GIN são apropriados para valores de dados que contém vários valores componentes, como datas.

GIN pode dar suporte a muitas estratégias de indexação definidas pelo usuário, e os operadores específicos com os quais um índice GIN pode ser usado variam dependendo da estratégia de indexação. Como exemplo, a distribuição padrão do PostgreSQL inclui uma classe de operador GIN para matrizes, que oferece suporte a consultas indexadas usando esses operadores:


```
CREATE EXTENSION IF NOT EXISTS pg_trgm;

CREATE INDEX idx_gin_data_pedido ON corrida USING gin (placa
gin_trgm_ops);
```

Índices: BRIN

Os índices BRIN (acrônimo para Block Range INdexes), armazenam resumos sobre os valores armazenados em intervalos, exemplo: correlação entre data e posição de blocos físicos consecutivos de uma tabela. Para tipos de dados que possuem uma ordem de classificação linear, os dados indexados correspondem aos valores mínimo e máximo dos valores na coluna para cada intervalo de bloco. Esse índice oferece suporte a consultas indexadas usando esses operadores:

C1 - Consultas SQL - Aplicação dos índices

```sql
SELECT C.NOME AS NOMECLIENTE,
  T.MARCA,
  T.MODELO,
  Z.ZONA,
  R.DATAPEDIDO
FROM CLIENTE C
JOIN CORRIDA R ON C.CLIID = R.CLIID
JOIN TAXI T ON R.PLACA = T.PLACA
JOIN MOTORISTA M ON T.PLACA = M.PLACA
JOIN FILA F ON M.CNH = F.CNH
JOIN ZONA Z ON F.ZONA = Z.ZONA
WHERE R.DATAPEDIDO BETWEEN
    (SELECT R.DATAPEDIDO
      FROM CLIENTE C
      JOIN CORRIDA R ON C.CLIID = R.CLIID
      JOIN TAXI T ON R.PLACA = T.PLACA
      JOIN MOTORISTA M ON T.PLACA = M.PLACA
      JOIN FILA F ON M.CNH = F.CNH
      JOIN ZONA Z ON F.ZONA = Z.ZONA
      ORDER BY R.DATAPEDIDO ASC
      LIMIT 1) AND NOW()
  AND T.MODELO like '%i%';
```

```sql
--Indice Hash
CREATE INDEX idx_hash_zona ON Zona USING hash(Zona);

--Indice Btree
CREATE INDEX idx_btree_cliente_cpf ON Cliente USING
BTREE(CPF);

--Indice multi colunas
create index idx_cliId_placa on corrida(cliId, placa);

--Indice BRIN
create index idx_brin_data_pedido on corrida using
brin(datapedido);

--Indice GIN + extensão pg_trgm define a classe de
operadores necessária para índices GIN em strings.
CREATE EXTENSION IF NOT EXISTS pg_trgm;

CREATE INDEX idx_gin_data_pedido  ON corrida USING gin
(placa gin_trgm_ops);
```

C1 - Resultados

Análise

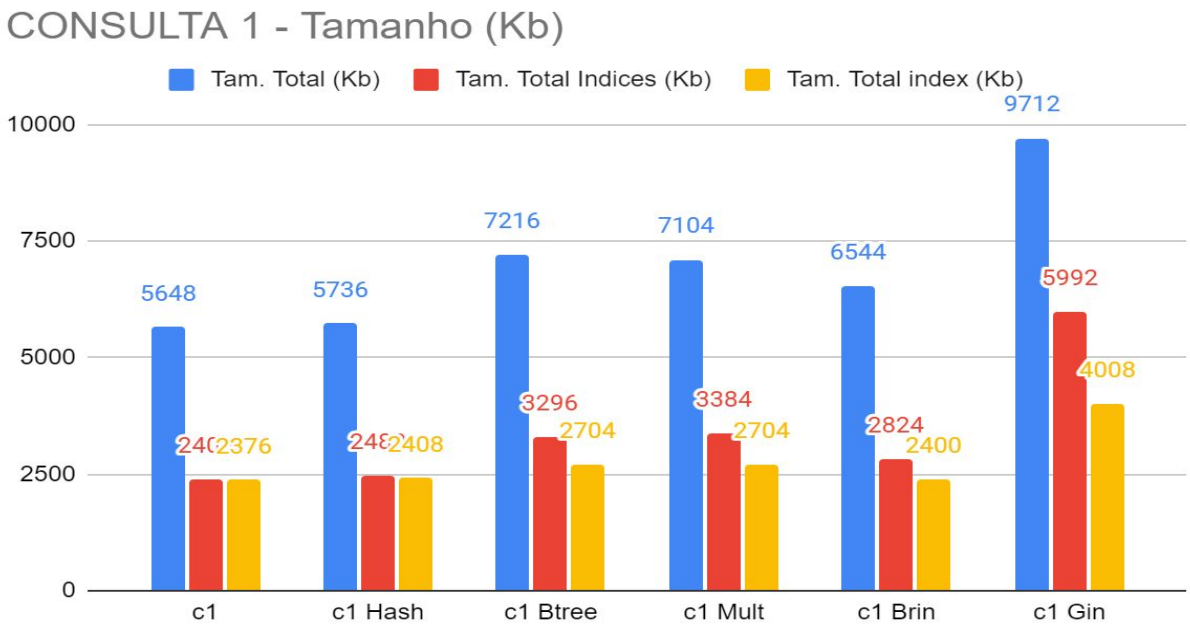
Menor tamanho:

- Hash;

Maior tamanho:

- Gin;

Observações:



C1 - Resultados

Análise

Menor tempo:

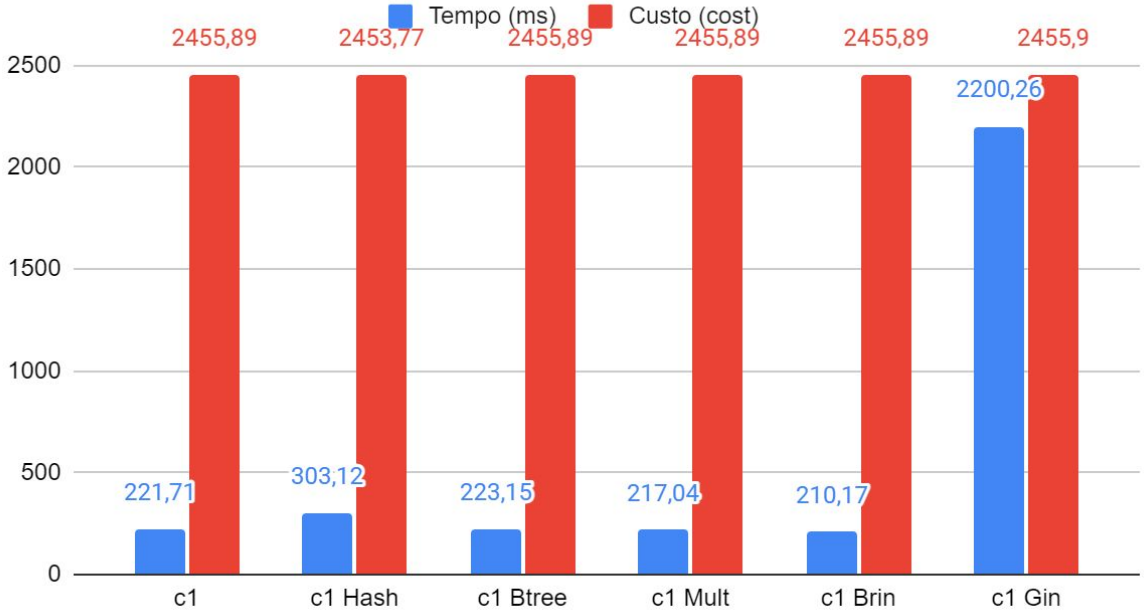
- Brin;

Maior tempo:

- Gin;

Observações:

Custos semelhantes



C2 - Consultas SQL - Aplicação dos índices

Retorna APENAS o nome de cada cliente que possuir o total de corridas maior que 1 e entre '1/1/2000' a '1/1/2023'.

```sql
SELECT
   C.NOME AS NOMECLIENTE,
   COUNT(*) AS TOTALCORRIDAS
FROM
   CLIENTE C
   JOIN CORRIDA R ON C.CLIID = R.CLIID
   JOIN TAXI T ON R.PLACA = T.PLACA
   JOIN MOTORISTA M ON T.PLACA = M.PLACA
   JOIN FILA F ON M.CNH = F.CNH
   JOIN ZONA Z ON F.ZONA = Z.ZONA
WHERE
   R.DATAPEDIDO BETWEEN '1/1/2000' AND
'1/1/2023'
GROUP BY
   C.CLIID
HAVING
   COUNT(*) > 1;
```

```sql
--Indice Hash
CREATE INDEX idx_hash_corrida_cliid ON Corrida
USING hash(CliId);

--Indice Btree
CREATE INDEX idx_btree_cliente_nome ON Cliente
USING BTREE(Nome);

--Indice multi colunas
CREATE INDEX idx_cliId_placa ON corrida(cliId,
placa);

--Indice BRIN
CREATE INDEX idx_brin_data_pedido ON corrida
USING brin(datapedido);

--Indice GIN + extensão pg_trgm define a classe
de operadores necessária para índices GIN em
strings.
CREATE EXTENSION IF NOT EXISTS pg_trgm;

CREATE INDEX idx_gin_placa  ON corrida USING
gin(placa gin_trgm_ops);
```

C2 - Resultados

Análise

Menor tempo:

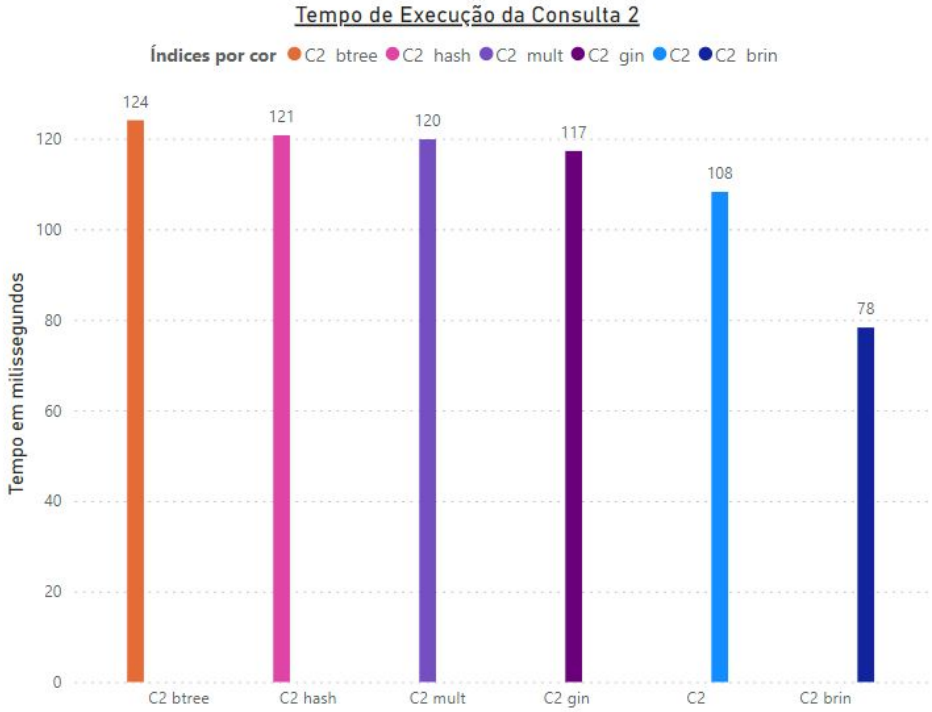
- Brin;

Maior tempo:

- B-tree;

Observações:

O Brin tem melhor performance em consultas com dados ordenados.



C2 - Resultados

Análise

Menor custo:

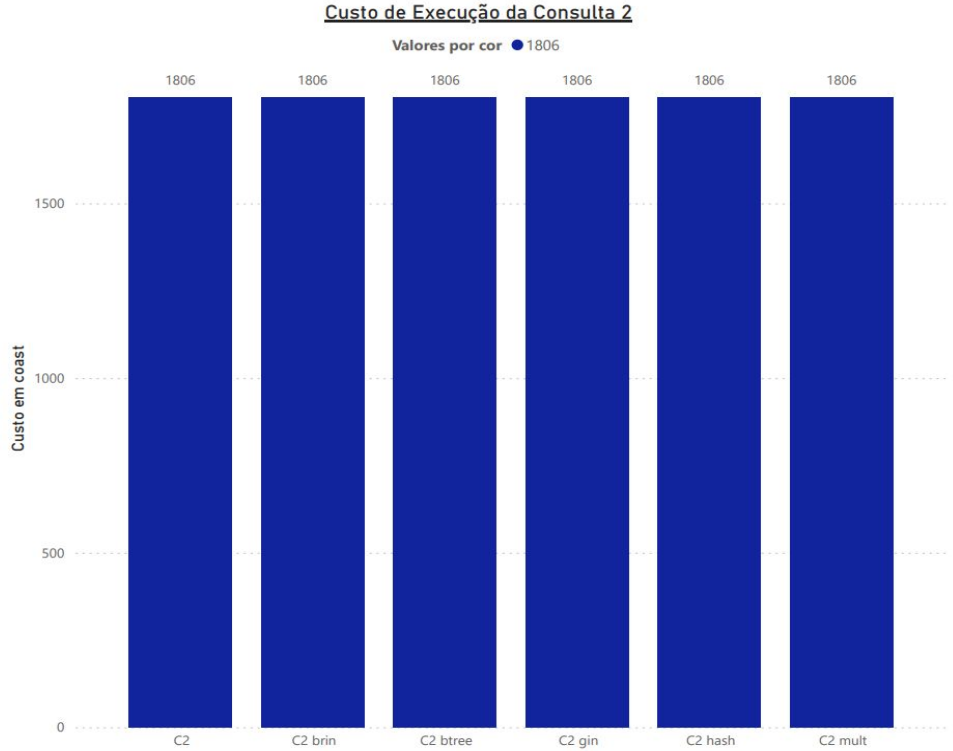
- Todos;

Maior custo:

- Todos;

Observações:

Os índices possuem custos idênticos, porém suas demais características são diferentes.



C2 - Resultados

Análise

Menor tamanho:

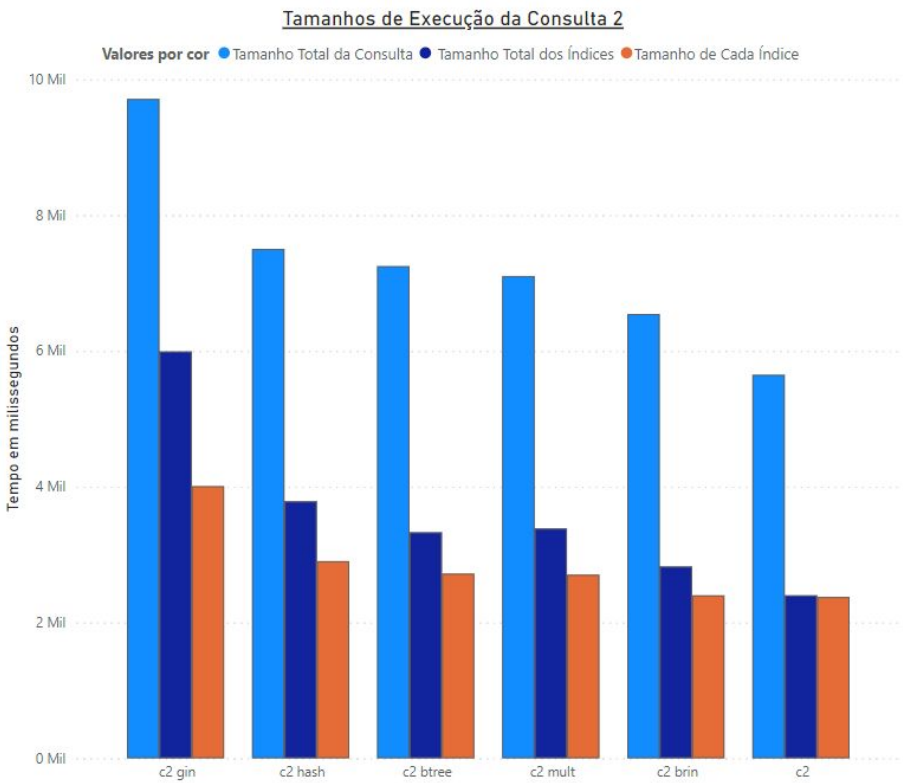
- Sem índice;

Maior tamanho:

- Gin;

Observações:

O Gin é um índice próprio para matrizes de dados.
Armazena informações adicionais de cada item indexado.



C3 - Consultas SQL - Aplicação dos índices

```sql
SELECT NOMEMOTORISTA,
   ZONA,
   MEDIAKM
FROM
   (SELECT M.NOME AS NOMEMOTORISTA,
           Z.ZONA,
           AVG(F.KMIN) AS MEDIAKM,
           COUNT(*) AS CONTAGEMOCORRENCIAS
       FROM MOTORISTA M
       JOIN TAXI T ON M.PLACA = T.PLACA
       JOIN CORRIDA R ON T.PLACA = R.PLACA
       JOIN CLIENTE C ON R.CLIID = C.CLIID
       JOIN FILA F ON M.CNH = F.CNH
       JOIN ZONA Z ON F.ZONA = Z.ZONA
       GROUP BY M.NOME,
           Z.ZONA
       HAVING COUNT(*) > 1) AS SUBCONSULTA
WHERE MEDIAKM % 2 = 0;
```

```sql
--Indice Hash
CREATE INDEX idx_hash_motorista_cnh ON Motorista USING
hash(CNH);

--Indice Btree
CREATE INDEX idx_btree_zona_zona ON Zona USING BTREE(Zona);

--Indice multi colunas
create index idx_cliId_placa on corrida(cliId, placa);

--Indice BRIN
CREATE INDEX idx_brin_placa ON MOTORISTA USING brin (PLACA);

--Indice GIN + extensão pg_trgm define a classe de operadores
necessária para índices GIN em strings.
CREATE EXTENSION IF NOT EXISTS pg_trgm;

CREATE INDEX idx_gin_placa ON corrida USING gin (placa
gin_trgm_ops);
```

C3 - Resultados

Análise

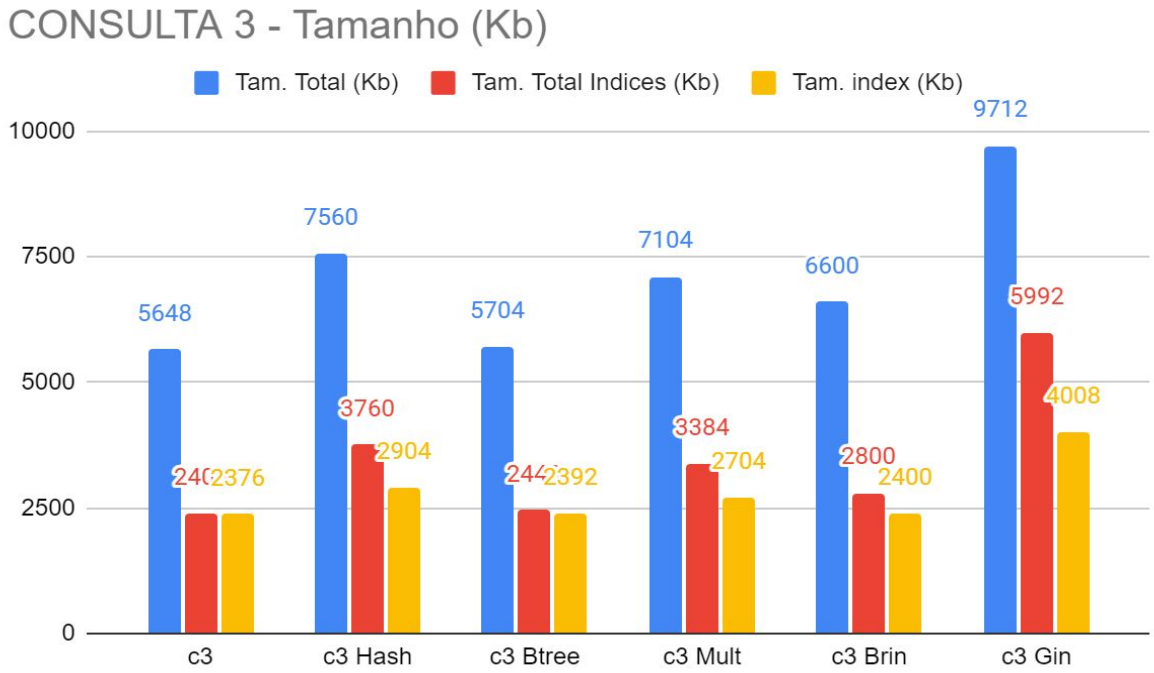
Menor tamanho:

- B-Tree;

Maior tamanho:

- Gin;

Observações:



C3 - Resultados

Análise

Menor tempo:

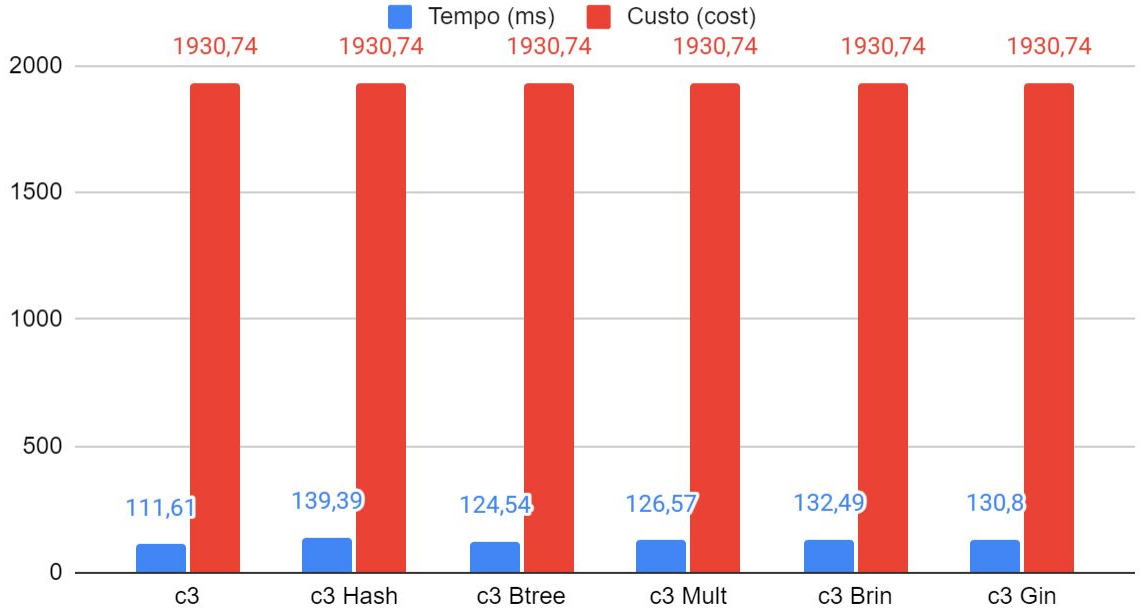
- B-Tree;

Maior tempo:

- Hash;

Observações:

Custo identico para todos os tipos de index



C4 - Consultas SQL - Aplicação dos índices

```sql
SELECT
    t.Placa,
    t.Marca,
    t.Modelo,
    m.Nome AS NomeMotorista,
    m.CNH
FROM
    Taxi t
JOIN
    Corrida r ON t.Placa = r.Placa
JOIN
    Motorista m ON t.Placa = m.Placa
WHERE
    EXISTS (
        SELECT 1
        FROM
            Fila f
        WHERE
            f.CNH = m.CNH
            AND f.Zona = 'Unicamp'
    )
    AND m.CNHValid = 1;
```

```sql
--Indice Hash
CREATE INDEX idx_hash_motorista_cnh ON Motorista USING
hash(CNH);

--Indice Btree
CREATE INDEX idx_btree_taxi_placa ON Taxi USING
BTREE(Placa);

--Indice multi colunas
CREATE INDEX idx_cliId_placa ON corrida(cliId, placa);

--Indice BRIN
CREATE INDEX idx_brin_cnh_zona ON Fila USING brin (CNH,
Zona);

--Indice GIN + extensão pg_trgm define a classe de
operadores necessária para índices GIN em strings.
CREATE EXTENSION IF NOT EXISTS pg_trgm;

CREATE INDEX idx_gin_placa ON corrida USING gin (placa
gin_trgm_ops);
```

C4 - Resultados

Análise

Menor tamanho:

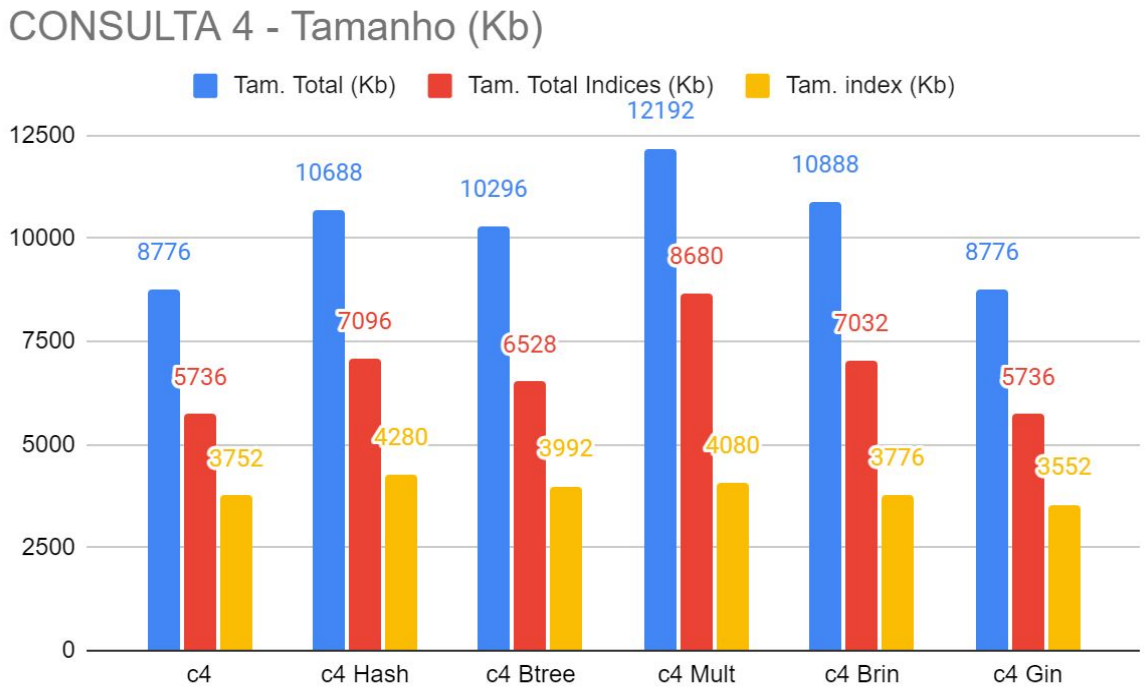
- Mult;

Maior tamanho:

- Gin;

Observações:

A criação do índice GIN não afetou o tamanho original da tabela



C4 - Resultados

Análise

Menor tempo:

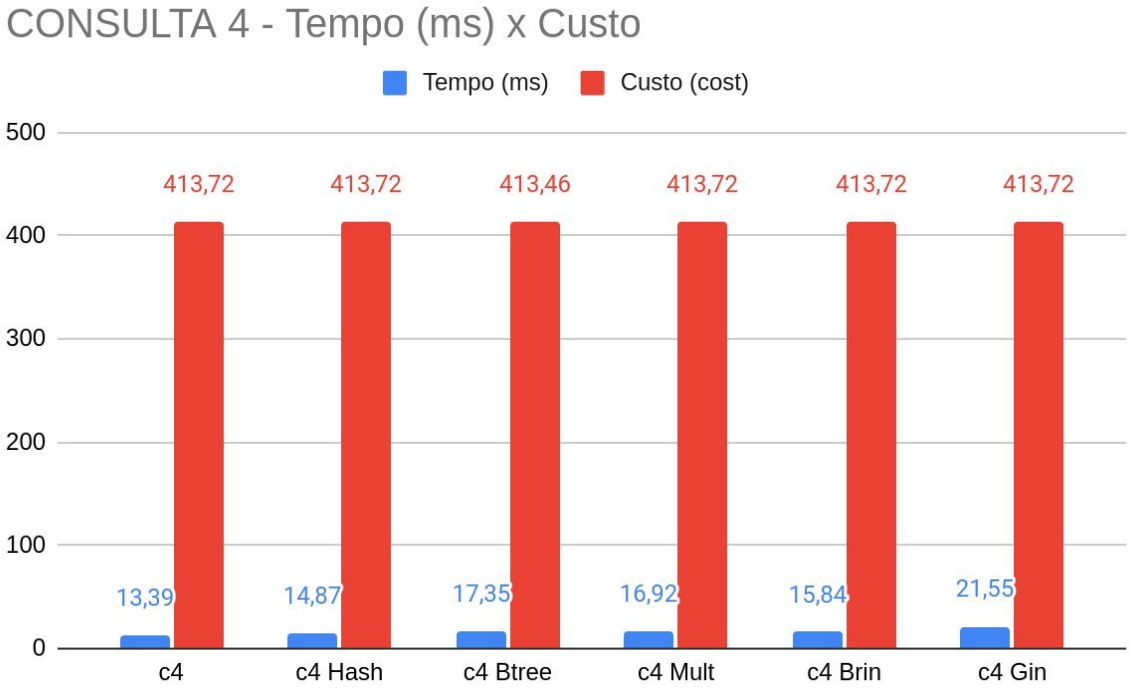
- Sem índice;

Maior tempo:

- Gin;

Observações:

Custo semelhantes



C5 - Consultas SQL - Aplicação dos índices

```sql
SELECT
   C.NOME AS NOMECLIENTE, C.CPF, CO.DATAPEDIDO,
   T.PLACA,T.MARCA AS MARCATAXI,
   T.MODELO AS MODELOTAXI,
   T.ANOFAB AS ANOFABRICACAO, M.CNH,
   M.NOME AS NOMEMOTORISTA,
   Z.ZONA AS ZONAINICIOCORRIDA,
   F.DATAHORAIN AS HORAINICIOCORRIDA,
   F.DATAHORAOUT AS HORAFIMCORRIDA, F.KMIN
FROM CORRIDA CO
   JOIN CLIENTE C ON CO.CLIID = C.CLIID
   JOIN TAXI T ON CO.PLACA = T.PLACA
   JOIN MOTORISTA M ON T.PLACA = M.PLACA
   JOIN FILA F ON M.CNH = F.CNH
   JOIN ZONA Z ON F.ZONA = Z.ZONA
WHERE C.NOME LIKE '%i%'
   AND M.CNH IN (SELECT CNH
       FROM MOTORISTA WHERE CNHVALID = 1)
   AND F.DATAHORAIN BETWEEN '01-01-1999' AND '31-12-2023'
   AND CO.DATAPEDIDO > CURRENT_DATE - INTERVAL '12 month'
ORDER BY
   F.DATAHORAIN DESC;
```

```sql
--Indice Hash
CREATE INDEX idx_hash_corrida_cliid ON Corrida USING
hash(CliId);

--Indice Btree
CREATE INDEX idx_btree_taxi_placa ON Taxi USING
BTREE(Placa);

--Indice multi colunas
CREATE INDEX idx_cliId_placa ON corrida(cliId, placa);

--Indice BRIN fila
CREATE INDEX idx_brin_datahora ON fila USING
brin(DataHoraIn)

--Indice GIN + extensão pg_trgm define a classe de
operadores necessária para índices GIN em strings.
CREATE EXTENSION IF NOT EXISTS pg_trgm;

CREATE INDEX idx_gin_placa ON corrida USING gin (placa
gin_trgm_ops);
```

C5 - Resultados

Análise

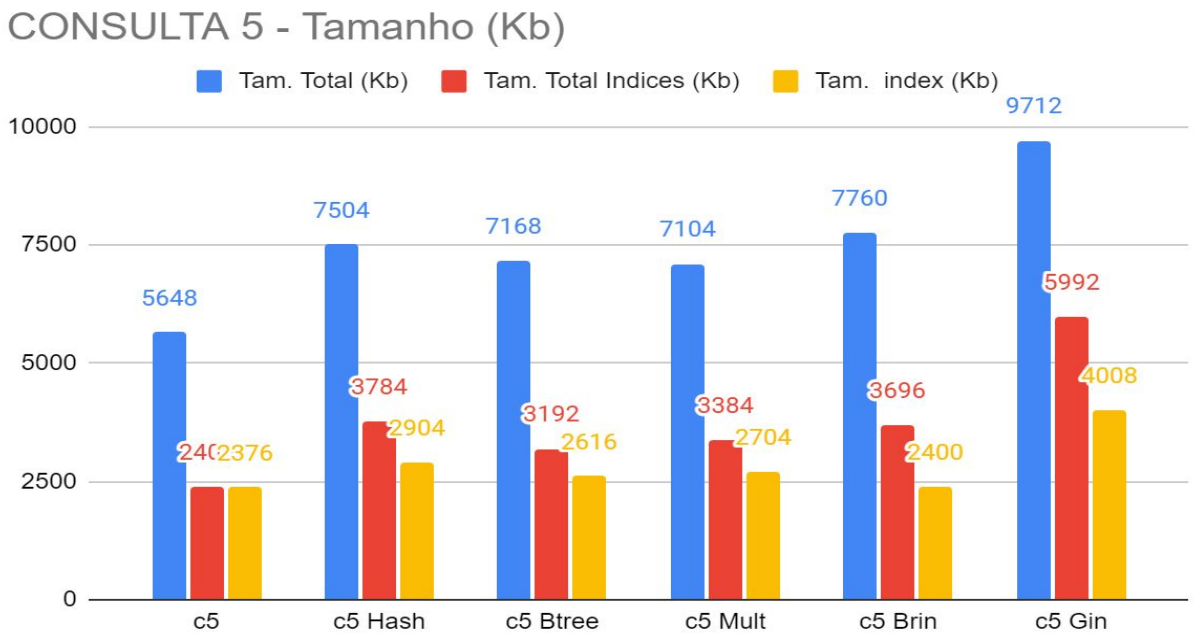
Menor tamanho:

- Gin;

Maior tamanho:

- Sem índice;

Observações: Em todos os índices o tempo de execução foi maior do que na consulta sem índice.



C5 - Resultados

Análise

Menor tempo:

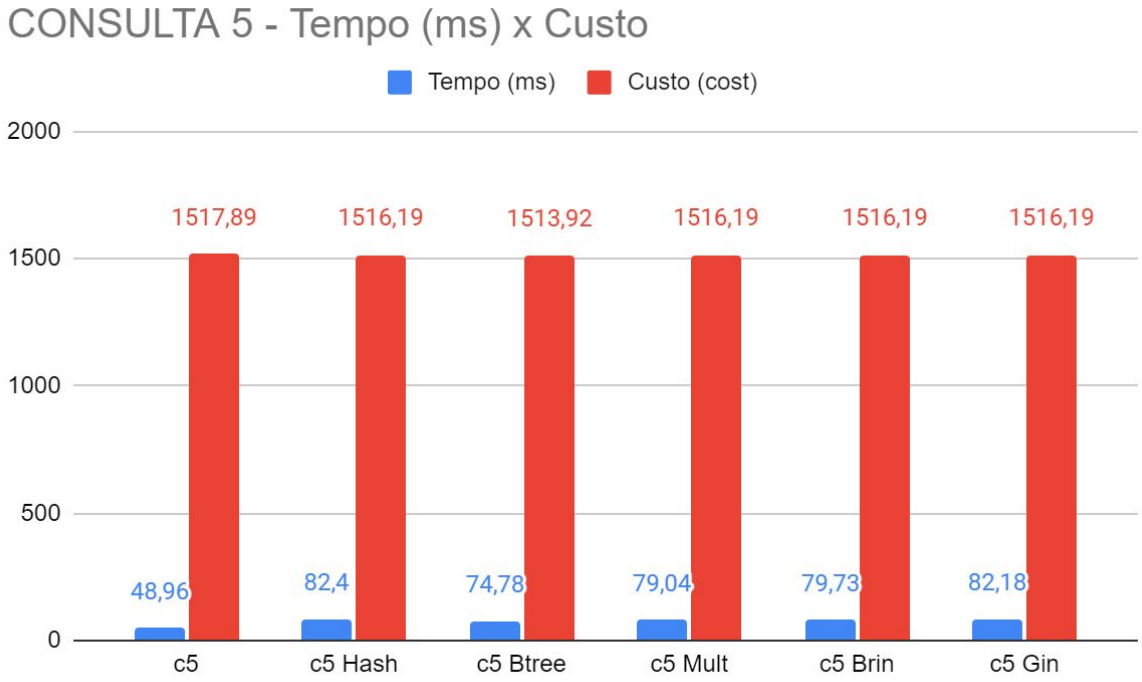
- Sem índice

Maior tempo:

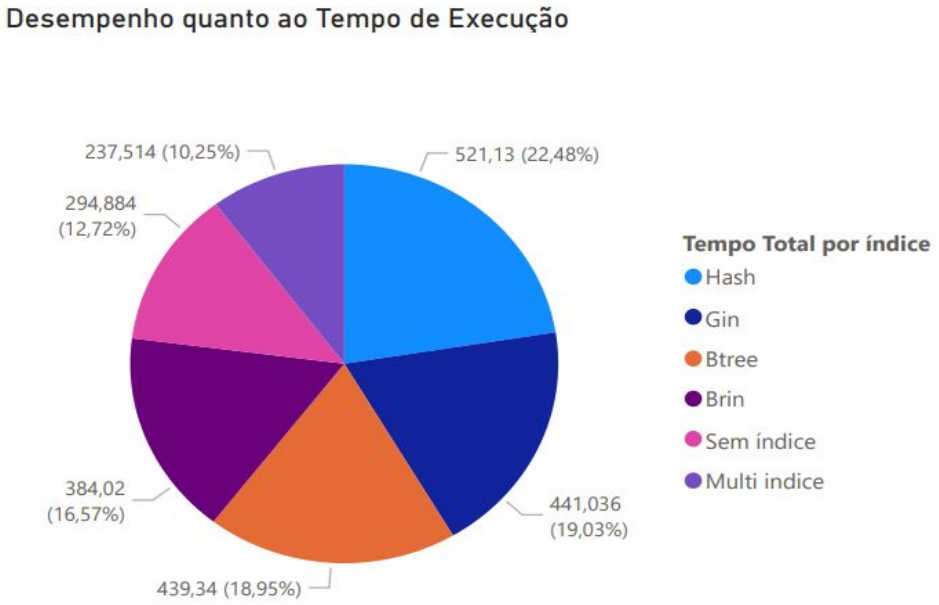
- Hash

Observações:

Custo semelhantes



Desempenho Total



Análise

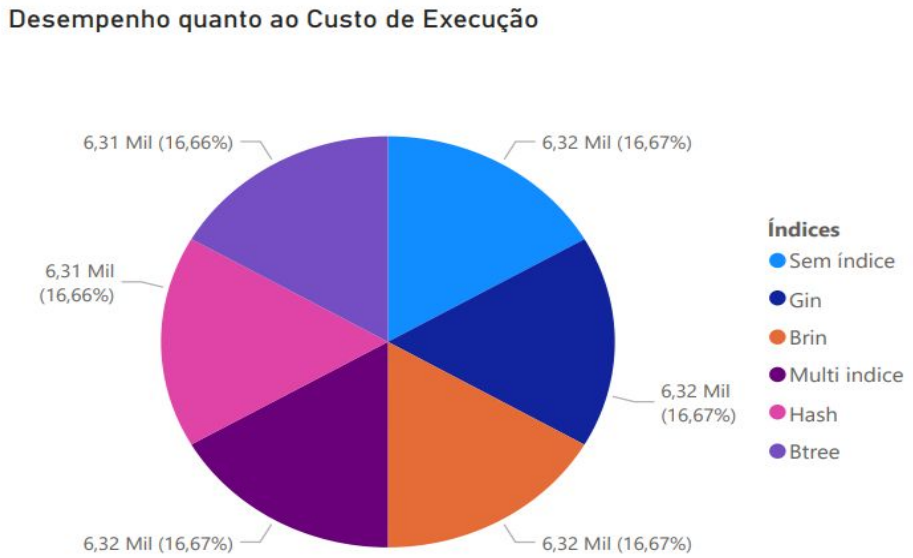
Menor tempo:

- Multi colunas

Maior tempo:

- Hash

Desempenho Total



Análise

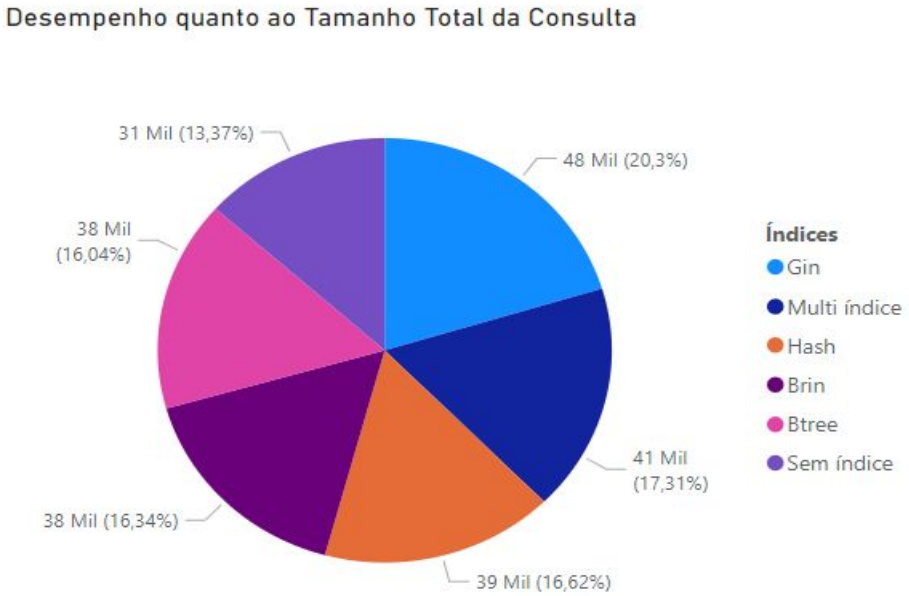
Menor custo:

- Hash;
- Btree;

Maior custo:

- Gin;
- Sem índice;
- Multi colunas;

Desempenho Total



Análise

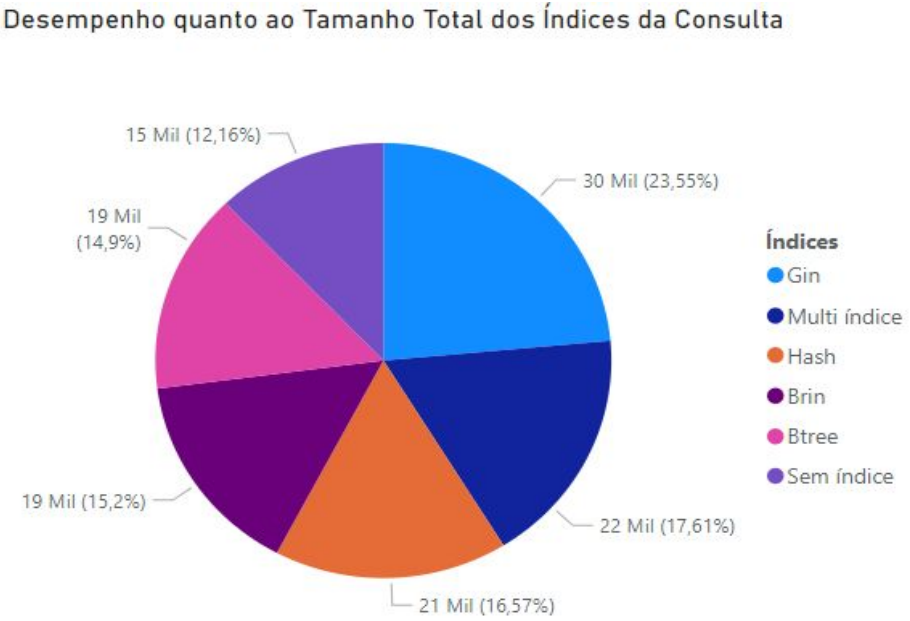
Menor tamanho:

- Sem índice;

Maior tamanho:

- Gin;

Desempenho Total



Análise

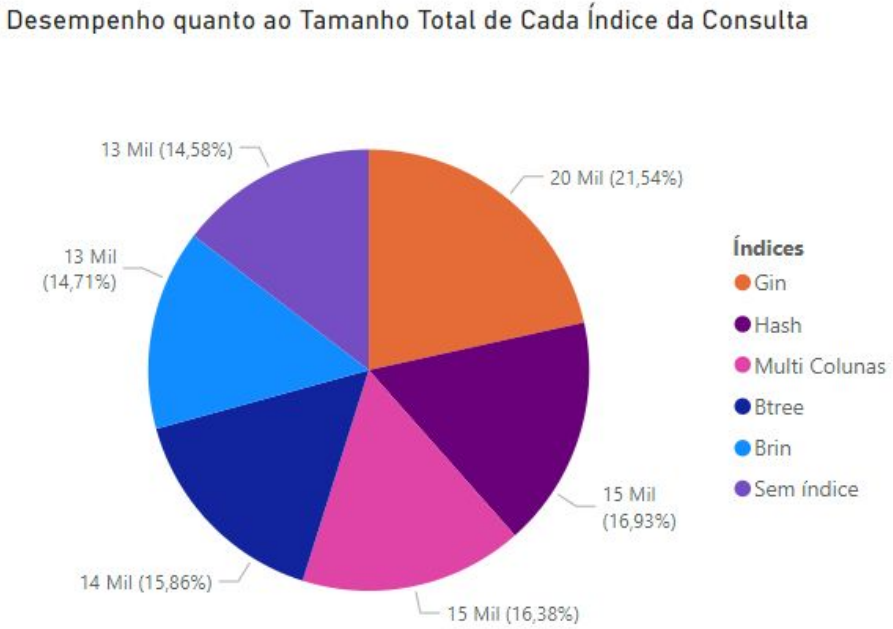
Menor tamanho:

- Sem índice;

Maior tamanho:

- Gin;

Desempenho Total



Análise

Menor tamanho:

- Sem índice;
- Brin;

Maior tamanho:

- Gin;

Links importantes:

Repositório dos códigos:

https://x.gd/swwNS

Documentação:

https://x.gd/6BAMy

Planilha :

https://x.gd/1Lwc1

Dúvidas?

Agradecimentos