# Trabalho de Banco de Dados

Índices

Anna Hellen Andrade Moura
Alana Oliveira
Rodrigo Fries
Thiago Giebmeyer

# 1. Introdução

O presente trabalho tem como objetivo apresentar e discutir resultados da utilização de “índices” aplicados em diferentes consultas SQL (Structured Query Language). Para o desenvolvimento deste trabalho foram utilizados 5 tipos de índices, sendo eles tipo Hash, B-tree, Brin, Gin e Multi-Colunas em um Banco de Dados Relacional criado com a estrutura do SGDB (Sistema de Gerenciamento de Banco de Dados) PostgreSQL.

# 2. Índices

Os índices são uma maneira de facilitar a busca de informações armazenadas em uma tabela, com a utilização de índices podemos reduzir a quantidade de operações de leitura para encontrar os dados relacionados a um conjunto de instruções presentes em um comando SQL.

## 2.1 Tipos de índice

- Esparso:
  - Um dado de entrada para cada página de arquivo
  - Usa menos espaço de armazenamento, porém leva mais tempo para localizar um registro dada a sua chave

- Denso
  - Pelo menos um dado de entrada por chave-valor
  - Sequência de blocos contendo apenas as chaves dos registros e os ponteiros para os próprios registros

## 2.2 Índice Hash

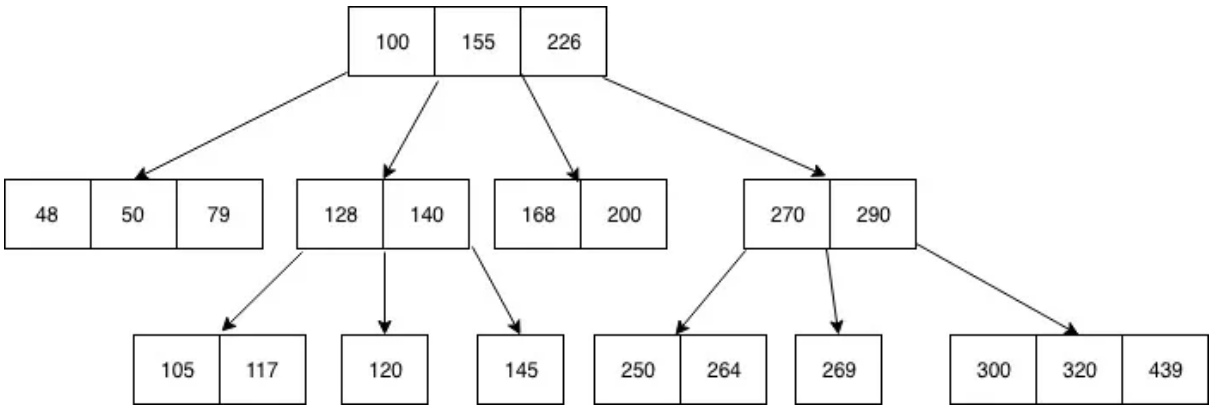
O índice do tipo “Hash”, é um índice que possui um tamanho fixo, sendo assim cada registro do banco de dados irá possuir um único ponteiro.

Esse tipo de índice é um valor gerado através de uma função de mapeamento que mapeia todo o conjunto de chaves de pesquisa K para o endereço onde os registros reais são colocados. É uma função de chaves de pesquisa para endereços dos Buckets que são considerados uma unidade de armazenamento. Normalmente armazena um bloco de disco completo, que por sua vez pode armazenar um ou mais registros.

### 2.3 Índice B-Tree

O índice do tipo “B-Tree”, é um índice utilizado para a busca de registros que possuam valores próximos.

O índice B-Tree opera como uma árvore binária de busca, na qual cada nó pode abrigar até um máximo definido de filhos. Essa estrutura é caracterizada pela sua organização balanceada dos dados, assegurando que a disparidade de alturas entre os filhos de um nó não exceda 1. O equilíbrio da árvore é mantido por meio de operações de rotação e redistribuição.




(Figura 1: Exemplo de um índice B-Tree)

### 2.4 Índice Multi-Colunas

Os índices do tipo “Multi-Colunas” (também conhecidos como índices compostos) são semelhantes aos índices padrão. Ambos armazenam uma “tabela” classificada de ponteiros para a tabela principal. Os índices de múltiplas colunas, entretanto, podem armazenar ponteiros classificados adicionais para outras colunas.

Índices multi-colunas são índices que armazenam dados em até 32 colunas. Ao criar um índice multicolunas, a ordem das colunas é muito importante. Isso se deve à estrutura que os índices multicolunas possuem. Os índices de múltiplas colunas são estruturados para ter uma estrutura hierárquica.

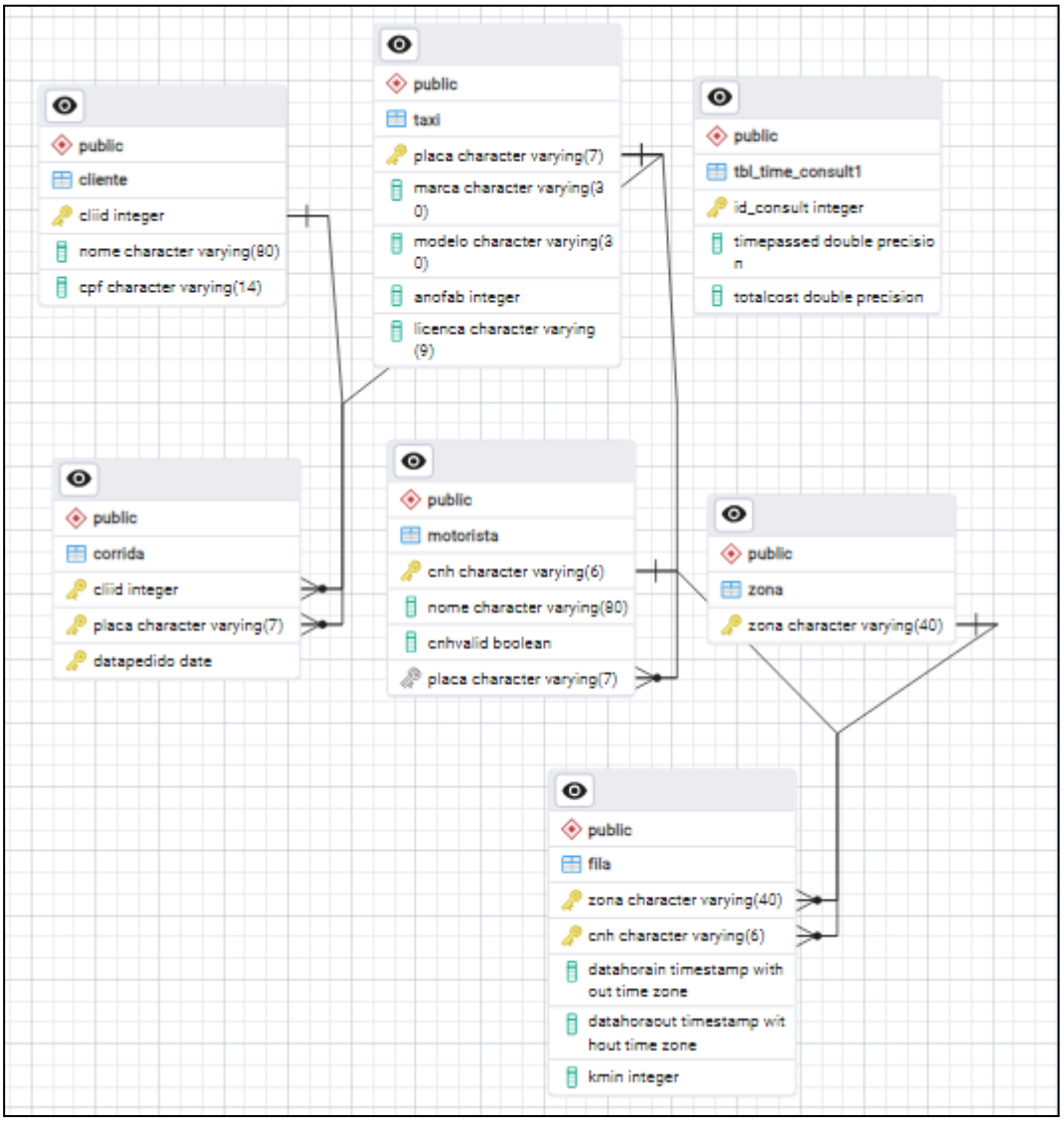
### 2.5 Índice GIN

O GIN foi projetado para lidar com casos em que os itens a serem indexados são valores compostos, e as consultas a serem tratadas pelo índice precisam procurar valores de elementos que aparecem nos itens compostos. Por exemplo, se você tem uma pilha de livros com palavras misturadas e quer encontrar rapidamente todos os livros que contêm uma palavra específica, o GIN organiza tudo para que você possa encontrar o que precisa de maneira rápida e fácil.

### 2.6 Índice Brin

Os índices BRIN - Block Range Indexes armazenam resumos sobre os valores armazenados em intervalos de blocos físicos consecutivos de uma tabela. Isso funciona melhor para colunas onde os valores seguem a mesma ordem que as linhas na tabela. Assim como outros tipos de índices, como GiST, SP-GiST e GIN, os índices BRIN suportam diferentes formas de organização e têm operações específicas para cada uma delas. Para tipos de dados que podem ser organizados em ordem, os índices BRIN guardam o menor e o maior valor em cada bloco.

## 3. Arquitetura do banco de dados




**(Figura 2: Diagrama ER do banco de dados)**

## 4. Consultas

Os dados apresentados a seguir foram organizados da seguinte forma: registros referentes a tamanho foram medidos em kilobytes (Kb), registros referentes a “Tempo” foram medidos em milissegundos (ms) e registros referentes a “Custo” foram medidos utilizando o custo de uma consulta no contexto retornados pelo comando ‘EXPLAIN ANALYZE’ que é tipicamente medido em unidades arbitrárias chamadas "custo de planejamento" (cost) pelo Postgre.

Nas Tabelas apresentadas abaixo foram organizados os dados referentes às 100 execuções com e sem índices e com base nelas foram gerados os graficos**.** As colunas de cabeçalho são referentes às médias dos seguintes dados: tempo de execução das consultas, custo de execução, tamanho total dos arquivos de dados, tamanho total dos arquivos de índices e tamanho do índex criado.

### 4.1 Consulta 1

| | Tempo (ms) | Custo (cost) | Tam. Total (Kb) | Tam. Total Índices (Kb) | Tam. index (Kb) |
|---|---|---|---|---|---|
| **c1** | 221,71 | 2455,89 | 5648 | 2400 | 2376 |
| **c1 Hash** | 303,12 | 2453,77 | 5736 | 2480 | 2408 |
| **c1 Btree** | 223,15 | 2455,89 | 7216 | 3296 | 2704 |
| **c1 Mult** | 217,04 | 2455,89 | 7104 | 3384 | 2704 |
| **c1 Brin** | 210,17 | 2455,89 | 6544 | 2824 | 2400 |
| **c1 Gin** | 2200,26 | 2455,9 | 9712 | 5992 | 4008 |

**(Tabela 1: Resultados para cada tipo de índice na consulta 1)**

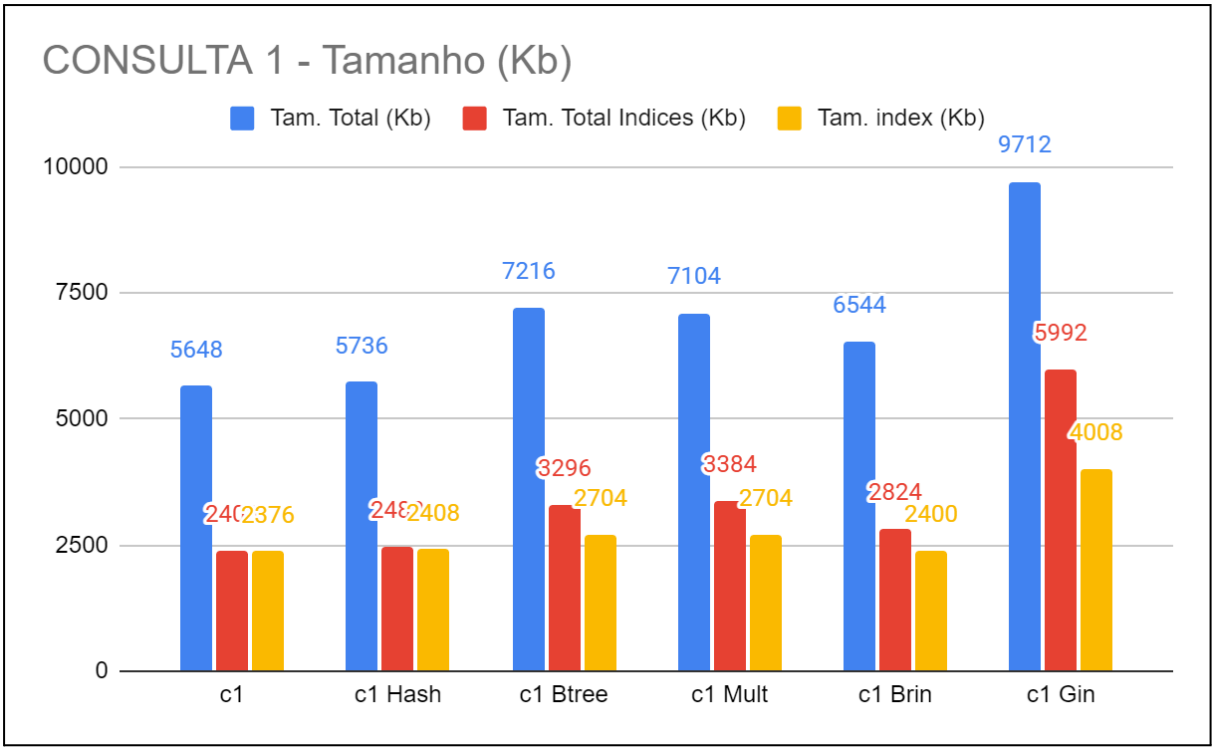
Para realizar a coleta dos dados apresentados na tabela 1, a seguinte consulta SQL foi executada:

```
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%';
```

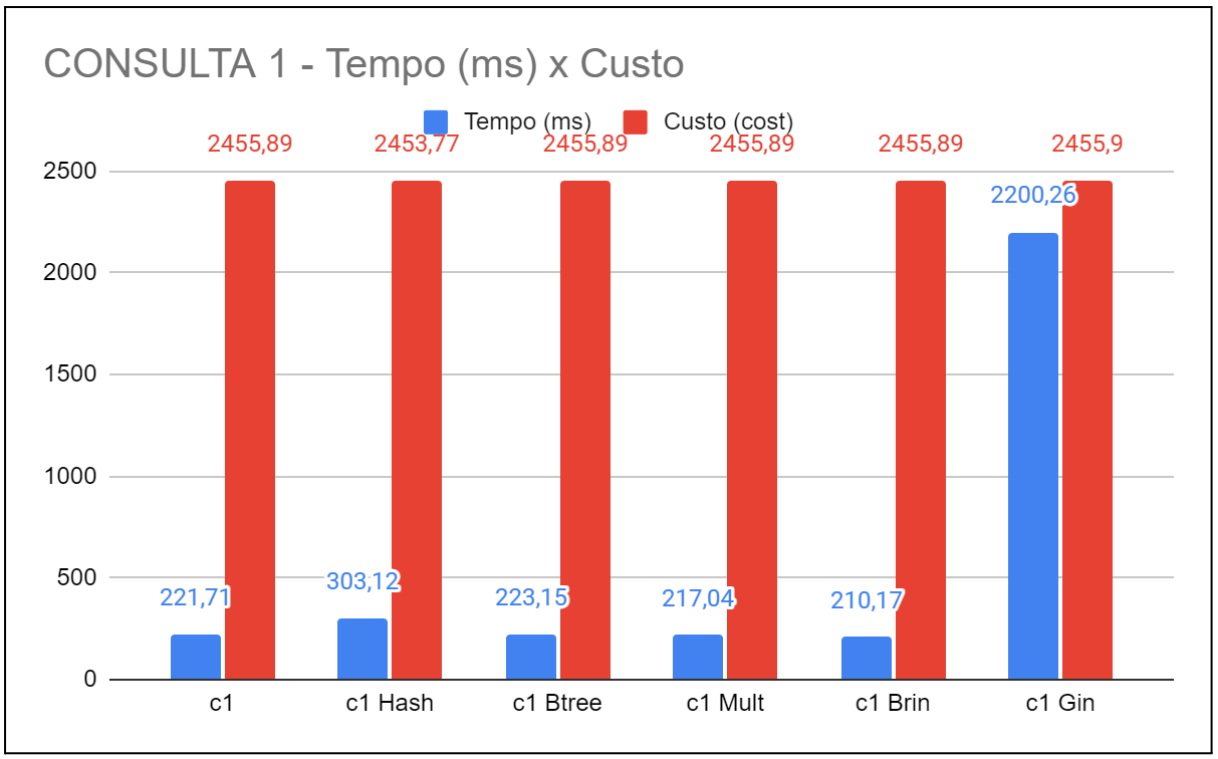
A consulta SQL acima é utilizada para retornar o nome do cliente, marca e modelo do táxi, a zona onde a corrida ocorreu e a data da corrida. Com base nesses dados, foram criado os índices:

```
--Indice Hash no campo zona
CREATE INDEX IDX_HASH_ZONA ON ZONA USING HASH(ZONA);

--Indice Btree no campo CPF da tabela Cliente
CREATE INDEX IDX_BTREE_CLIENTE_CPF ON CLIENTE USING BTREE(CPF);

--Indice multi colunas nos campos CliId e Placa da tabela Corrida
CREATE INDEX IDX_CLIID_PLACA ON CORRIDA(CLIID, PLACA);

--Indice BRIN no campo DataPedido da tabela corrida
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);
```




**(Gráfico 1: Representação gráfica do tamanho em Kilobytes para cada tipo de índice na consulta 1)**

Para esta consulta, não houve grande diferença no tamanho dos arquivos entre a execução dela sem nenhum índice comparada com quando foi criado o índice Hash, já quando comparada com os índices B-tree,  e B-tree multi coluna houve um pequeno aumento em todos os valores. Quanto ao índice Brin, o tamanho total dos índices e o tamanho do índice em questão diminuíram em relação a todos os outros, sendo ele o com menores valores nesses campos, isso se dá ao fato de que a data indicada pelo índice do tipo BRIN era o campo de data porém o mesmo não foi utilizado como cláusula na consulta. Já o índice Gin todos os valores aumentaram significativamente em relação aos demais, isso se deve ao fato da indexação usando GIN pode consumir mais espaço em disco do que outros tipos de índices devido à natureza do método de indexação trigram (pg_trgm). Ele cria uma estrutura de árvore B+ para armazenar trigramas, o que pode aumentar o tamanho total dos índices em comparação com índices convencionais.




**(Gráfico 2: Representação gráfica do Tempo em milissegundos x Custo para cada tipo de índice na consulta 1)**

No **Gráfico 2** é possível verificar que o custo total para todas as 100 execuções com e sem índices foi praticamente o mesmo. O tempo de execução, para as execuções sem índice, com índice Hash, B-tree, Multi colunas, e Brin, a variação foi de 210,17 ms para um máximo de 303,12 ms. Já no índice Brin houve um aumento abrupto no tempo de execução, chegando ao valor de 2200,26 ms, isto pode estar associado ao fato que na consulta o campo DataPedido foi utilizado com o operador BETWEEN, sendo uma busca em um intervalo de tempo, e o GIN possui melhor desempenho em buscas de textos livres ou para localizar onde um valor específico está.

### 4.2 Consulta 2

| Tempo (ms) | Custo (cost) | Tam. Total (Kb) | Tam. Total Índices (Kb) | Tam. index (Kb) |
|---|---|---|---|---|
| 108,24 | 1805,57 | 5648 | 2400 | 2376 |
| 120,74 | 1805,57 | 7504 | 3784 | 2904 |
| 124,06 | 1805,57 | 7248 | 3328 | 2720 |
| 119,85 | 1805,57 | 7104 | 3384 | 2704 |
| 78,28 | 1805,57 | 6544 | 2824 | 2400 |
| 117,28 | 1805,57 | 9712 | 5992 | 4008 |

**(Tabela 2: Resultados para cada tipo de índice na consulta 2)**

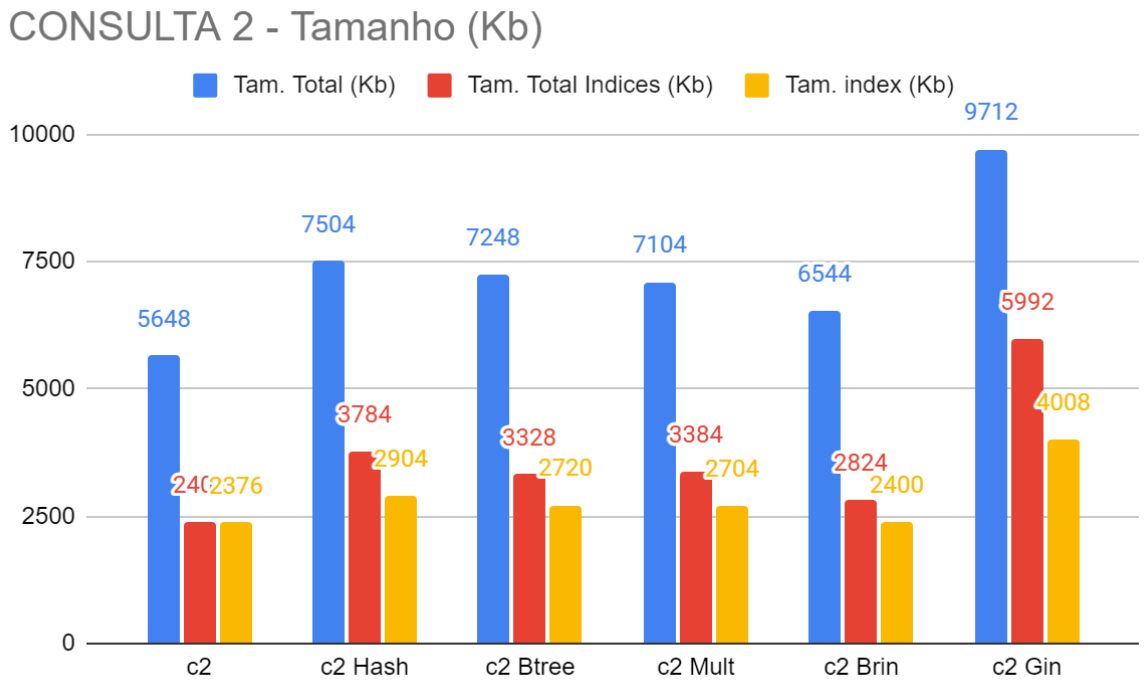
Para realizar a coleta dos dados apresentados na tabela 2, a seguinte consulta SQL foi executada:

```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;
```

A consulta SQL acima é utilizada para retornar o apenas o nome de cada cliente que possuir o total de corridas maior que 1 e entre '1/1/2000' a '1/1/2023'. Com base nesses dados, foram criado os índices:

```sql
--Indice Hash no campo CliId da tabela Corrida
CREATE INDEX IDX_HASH_CORRIDA_CLIID ON CORRIDA USING HASH(CLIID);
--Indice Btree no campo Nome da tabela Cliente
CREATE INDEX IDX_BTREE_CLIENTE_NOME ON CLIENTE USING BTREE(NOME);
--Indice multi colunas nos campos CliId e Placa na tabela Corrida
CREATE INDEX IDX_CLIID_PLACA ON CORRIDA(CLIID, PLACA);
--Indice BRIN no campo DataPedido da tabela Corrida
CREATE INDEX IDX_BRIN_DATA_PEDIDO ON CORRIDA USING BRIN(DATAPEDIDO);
--Indice GIN + Extensão pg_trgm define a classe de operadores necessária para indices GIN em strings.
CREATE EXTENSION IF NOT EXISTS PG_TRGM;
CREATE INDEX IDX_GIN_PLACA ON CORRIDA USING GIN (PLACA GIN_TRGM_OPS);
```

```
CREATE EXTENSION IF NOT EXISTS PG_TRGM;
CREATE INDEX IDX_GIN_DATA_PEDIDO ON CORRIDA USING GIN (PLACA GIN_TRGM_OPS);
```

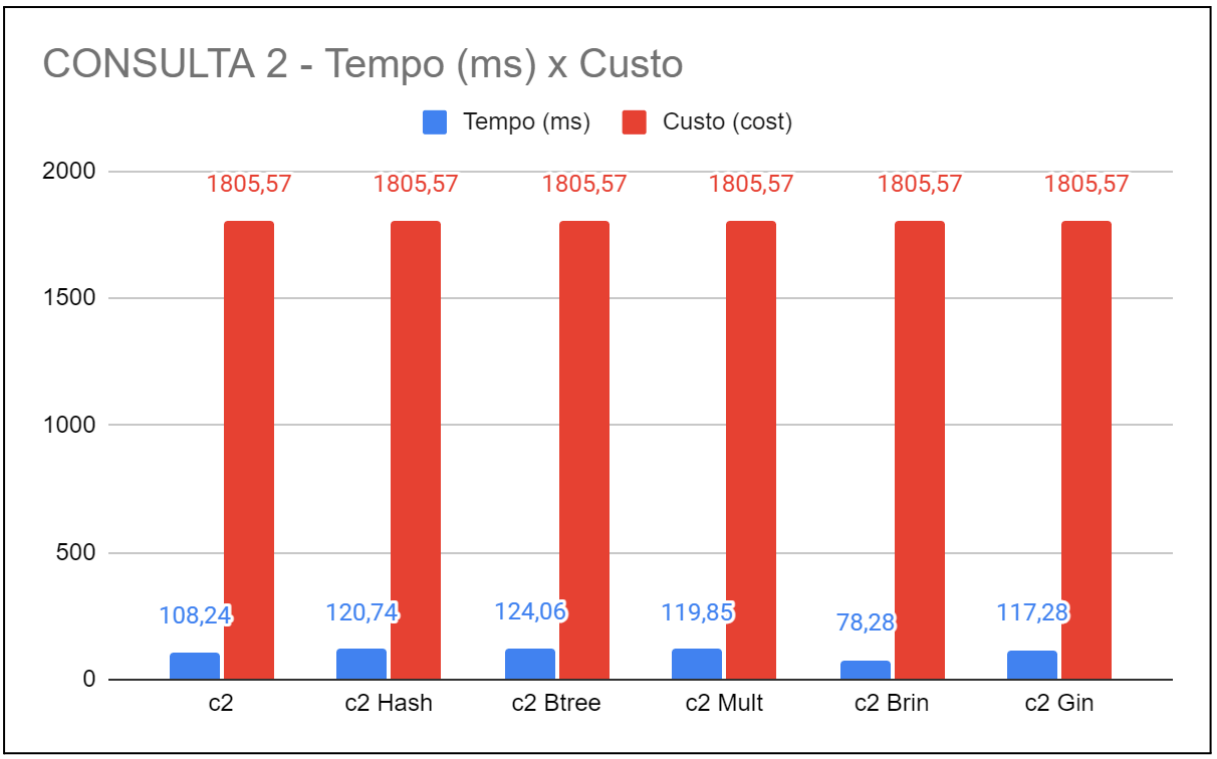



**(Gráfico 3: Representação gráfica do tamanho em Kilobytes para cada tipo de índice na consulta 2)**

No **gráfico 3**, podemos observar que o índice GIN tem um destaque devido a sua diferença de tamanho entre os outros índices. O maior volumes de memória ocupada pelo GIN deve-se principalmente pela razão pelo qual o mesmo foi projetado, ele é ideal para lidar com casos em que os itens a serem indexados são valores compostos ( matrizes ), e o valor a ser retornado na consulta deve fazer parte deste item. Por exemplo, no caso acima os itens seriam as listas de nomes dos clientes e a consulta seria procurar esses nomes que atendessem à condição especificada.

Ademais, o GIN tende a armazenar informações para cada item indexado, como um identificador de cada linha correspondente. E, por isso, nesta consulta ele teve um tamanho significativamente maior que os demais índices.

Já o índice BRIN tende a ser menor em comparação aos demais índices, e a sua varredura possui uma sobrecarga menor em relação à varredura sequencial por exemplo; mas, como no caso dessa consulta, o índice BRIN evitou a varredura em parte da tabela que não continha dados correspondentes, diminuindo assim o tempo de execução da mesma como veremos no **gráfico 4**.




**(Gráfico 4: Representação gráfica do Tempo em milissegundos x Custo para cada tipo de índice na consulta 2)**

No **gráfico 4**, o índice BRIN destaca-se por seu menor tempo de execução, isso deve-se, além dos fatos já mencionados acima, por seu funcionamento em intervalos de blocos ( páginas adjacentes nas tabelas). O mesmo faz consultas por meio de varreduras de índices bitmap, porém, o mesmo deve ser usado de preferência em dados ordenados; por isso, ele foi usado na tabela corrida, na qual continha datas, já que o índice BRIN ordena tais dados otimizando assim a consulta.

Vale ressaltar que, como apresentado no **gráfico 4**, o custo é o mesmo para todos os índices, porém todos os outros valores analisados são divergentes, ou seja, pode-se concluir que o custo de execução da consulta não tem relação direta com o tamanho e o tempo de execução da mesma.

### 4.3 Consulta 3

| | Tempo (ms) | Custo (cost) | Tam. Total (Kb) | Tam. Total Indices (Kb) | Tam. index (Kb) |
|---|---|---|---|---|---|
| **c3** | 111,61 | 1930,74 | 5648 | 2400 | 2376 |
| **c3 Hash** | 139,39 | 1930,74 | 7560 | 3760 | 2904 |
| **c3 Btree** | 124,54 | 1930,74 | 5704 | 2448 | 2392 |
| **c3 Mult** | 126,57 | 1930,74 | 7104 | 3384 | 2704 |
| **c3 Brin** | 132,49 | 1930,74 | 6600 | 2800 | 2400 |
| **c3 Gin** | 130,8 | 1930,74 | 9712 | 5992 | 4008 |

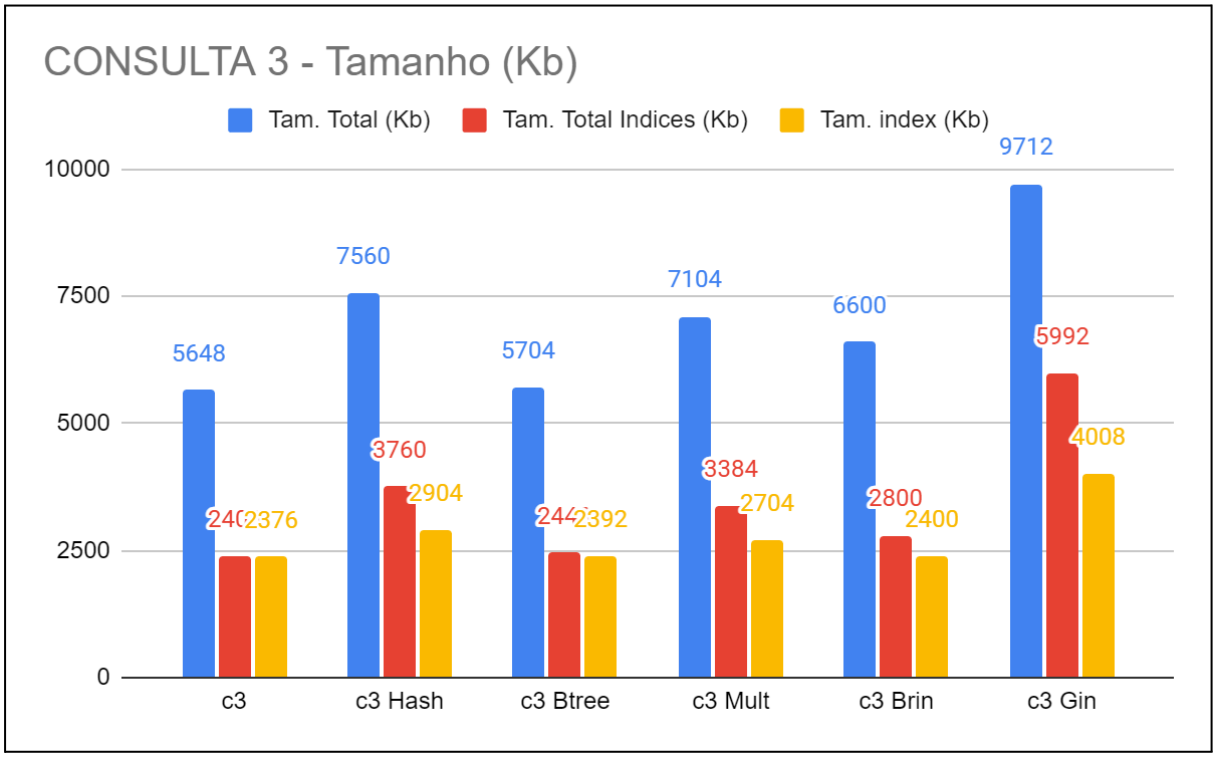
**(Tabela 3: Resultados para cada tipo de índice na consulta 3)**

Para realizar a coleta dos dados apresentados na tabela 3, a seguinte consulta SQL foi executada:

```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;
```

A consulta SQL acima é utilizada para retornar o apenas o nome do motorista, a zona onde a corrida ocorreu e a média de quilômetros percorridos por cada motorista em cada zona apenas para os motoristas que tiveram mais de uma ocorrência de corrida e que possua a média de quilometragem sendo um número par. Com base nesses dados, foram criado os índices:

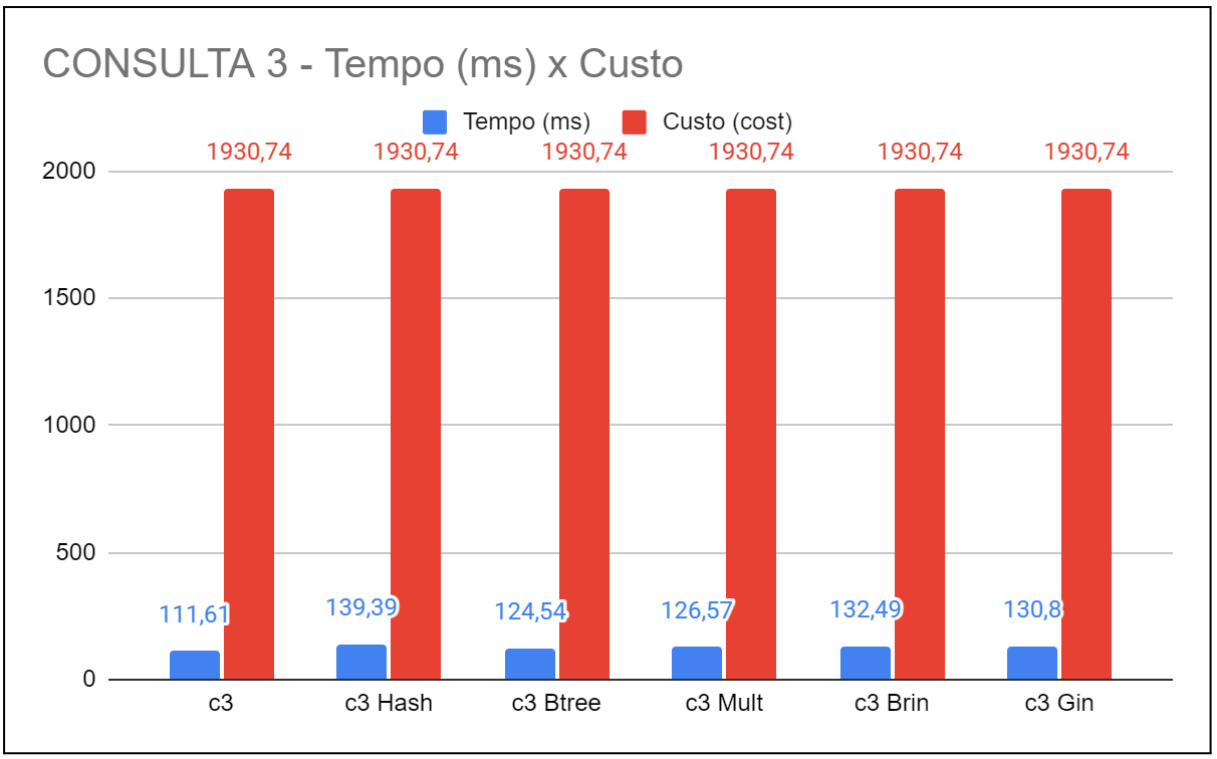
```
--Indice Hash no campo CNH da tabela Motorista
CREATE INDEX IDX_HASH_MOTORISTA_CNH ON MOTORISTA USING HASH(CNH);
--Indice Btree no campo Zona
CREATE INDEX IDX_BTREE_ZONA_ZONA ON ZONA USING BTREE(ZONA);
--Indice multi colunas nos campos CliId e Placa na tabela Corrida
CREATE INDEX IDX_CLIID_PLACA ON CORRIDA(CLIID, PLACA);
--Indice BRIN no campo Placa da tabela Motorista
CREATE INDEX IDX_BRIN_PLACA ON MOTORISTA USING BRIN (PLACA);
--Indice GIN + Essa 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);
```




**(Gráfico 5: Representação gráfica do tamanho em Kilobytes para cada tipo de índice na consulta 3)**

Esta consulta envolve várias junções entre todas as tabelas, faz utilização de operações de agregação e filtro “Having” também está presente, o que pode resultar em um conjunto de dados menor após a agregação. A consulta também calcula uma média (AVG) e aplica uma condição adicional para filtrar apenas as médias que sejam um número par. O índice GIN criado na coluna placa da tabela corrida foi útil nesta consulta, pois parece haver muitas junções envolvendo essa tabela e pode ajudar a acelerar o processo de busca e filtragem.

A diferença na alteração do tamanho dos registros após a criação do índice GIN pode ser atribuída à natureza da consulta, a utilização dos filtros, campos, às operações realizadas e à relevância do índice para cada uma delas.




**(Gráfico 6: Representação gráfica do Tempo em milissegundos x Custo para cada tipo de índice na consulta 3)**

Para essa consulta, não possui diferença entre o custo das consultas e não foi coletado uma grande diferença entre os tempos sem e com a utilização dos índices, isso se dá pela forma que os índices foram utilizados com relação a consulta, onde os índices selecionados podem não ser os mais adequados para a consulta em questão. Alguns índices podem não ser acionados pelo otimizador de consultas devido à forma como as cláusulas WHERE e JOIN são formuladas.

### 4.4 Consulta 4

| | Tempo (ms) | Custo (cost) | Tam. Total (Kb) | Tam. Total Indices (Kb) | Tam. index (Kb) |
|---|---|---|---|---|---|
| **c4** | 13,39 | 413,72 | 8776 | 5736 | 3752 |
| **c4 Hash** | 14,87 | 413,72 | 10688 | 7096 | 4280 |
| **c4 Btree** | 17,35 | 413,46 | 10296 | 6528 | 3992 |
| **c4 Mult** | 16,92 | 413,72 | 12192 | 8680 | 4080 |
| **c4 Brin** | 15,84 | 413,72 | 10888 | 7032 | 3776 |
| **c4 Gin** | 21,55 | 413,72 | 8776 | 5736 | 3552 |

**(Tabela 4: Resultados para cada tipo de índice na consulta 4)**

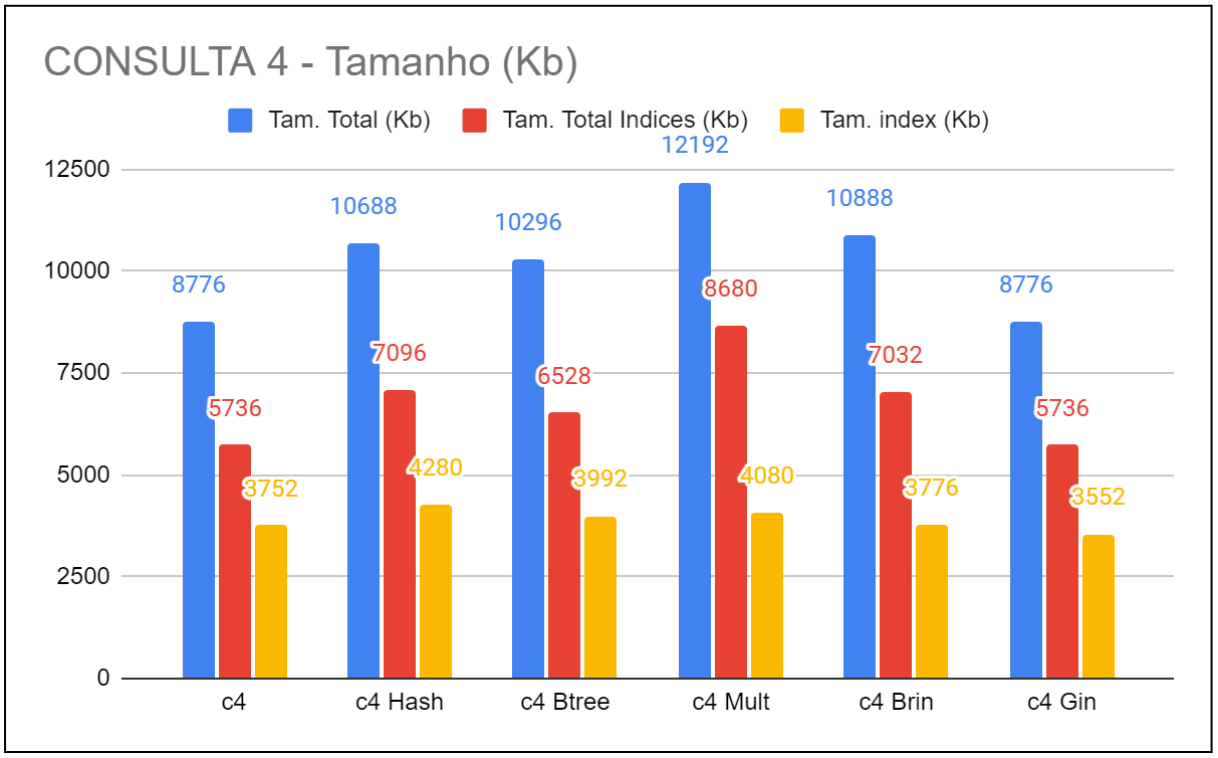
Para realizar a coleta dos dados apresentados na tabela4, a seguinte consulta SQL foi executada:

```
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;
```

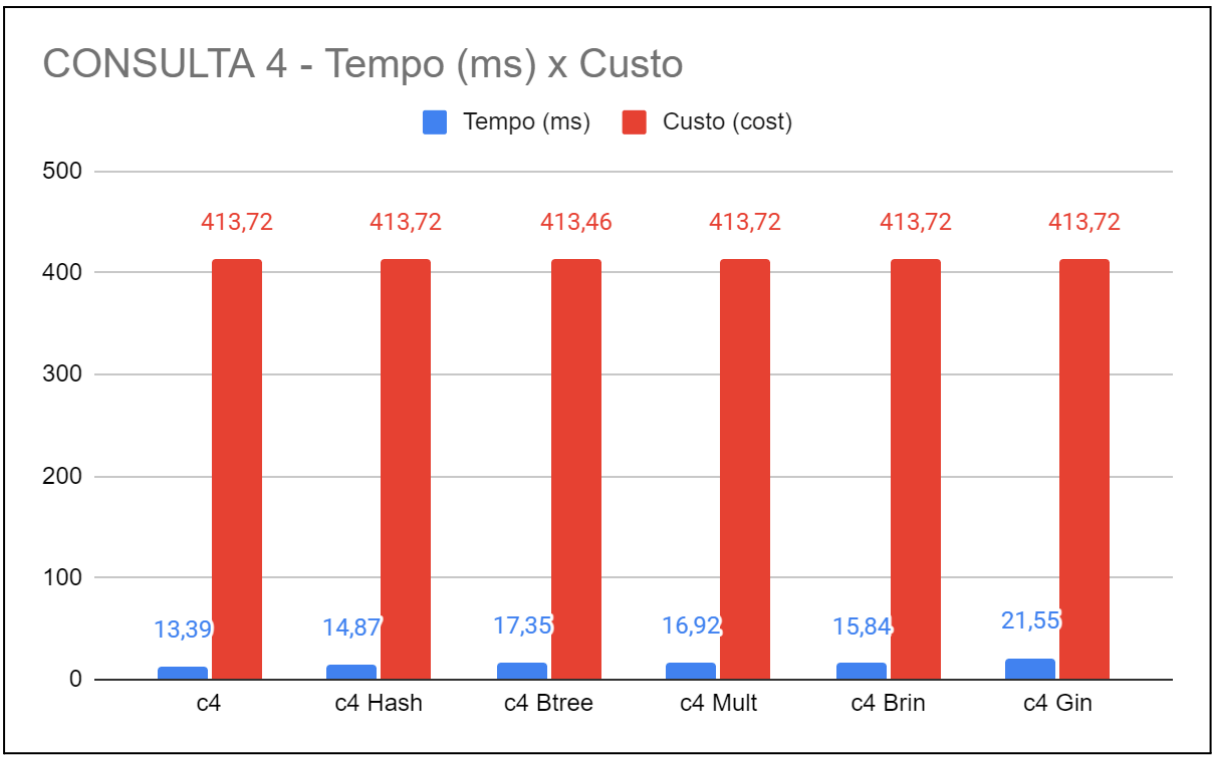
A consulta SQL acima é utilizada para retornar placa, marca e modelo do táxi, juntamente com o nome e CNH do motorista, para os táxis que realizaram corridas na zona 'Unicamp' e foram conduzidos por motoristas com licença válida. Com base nesses dados, foram criados os índices:

```
--Indice Hash no campo CNH da tabela Motorista
CREATE INDEX IDX_HASH_MOTORISTA_CNH ON MOTORISTA USING HASH(CNH);
--Indice Btree no campo Placa da tabela Taxi
CREATE INDEX IDX_BTREE_TAXI_PLACA ON TAXI USING BTREE(PLACA);
--Indice multi colunas nos campos CliId e Placa da tabela Corrida
```

```
CREATE INDEX IDX_CLIID_PLACA ON CORRIDA(CLIID, PLACA);
--Indice BRIN nos campos CNH e Zona da tabela Fila
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);
```




**(Gráfico 7: Representação gráfica do tamanho em Kilobytes para cada tipo de índice na consulta 4)**




**(Gráfico 8: Representação gráfica do Tempo em milissegundos x Custo para cada tipo de índice na consulta 4)**

O mult se destacou por ter o menor tamanho em relação ao da média das consultas, o gin não é adequado para essa consulta, pois foi o segundo maior em tamanho e mais lento, a consulta sem índice é o que tem maior tamanho de índice em relação à média dos dados, apesar da c4 sem índice ocupar mais espaço, foi a consulta que levou o menor tempo de busca.

### 4.5 Consulta 5

| | Tempo (ms) | Custo (cost) | Tam. Total (Kb) | Tam. Total Indices (Kb) | Tam. index (Kb) |
|---|---|---|---|---|---|
| **c5** | 48,96 | 1517,89 | 5648 | 2400 | 2376 |
| **c5 Hash** | 82,4 | 1516,19 | 7504 | 3784 | 2904 |
| **c5 Btree** | 74,78 | 1513,92 | 7168 | 3192 | 2616 |
| **c5 Mult** | 79,04 | 1516,19 | 7104 | 3384 | 2704 |
| **c5 Brin** | 79,73 | 1516,19 | 7760 | 3696 | 2400 |
| **c5 Gin** | 82,18 | 1516,19 | 9712 | 5992 | 4008 |

**(Tabela 5: Resultados para cada tipo de índice na consulta 5)**

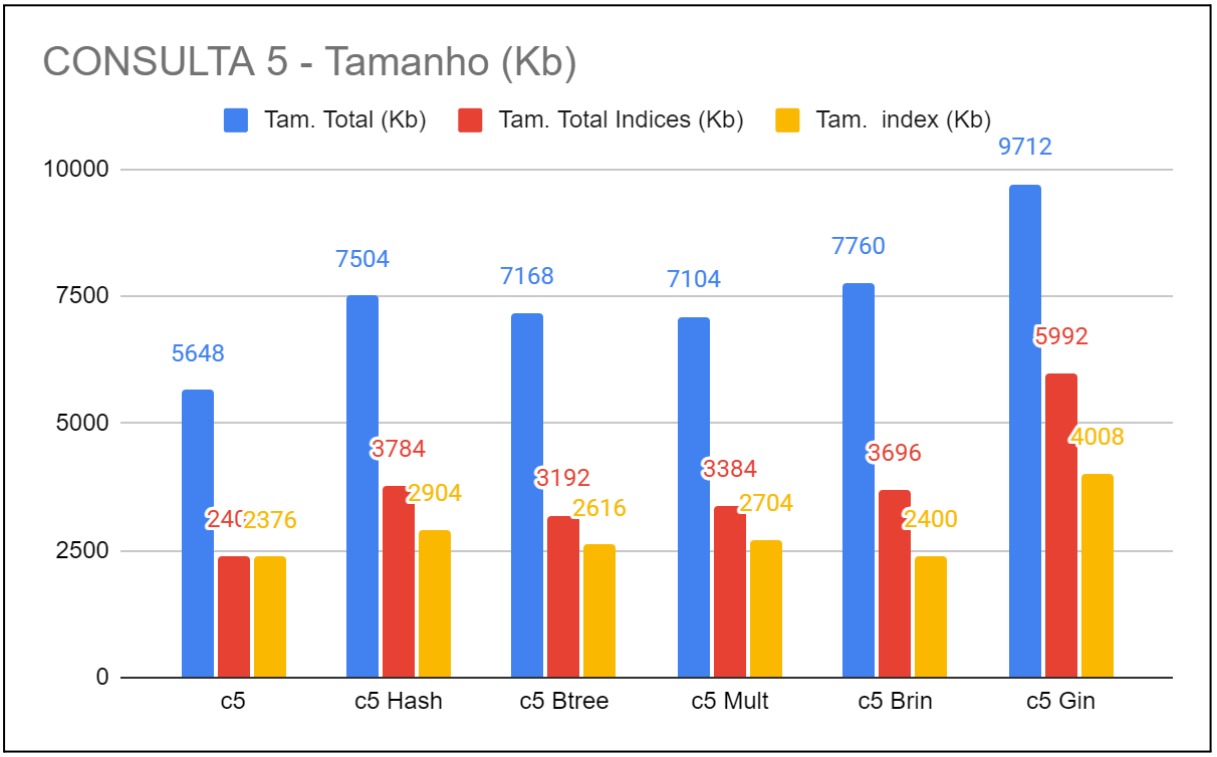
Para realizar a coleta dos dados apresentados na tabela 5, a seguinte consulta SQL foi executada:

```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;
```

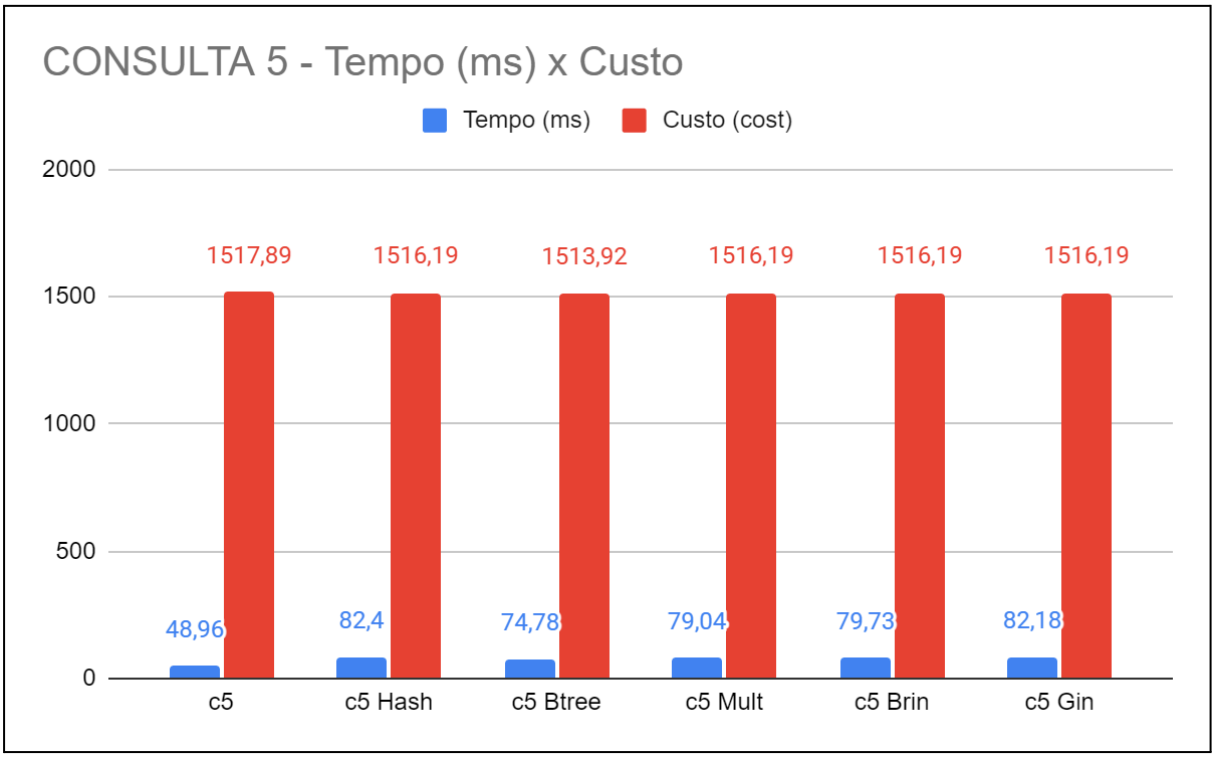
A consulta SQL acima é utilizada para retornar dados de todas as tabelas filtrando por nomes de clientes que contenham 'i', motoristas que possuam CNH valida, data do pedido entre '01-01-1999' e '31-12-2023' e também que a data o pedido tenha sido feita nos últimos 12 meses.. Com base nesses dados, foram criado os índices:

```sql
--Indice Hash no campo CliId da tabela Corrida
CREATE INDEX idx_hash_corrida_cliid ON Corrida USING hash(CliId);
--Indice Btree no campo Placa da tabela taxi
CREATE INDEX idx_btree_taxi_placa ON Taxi USING BTREE(Placa);
--Indice multi colunas nos campos CliId e Placa da tabela corrida
create index idx_cliId_placa on corrida(cliId, placa);
--Indice BRIN no campo DataHoraIn da tabela 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);
```




**(Gráfico 9: Representação gráfica do tamanho em Kilobytes para cada tipo de índice na consulta 5)**

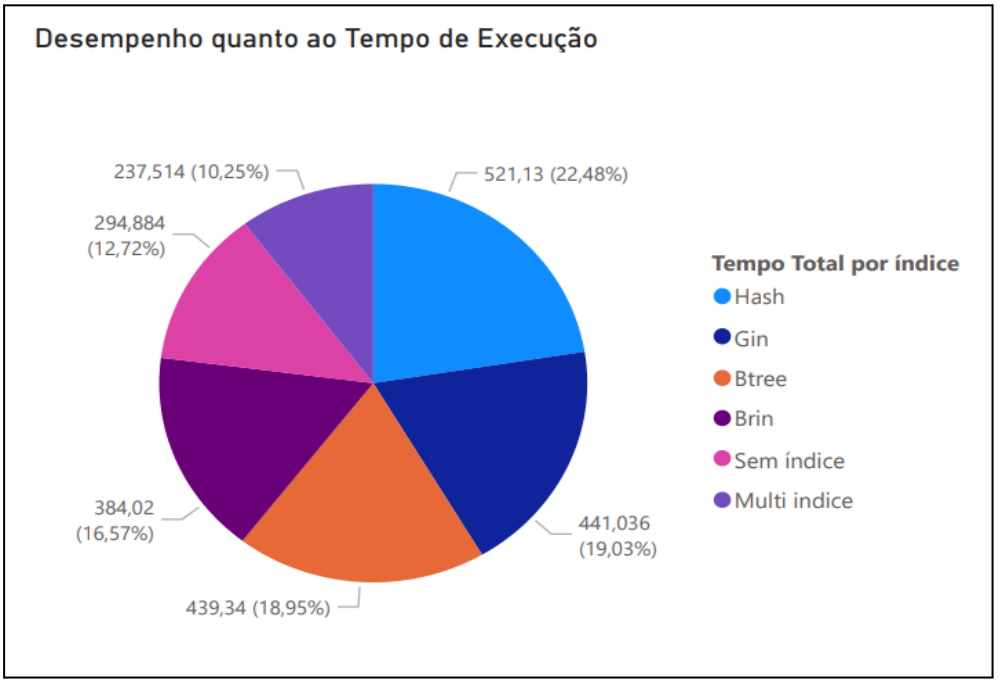



**(Gráfico 10: Representação gráfica do Tempo em milissegundos x Custo para cada tipo de índice na consulta 5)**

O gráfico apresenta a c5 com índice padrão teve o melhor tempo de busca, entretanto, o tamanho do índice ocupou mais espaço em relação à quantidade de dados, o tempo de execução dos demais foram similares e tiveram um tamanho também similar, com exceção do gin que teve um tamanho consideravelmente menor.

# 5. Resultados

## 5.1 Desempenho total quanto ao Tempo de Execução

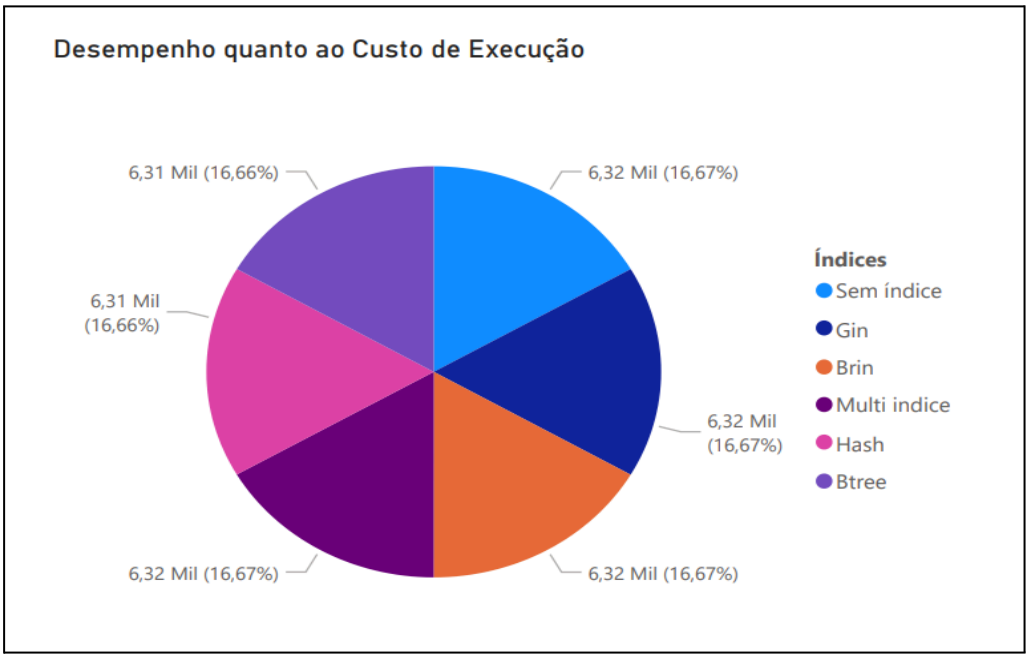



**(Gráfico 11: Representação gráfica do Tempo em milissegundos para cada tipo de índice em todas as consultas)**

Levando-se em consideração o gráfico acima, podemos concluir que o índice BRIN, de acordo com o valor médio de tempo de todos os índices em todas as consultas deste projeto, teve o melhor desempenho; já que, o mesmo obteve as execuções mais rápidas dentre os outros índices. Seu desempenho superior deve-se às suas características e particularidades já citadas em **2.6, 4.1, 4.2, 4.3, 4.4 e 4.5**.

Enquanto o índice multi-colunas teve as execuções mais lentas dentre todas as consultas apresentadas. Seu desempenho inferior deve-se às suas características e particularidades já citadas em **2.4, 4.1, 4.2, 4.3, 4.4 e 4.5**.

## 5.2 Desempenho total quanto ao Custo de Execução

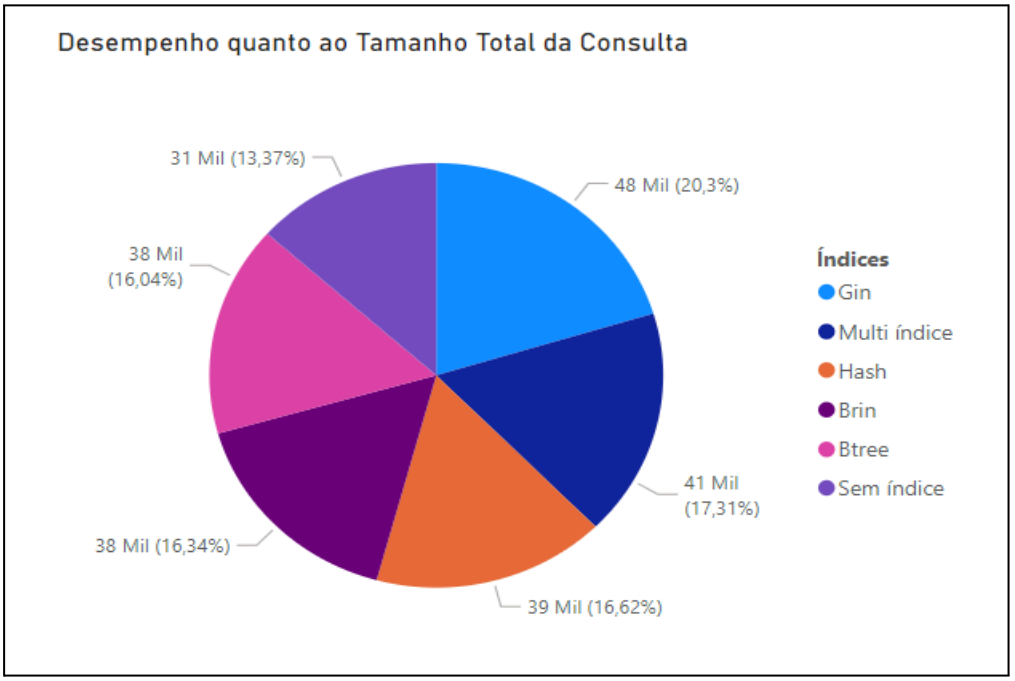



**(Gráfico 12: Representação gráfica do Custo para cada tipo de índice em todas as consultas)**

Levando-se em consideração o gráfico acima, podemos concluir que as consultas com índices hash e b-tree, de acordo com o valor médio de custo de todos os índices em todas as consultas deste projeto, obtiveram o melhor desempenho; já que, os mesmos tiveram as execuções menos custosas dentre os outros índices. Seu desempenho superior deve-se às suas características e particularidades já citadas em **2.2,2.3,4.1, 4.2, 4.3, 4.4 e 4.5**.

Enquanto os índices Gin, BRIN, MULTI COLUNAS e sem índices obtiveram as execuções mais custosas dentre todas as consultas apresentadas. Seu desempenho inferior deve-se às suas características e particularidades já citadas em **2.4, 2.5, 2.6, 4.1, 4.2, 4.3, 4.4 e 4.5**.

## 5.3 Desempenho total quanto ao Tamanho das Consultas

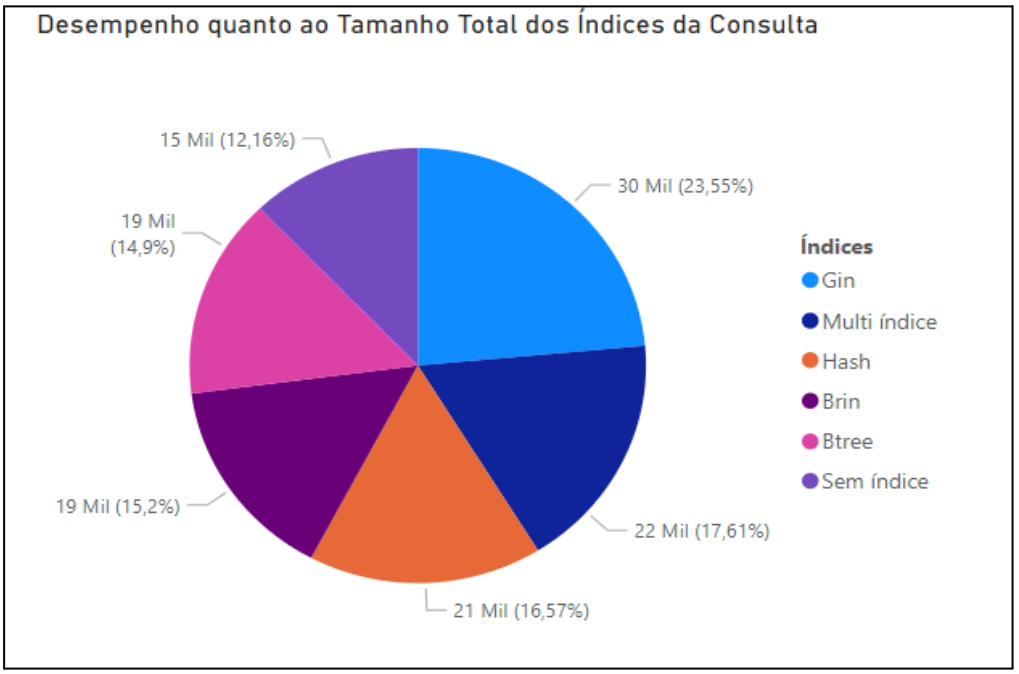



**(Gráfico 13: Representação gráfica do tamanho em Kilobytes para cada tipo de índice em todas as consultas)**

Levando-se em consideração o gráfico acima, podemos concluir que as consultas sem índice, de acordo com o valor médio do tamanho das consultas de todos os índices em todas as consultas deste projeto, teve o melhor desempenho; já que, o mesmo obteve o menor tamanho de consulta dentre os outros índices. Seu desempenho superior deve-se às suas características e particularidades já citadas em **4.1, 4.2, 4.3, 4.4 e 4.5**.

Enquanto o índice GIN teve as maiores execuções de consultas no quesito tamanho dentre todas as consultas apresentadas. Seu desempenho inferior deve-se às suas características e particularidades já citadas em **2.5, 4.1, 4.2, 4.3, 4.4 e 4.5**.

## 5.3 Desempenho total quanto ao Tamanho dos Índices das Consultas




**(Gráfico 14: Representação gráfica do desempenho para cada tipo de índice em todas as consultas)**

Levando-se em consideração o gráfico acima, podemos concluir que as consultas sem índice, de acordo com o valor médio do tamanho das consultas de todos os índices em todas as consultas deste projeto, teve o melhor desempenho; já que, o mesmo obteve o menor tamanho de consulta dentre os outros índices. Seu desempenho superior deve-se às suas características e particularidades já citadas em **4.1, 4.2, 4.3, 4.4 e 4.5**.

Enquanto o índice GIN teve as maiores execuções de consultas no quesito tamanho dentre todas as consultas apresentadas. Seu desempenho inferior deve-se às suas características e particularidades já citadas em **2.5, 4.1, 4.2, 4.3, 4.4 e 4.5**.

## 6. Referências bibliográficas

**Índices no PostgreSQL, Francisco Summa**. Disponível em: https://franciscosumma.blogspot.com/2016/02/indices-no-postgresql.html

**Laboratório de Banco de Dados, Índices**. Disponível em: https://sae.unb.br/cae/conteudo/unbfga/lbd/banco2_indices.html

**PostgreSQL - Índices com várias colunas**. Disponível em: https://acervolima.com/postgresql-indices-com-varias-colunas/

**Understanding Postgres GIN Indexes**: **The Good and the Bad**. Disponível em: https://pganalyze.com/blog/gin-index

**Stack Overflow. O que são os índices B-tree, hash, GiST e GIN?** Disponível em: https://pt.stackoverflow.com/questions/101065/o-que-são-os-índices-b-tree-hash-gist-e-gin

**Documentação PostgreSQl. Capítulo 11, Índices.** Disponível em: https://www.postgresql.org/docs/current/indexes.html

**Documentação PostgreSQl. Planos de consulta.** Disponível em: https://www.postgresql.org/docs/current/using-explain.html