UNIVERSIDADE FEDERAL DE SÃO CARLOS
CAMPUS SOROCABA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

SISTEMAS DE BANCO DE DADOS
Prof. Sahudy Montenegro González

PROJETO INTEGRADO
Fase Intermediária - Parte II

BASE DE DADOS BRASILEIRA
TEMA 4 - Cadastro Brasileiro de Escolas

Bianca Gomes Rodrigues - 743512
Pietro Zuntini Bonfim - 743588

Sorocaba-SP
21 de Abril de 2019

ÍNDICE

1 Descrição do Mini-Mundo

O objetivo deste projeto é criar uma aplicação que permita o armazenamento dos dados das escolas brasileiras do Brasil. O projeto, integrado com as disciplinas de `Desenvolvimento para Web` e `Sistemas de Bancos de Dados`, permitirá o gerenciamento e a visualização das escolas brasileiras.

Sobre os dados: É importante ressaltar que os dados escolhidos para o cadastramento das escola foram baseado nos microdados fornecidos pelo INEP.

2 Esquema do Banco de Dados

Nesta seção será apresentado o diagrama que contém as tabelas e atributos do banco de dados, além do significado de cada um dos atributos. Todas as informações encontram-se a seguir na figura 1.

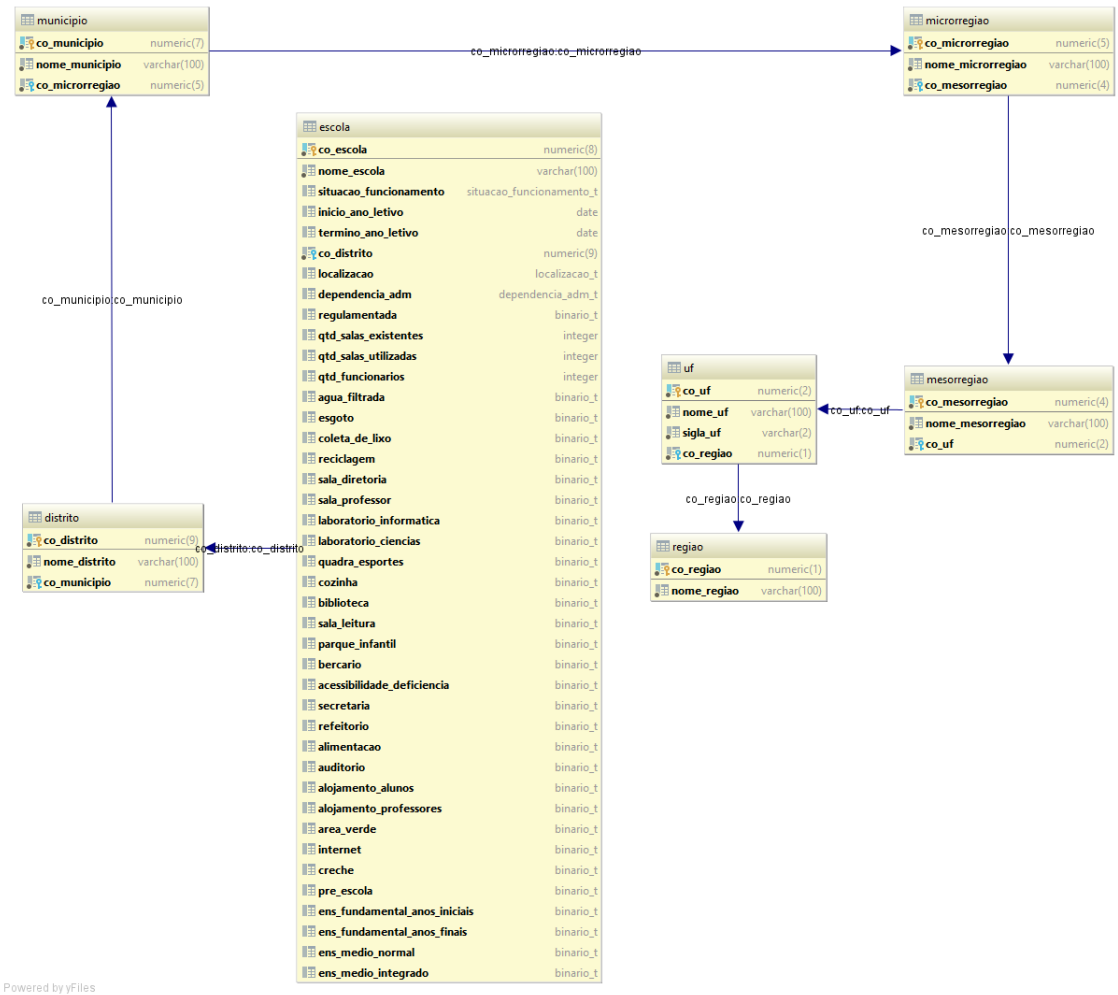

Figura 1: Diagrama do BD

Informações Básicas sobre a Escola

- `co_escola`: Código da Escola
- `nome_escola`: Nome da Escola
- `situacao_funcionamento`: Situação de Funcionamento da Escola (Em Atividade, Paralisada ou Extinta)
- `inicio_ano_letivo`: Data de Início do Ano Letivo
- `termino_ano_letivo`: Data de Término do Ano Letivo
- `dependencia_adm`: Tipo de Dependência Administrativa da Escola (Federal, Estadual, Municipal ou Privada)
- `regulamentada`: Se a Escola é regulamentada ou não
- `qtd_salas_existentes`: Número de salas existentes na escola
- `qtd_salas_utilizadas`: Número de salas sendo efetivamente utilizadas na escola
- `qtd_funcionarios`: Número de funcionários da escola

Informações de Localização Escola

- `co_distrito`: Código Completo do Distrito da Escola
- `localizacao`: Área da Localização da Escola (Urbana ou Rural)

Informações Adicionais sobre a Escola

- `agua_filtrada`: Se a Escola possui água filtrada ou não
- `esgoto`: Se a Escola possui sistema de esgoto ou não
- `coleta_de_lixo`: Se a Escola possui sistema de coleta de lixo ou não
- `reciclagem`: Se a Escola possui sistema de reciclagem de lixo ou não

Informações de Dependências da Escola

- `sala_diretoria`: Se a Escola possui uma sala de diretoria ou não
- `sala_professor`: Se a Escola possui salas de professor ou não
- `laboratorio_informatica`: Se a Escola possui um laboratório de informática ou não
- `laboratorio_ciencias`: Se a Escola possui um laboratório de ciências ou não
- `quadra_esportes`: Se a Escola possui quadra de esportes ou não
- `cozinha`: Se a Escola possui cozinha ou não

- `biblioteca`: Se a Escola possui biblioteca ou não
- `sala_leitura`: Se a Escola possui sala de leitura ou não
- `parque_infantil`: Se a Escola possui parque infantil ou não
- `bercario`: Se a Escola possui berçário ou não
- `acessibilidade_deficiencia`: Se a Escola possui dependências e vias adequadas à alunos com deficiência ou mobilidade reduzida ou não
- `secretaria`: Se a Escola possui secretaria ou não
- `refeitorio`: Se a Escola possui refeitório ou não
- `alimentacao`: Se a Escola oferece alimentação ou não
- `auditorio`: Se a Escola possui auditório ou não
- `alojamento_alunos`: Se a Escola possui alojamento para alunos ou não
- `alojamento_professores`: Se a Escola possui alojamento para professores ou não
- `area_verde`: Se a Escola possui uma área verde ou não
- `internet`: Se a Escola possui acesso à internet ou não

Informações de Oferta de Matrícula

- `creche`: Se a Escola oferece creche ou não
- `pre_escola`: Se a Escola oferece pré-escola ou não
- `ens_fundamental_anos_iniciais`: Se a Escola oferece Ensino Fundamental do 1º ao 5º ano ou não
- `ens_fundamental_anos_finais`: Se a Escola oferece Ensino Fundamental do 5º ao 9º ano ou não
- `ens_medio_normal`: Se a Escola oferece Ensino Médio do 1º ao 3º ano ou não
- `ens_medio_integrado`: Se a Escola oferece Ensino Médio integrado com Curso Técnico ou não

3 Especificação de Consultas

Nesta seção serão especificadas as consultas que farão parte do projeto. É importante ressaltar que são **duas** consultas, com atributos relativos (expressões regulares) e absolutos (expressões exatas). As consultas definidas pelo grupo serão apresentadas à seguir.

1. Buscar todas as escolas disponíveis em um determinado Estado (UF) e Munícipio (do Estado previamente selecionado), especificando um trecho do nome da Escola (dentre as previamente selecionadas no Munícipio). Além disso, o resultado obtido será ranqueado pelo número de funcionários da Escola.

   Campos de busca: UF e Município (absolutos), e Nome da Escola (relativo).

   Campos de visualização do resultado: Inicialmente código da escola, nome, situação de funcionamento, dependência administrativa e ofertas de matricula. Posteriormente, será possível a visualização dos demais campos da escola.

   Operadores das condições: nome da escola (ILIKE) e os demais (=)

   SQL:

```
SELECT e.co_escola, e.nome_escola, e.situacao_funcionamento,
       e.dependencia_adm, e.bercario, e.creche, e.pre_escola,
       e.ens_fundamental_anos_iniciais, e.ens_fundamental_anos_finais,
       e.ens_medio_normal, e.ens_medio_integrado
FROM escola e
JOIN distrito d on e.co_distrito = d.co_distrito
JOIN municipio m on d.co_municipio = m.co_municipio
JOIN microrregiao mi on m.co_microrregiao = mi.co_microrregiao
JOIN mesorregiao me on mi.co_mesorregiao = me.co_mesorregiao
JOIN uf u on me.co_uf = u.co_uf
WHERE u.co_uf = <codigo_uf>
AND m.co_municipio = <codigo_municipio>
AND e.nome_escola ILIKE '%<nome_escola>%'
ORDER BY e.qtd_funcionarios;
```

2. Buscar o número de escolas com cada tipo de dependência administrativa: Federal, Estadual, Municipal e Privada em uma determinada Região.

   Campos de busca: Código da Região.

   Campos de visualização do resultado: Quantidade de Escolas na Região, quantidade de Escolas Federais, quantidade de Escolas Estaduais, quantidade de Escolas Municipais e quantidade de Escolas Privadas.

   Operadores das condições: Código da Região (=).

   SQL:

```
SELECT count(e.co_escola) as qtd_escolas, (
     SELECT count(e2.co_escola)
     FROM escola e2
     WHERE e2.dependencia_adm = 'Federal'
  ) as qtd_federal, (
     SELECT count(e3.co_escola)
     FROM escola e3
     WHERE e3.dependencia_adm = 'Estadual'
  ) as qtd_estadual, (
```

```
        SELECT count(e4.co_escola)
        FROM escola e4
        WHERE e4.dependencia_adm = 'Municipal'
    ) as qtd_municipal, (
        SELECT count(e5.co_escola)
        FROM escola e5
        WHERE e5.dependencia_adm = 'Privada'
    ) as qtd_privada
FROM escola e
JOIN distrito d on e.co_distrito = d.co_distrito
JOIN municipio m on d.co_municipio = m.co_municipio
JOIN microrregiao m2 on m.co_microrregiao = m2.co_microrregiao
JOIN mesorregiao m3 on m2.co_mesorregiao = m3.co_mesorregiao
JOIN uf u on m3.co_uf = u.co_uf
JOIN regiao r on u.co_regiao = r.co_regiao
WHERE r.co_regiao = <codigo_regiao>;
```

4 Otimização das Consultas

Nesta seção iremos analisar o desempenho de cada uma das consultas especificadas na seção 3 (Especificação das Consultas). Utilizando técnicas de indexação e otimização, tentaremos alcançar um aumento significativo no desempenho das consultas. O sistema de gerenciamento de banco de dados utilizado é o PostgreSQL, versão 9.5. A máquina utilizada para testes conta com um processador i7 e 8GB de memória RAM.

4.1 Consulta 1 - Buscar Escolas Disponíveis

Para fins de teste, utilizaremos os seguintes dados como parâmetros para realização das consultas:

- codigo_uf = 35 (São Paulo)
- codigo_municipio = 3552205 (Sorocaba)
- nome_escola ILIKE '%edu%' (contém no nome o trecho 'edu')

4.1.1 Com JOIN

Inicialmente executamos a consulta para obter o tempo de execução.

CONSULTA SQL - JOIN

```
SELECT e.co_escola, e.nome_escola, e.situacao_funcionamento,
       e.dependencia_adm, e.bercario, e.creche, e.pre_escola,
       e.ens_fundamental_anos_iniciais, e.ens_fundamental_anos_finais,
       e.ens_medio_normal, e.ens_medio_integrado
FROM escola e
```

```
JOIN distrito d on e.co_distrito = d.co_distrito
JOIN municipio m on d.co_municipio = m.co_municipio
JOIN microrregiao mi on m.co_microrregiao = mi.co_microrregiao
JOIN mesorregiao me on mi.co_mesorregiao = me.co_mesorregiao
JOIN uf u on me.co_uf = u.co_uf
WHERE u.co_uf = 35
AND m.co_municipio = 3552205
AND e.nome_escola ILIKE '%edu%'
ORDER BY e.qtd_funcionarios;
```

TEMPO DE EXECUÇÃO

30 secs 246 msec.
96 rows affected.

Conforme o esperado, devido a muitos JOINs, o tempo da consulta foi ruim. Por conseguinte, executamos a mesma consulta com o comando `EXPLAIN ANALYZE`, obtendo o seguinte plano de execução:

PLANO DE EXECUÇÃO

```
"Sort  (cost=12209.24..12209.25 rows=1 width=76) (actual time=526.264..526.271 rows=96 loops=1)"
"  Sort Key: e.qtd_funcionarios"
"  Sort Method: quicksort  Memory: 38kB"
"  ->  Nested Loop  (cost=206.50..12209.23 rows=1 width=76) (actual time=371.788..526.146 rows=96 loops=1)"
"        ->  Nested Loop  (cost=206.50..12207.89 rows=1 width=81) (actual time=371.761..524.926 rows=96 loops=1)"
"              ->  Nested Loop  (cost=0.70..16.81 rows=1 width=11) (actual time=0.025..0.031 rows=1 loops=1)"
"                    ->  Nested Loop  (cost=0.56..16.60 rows=1 width=11) (actual time=0.019..0.022 rows=1 loops=1)"
"                          ->  Index Scan using municipio_pkey on municipio m  (cost=0.28..8.30 rows=1 width=13) (actual time=0.013..0.014 rows=1 loops=1)"
"                                Index Cond: (co_municipio = '3552205'::numeric)"
"                          ->  Index Scan using microrregiao_pkey on microrregiao mi  (cost=0.28..8.29 rows=1 width=12) (actual time=0.003..0.004 rows=1 loops=1)"
"                                Index Cond: (co_microrregiao = m.co_microrregiao)"
"                    ->  Index Scan using mesorregiao_pkey on mesorregiao me  (cost=0.14..0.19 rows=1 width=10) (actual time=0.004..0.006 rows=1 loops=1)"
"                          Index Cond: (co_mesorregiao = mi.co_mesorregiao)"
"                          Filter: (co_uf = '35'::numeric)"
"              ->  Hash Join  (cost=205.80..12191.00 rows=8 width=82) (actual time=371.727..524.859 rows=96 loops=1)"
"                    Hash Cond: (e.co_distrito = d.co_distrito)"
"                    ->  Seq Scan on escola e  (cost=0.00..11836.69 rows=39582 width=85) (actual time=0.094..516.432 rows=30776 loops=1)"
"                          Filter: ((nome_escola)::text ~~* '%edu%'::text)"
"                          Rows Removed by Filter: 255199"
"                    ->  Hash  (cost=205.78..205.78 rows=2 width=15) (actual time=1.994..1.994 rows=1 loops=1)"
"                          Buckets: 1024  Batches: 1  Memory Usage: 9kB"
"                          ->  Seq Scan on distrito d  (cost=0.00..205.78 rows=2 width=15) (actual time=1.452..1.977 rows=1 loops=1)"
"                                Filter: (co_municipio = '3552205'::numeric)"
"                                Rows Removed by Filter: 10301"
"        ->  Seq Scan on uf u  (cost=0.00..1.34 rows=1 width=12) (actual time=0.001..0.004 rows=1 loops=96)"
"              Filter: (co_uf = '35'::numeric)"
"              Rows Removed by Filter: 26"
"Planning time: 1.351 ms"
"Execution time: 526.530 ms"
```

Figura 2: Consulta 1 - Original

Analisando o plano de execução, podemos observar que a `qtd_funcionarios` proveniente da tabela `Escola` foi utilizada para ordenador os resultados obtidos e que o método de ordenação adotado foi o `quicksort`

4.1.2 COM IN

A consulta inicial utiliza `JOIN`. Todavia, apesar de obtermos o resultado esperado, o tempo de execução foi consideravelmente alto. Uma medida adotada para obter o mesmo resultado, mas em um tempo de execução menor, foi utilizar `IN` ao invés de `JOIN`.

CONSULTA SQL - IN

```
SELECT e.co_escola, e.nome_escola, e.situacao_funcionamento,
     e.dependencia_adm, e.bercario, e.creche, e.pre_escola,
     e.ens_fundamental_anos_iniciais, e.ens_fundamental_anos_finais,
     e.ens_medio_normal, e.ens_medio_integrado
FROM escola e
WHERE e.co_distrito IN (
   SELECT d.co_distrito
   FROM distrito d
   WHERE d.co_municipio IN (
      SELECT m.co_municipio
      FROM municipio m
      WHERE m.co_microrregiao IN (
         SELECT mi.co_microrregiao
         FROM microrregiao mi
         WHERE mi.co_mesorregiao IN (
            SELECT me.co_mesorregiao
            FROM mesorregiao me
            WHERE me.co_uf = 35
         )
      )
   )
   AND d.co_municipio = 3552205
)
AND e.nome_escola ILIKE '%edu%'
ORDER BY e.qtd_funcionarios;
```

TEMPO DE EXECUÇÃO

19 secs 94 msec.
2 rows affected.

PLANO DE EXECUÇÃO

```
"Sort  (cost=12068.04..12068.04 rows=1 width=76) (actual time=513.862..513.862 rows=2 loops=1)"
"  Sort Key: e.qtd_funcionarios"
"  Sort Method: quicksort  Memory: 25kB"
"  ->  Nested Loop Semi Join  (cost=16.67..12068.03 rows=1 width=76) (actual time=392.434..513.847 rows=2 loops=1)"
"        Join Filter: (e.co_distrito = d.co_distrito)"
"        Rows Removed by Join Filter: 29"
"        ->  Seq Scan on escola e  (cost=0.00..11836.69 rows=28 width=85) (actual time=0.740..511.727 rows=31 loops=1)"
"              Filter: ((nome_escola)::text ~~* '%uirapuru%'::text)"
"              Rows Removed by Filter: 285944"
"        ->  Materialize  (cost=16.67..230.50 rows=2 width=9) (actual time=0.047..0.064 rows=1 loops=31)"
"              ->  Nested Loop Semi Join  (cost=16.67..230.49 rows=2 width=9) (actual time=1.442..1.965 rows=1 loops=1)"
"                    ->  Seq Scan on distrito d  (cost=0.00..205.78 rows=2 width=15) (actual time=1.171..1.693 rows=1 loops=1)"
"                          Filter: (co_municipio = '3552205'::numeric)"
"                          Rows Removed by Filter: 10301"
"                    ->  Materialize  (cost=16.67..24.69 rows=1 width=6) (actual time=0.269..0.269 rows=1 loops=1)"
"                          ->  Hash Semi Join  (cost=16.67..24.69 rows=1 width=6) (actual time=0.264..0.264 rows=1 loops=1)"
"                                Hash Cond: (m.co_microrregiao = mi.co_microrregiao)"
"                                ->  Index Scan using municipio_pkey on municipio m  (cost=0.28..8.30 rows=1 width=13) (actual time=0.028..0.028 rows=1 loops=1)"
"                                      Index Cond: (co_municipio = '3552205'::numeric)"
"                                ->  Hash  (cost=15.62..15.62 rows=61 width=7) (actual time=0.218..0.218 rows=63 loops=1)"
"                                      Buckets: 1024  Batches: 1  Memory Usage: 11kB"
"                                      ->  Hash Semi Join  (cost=3.90..15.62 rows=61 width=7) (actual time=0.112..0.203 rows=63 loops=1)"
"                                            Hash Cond: (mi.co_mesorregiao = me.co_mesorregiao)"
"                                            ->  Seq Scan on microrregiao mi  (cost=0.00..9.58 rows=558 width=12) (actual time=0.004..0.034 rows=558 loops=1)"
"                                            ->  Hash  (cost=3.71..3.71 rows=15 width=5) (actual time=0.065..0.065 rows=15 loops=1)"
"                                                  Buckets: 1024  Batches: 1  Memory Usage: 9kB"
"                                                  ->  Seq Scan on mesorregiao me  (cost=0.00..3.71 rows=15 width=5) (actual time=0.022..0.056 rows=15 loops=1)"
"                                                        Filter: (co_uf = '35'::numeric)"
"                                                        Rows Removed by Filter: 122"
"Planning time: 1.952 ms"
"Execution time: 514.239 ms"
```

Figura 3: Consulta 1 - Operador IN

É possível notar que ao adotarmos o `IN` no lugar de `JOIN` houve uma queda considerável de 11 secs 152 msec, aproximadamente 37% do tempo inicial.

4.1.3 CRIAÇÃO DO ÍNDICE EM NOME ESCOLA

Uma tentativa para melhorar o desempenho da consulta foi criar um índice para a coluna `nome_escola` da tabela `Escola`:

```
CREATE INDEX nome_escola_index
ON escola(nome_escola)
2 secs 936 msec.
```

Realizamos novamente a mesma consulta com o operador `IN` e obtemos o seguinte tempo de execuçã:

16 secs 504 msec.
96 rows affected.

PLANO DE EXECUÇÃO

```
"Sort  (cost=12171.45..12171.48 rows=12 width=76) (actual time=498.379..498.387 rows=96 loops=1)"
"  Sort Key: e.qtd_funcionarios"
"  Sort Method: quicksort  Memory: 38kB"
"  ->  Hash Semi Join  (cost=230.52..12171.24 rows=12 width=76) (actual time=356.092..498.304 rows=96 loops=1)"
"        Hash Cond: (e.co_distrito = d.co_distrito)"
"        ->  Seq Scan on escola e  (cost=0.00..11836.69 rows=39582 width=85) (actual time=0.094..490.185 rows=30776 loops=1)"
"              Filter: ((nome_escola)::text ~~* '%edu%'::text)"
"              Rows Removed by Filter: 255199"
"        ->  Hash  (cost=230.49..230.49 rows=2 width=9) (actual time=2.310..2.310 rows=1 loops=1)"
"              Buckets: 1024  Batches: 1  Memory Usage: 9kB"
"              ->  Nested Loop Semi Join  (cost=16.67..230.49 rows=2 width=9) (actual time=1.655..2.306 rows=1 loops=1)"
"                    ->  Seq Scan on distrito d  (cost=0.00..205.78 rows=2 width=15) (actual time=1.358..2.009 rows=1 loops=1)"
"                          Filter: (co_municipio = '3552205'::numeric)"
"                          Rows Removed by Filter: 10301"
"                    ->  Materialize  (cost=16.67..24.69 rows=1 width=6) (actual time=0.293..0.293 rows=1 loops=1)"
"                          ->  Hash Semi Join  (cost=16.67..24.69 rows=1 width=6) (actual time=0.248..0.248 rows=1 loops=1)"
"                                Hash Cond: (m.co_microrregiao = mi.co_microrregiao)"
"                                ->  Index Scan using municipio_pkey on municipio m  (cost=0.28..8.30 rows=1 width=13) (actual time=0.008..0.008 rows=1 loops=1)"
"                                      Index Cond: (co_municipio = '3552205'::numeric)"
"                                ->  Hash  (cost=15.62..15.62 rows=61 width=7) (actual time=0.226..0.226 rows=63 loops=1)"
"                                      Buckets: 1024  Batches: 1  Memory Usage: 11kB"
"                                      ->  Hash Semi Join  (cost=3.90..15.62 rows=61 width=7) (actual time=0.069..0.205 rows=63 loops=1)"
"                                            Hash Cond: (mi.co_mesorregiao = me.co_mesorregiao)"
"                                            ->  Seq Scan on microrregiao mi  (cost=0.00..9.58 rows=558 width=12) (actual time=0.004..0.045 rows=558 loops=1)"
"                                            ->  Hash  (cost=3.71..3.71 rows=15 width=5) (actual time=0.040..0.040 rows=15 loops=1)"
"                                                  Buckets: 1024  Batches: 1  Memory Usage: 9kB"
"                                                  ->  Seq Scan on mesorregiao me  (cost=0.00..3.71 rows=15 width=5) (actual time=0.009..0.032 rows=15 loops=1)"
"                                                        Filter: (co_uf = '35'::numeric)"
"                                                        Rows Removed by Filter: 122"
"Planning time: 1.180 ms"
"Execution time: 498.556 ms"
```

Figura 4: Consulta 1 - Criação do Índice `nome_escola`

Houve uma queda de aproximadamente 3 segundos em comparação com a consulta anterior, em que não havia o índice. Este valor representa uma diferença de aproximadamente 14%.

Porém, podemos observar que mesmo com a criação do índice, este não foi utilizado. O otimizador de consultas do PostgreSQL achou melhor não utilizar o índice. Mesmo assim, podemos ver que ele modificou a ordem das operações e conseguiu uma queda de aproximadamente 3 segundos.

4.1.4 Troca de ILIKE por LIKE

A última técnica de otimização utilizada foi trocar o operador `ILIKE` pelo operador `LIKE`. O operador `ILIKE` não considera letras maiúsculas e minúsculas, fato que faz com que haja um aumento no tempo de execução da consulta.

CONSULTA SQL - LIKE

```
SELECT  e.co_escola, e.nome_escola, e.situacao_funcionamento,
     e.dependencia_adm, e.bercario, e.creche, e.pre_escola,
     e.ens_fundamental_anos_iniciais, e.ens_fundamental_anos_finais,
     e.ens_medio_normal, e.ens_medio_integrado
FROM escola e
WHERE e.co_distrito IN (
   SELECT d.co_distrito
   FROM distrito d
   WHERE d.co_municipio IN (
```

```
      SELECT m.co_municipio
      FROM municipio m
      WHERE m.co_microrregiao IN (
         SELECT mi.co_microrregiao
         FROM microrregiao mi
         WHERE mi.co_mesorregiao IN (
            SELECT me.co_mesorregiao
            FROM mesorregiao me
            WHERE me.co_uf = 35
         )
      )
   )
   AND d.co_municipio = 3552205
)
AND e.nome_escola LIKE '%EDU%'
ORDER BY e.qtd_funcionarios;
```

TEMPO DE EXECUÇÃO

2 secs 8 msec.
96 rows affected.

PLANO DE EXECUÇÃO

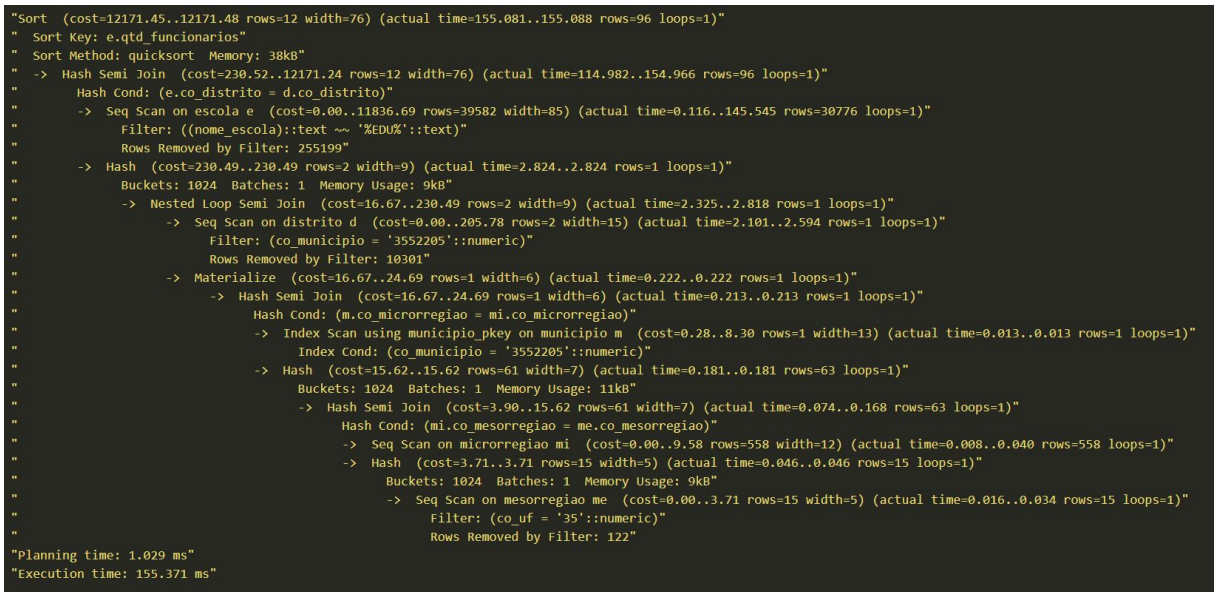

Figura 5: Consulta 1 - Operador LIKE

Podemos notar que ao realizarmos a mudança dos operadores, houve uma queda aproximadamente 14 segundo em relação ao tempo de execução da consulta anterior. Este tempo representa um diferença de aproximadamente 87%.

Apesar do plano de execução dos operadores serem iguais, como o operador `LIKE` leva em consideração letras minúsculas e maiúsculas, o otimizador conseguiu realizar a consulta muito mais rapidamente.

Na Tabela 1 podemos observar as diferenças, em porcentagem, dos tempos de execução de cada um dos testes realizados para a Consulta 1:

Tabela 1: Comparação Consulta 1

Testes	**Tempo (ms)**	**Diferença com Anterior (%)**	**Diferença com Original (%)**
Consulta Inicial (JOIN)	30246	-	-
IN	19094	36,87	36,87
ÍNDICE	16504	13,56	45,43
LIKE	2008	87,83	93,36

4.2 Consulta 2 - Quantidade de Escolas de cada Dependência Administrativa

A segunda consulta também foi inicialmente realizada com o operador `JOIN`. Nesta consulta foram utilizadas subconsultas correlatas na cláusula do `SELECT` para que fosse possível recuperar as quantidades de cada dependência de uma região em uma única consulta. Para fins de teste, utilizaremos o seguinte dado como parâmetro para realização das consultas:

- codigo_regiao = 3 (Sudeste)

4.2.1 Com JOIN

CONSULTA SQL - JOIN

```
SELECT count(e.co_escola) as qtd_escolas, (
     SELECT count(e2.co_escola)
     FROM escola e2
     WHERE e2.dependencia_adm = 'Federal'
  ) as qtd_federal, (
     SELECT count(e3.co_escola)
     FROM escola e3
     WHERE e3.dependencia_adm = 'Estadual'
  ) as qtd_estadual, (
     SELECT count(e4.co_escola)
     FROM escola e4
     WHERE e4.dependencia_adm = 'Municipal'
  ) as qtd_municipal, (
     SELECT count(e5.co_escola)
     FROM escola e5
     WHERE e5.dependencia_adm = 'Privada'
  ) as qtd_privada
```

```
FROM escola e
JOIN distrito d on e.co_distrito = d.co_distrito
JOIN municipio m on d.co_municipio = m.co_municipio
JOIN microrregiao m2 on m.co_microrregiao = m2.co_microrregiao
JOIN mesorregiao m3 on m2.co_mesorregiao = m3.co_mesorregiao
JOIN uf u on m3.co_uf = u.co_uf
JOIN regiao r on u.co_regiao = r.co_regiao
WHERE r.co_regiao = 3;
```

TEMPO DE EXECUÇÃO

2 secs 522 msec.
1 rows affected.

PLANO DE EXECUÇÃO

```
"Aggregate  (cost=60755.66..60755.67 rows=1 width=6) (actual time=548.040..548.040 rows=1 loops=1)"
"  InitPlan 1 (returns $0)"
"    ->  Aggregate  (cost=11838.29..11838.30 rows=1 width=6) (actual time=131.124..131.124 rows=1 loops=1)"
"          ->  Seq Scan on escola e2  (cost=0.00..11836.69 rows=639 width=6) (actual time=0.471..130.623 rows=795 loops=1)"
"                Filter: (dependencia_adm = 'Federal'::dependencia_adm_t)"
"                Rows Removed by Filter: 285180"
"  InitPlan 2 (returns $1)"
"    ->  Aggregate  (cost=11931.61..11931.62 rows=1 width=6) (actual time=81.856..81.856 rows=1 loops=1)"
"          ->  Seq Scan on escola e3  (cost=0.00..11836.69 rows=37968 width=6) (actual time=0.055..79.661 rows=37746 loops=1)"
"                Filter: (dependencia_adm = 'Estadual'::dependencia_adm_t)"
"                Rows Removed by Filter: 248229"
"  InitPlan 3 (returns $2)"
"    ->  Aggregate  (cost=12289.17..12289.18 rows=1 width=6) (actual time=95.282..95.283 rows=1 loops=1)"
"          ->  Seq Scan on escola e4  (cost=0.00..11836.69 rows=180994 width=6) (actual time=0.025..85.757 rows=181459 loops=1)"
"                Filter: (dependencia_adm = 'Municipal'::dependencia_adm_t)"
"                Rows Removed by Filter: 104516"
"  InitPlan 4 (returns $3)"
"    ->  Aggregate  (cost=12002.63..12002.64 rows=1 width=6) (actual time=85.683..85.683 rows=1 loops=1)"
"          ->  Seq Scan on escola e5  (cost=0.00..11836.69 rows=66375 width=6) (actual time=0.028..82.128 rows=65975 loops=1)"
"                Filter: (dependencia_adm = 'Privada'::dependencia_adm_t)"
"                Rows Removed by Filter: 220000"
"  ->  Hash Join  (cost=367.25..12667.45 rows=10592 width=6) (actual time=22.461..149.296 rows=90720 loops=1)"
"        Hash Cond: (e.co_distrito = d.co_distrito)"
"        ->  Seq Scan on escola e  (cost=0.00..11121.75 rows=285975 width=15) (actual time=0.048..76.124 rows=285975 loops=1)"
"        ->  Hash  (cost=362.48..362.48 rows=382 width=9) (actual time=9.882..9.882 rows=3248 loops=1)"
"              Buckets: 4096 (originally 1024)  Batches: 1 (originally 1)  Memory Usage: 163kB"
"              ->  Hash Join  (cost=140.02..362.48 rows=382 width=9) (actual time=5.528..8.763 rows=3248 loops=1)"
"                    Hash Cond: (d.co_municipio = m.co_municipio)"
"                    ->  Seq Scan on distrito d  (cost=0.00..180.02 rows=10302 width=15) (actual time=0.004..1.124 rows=10302 loops=1)"
"                    ->  Hash  (cost=137.44..137.44 rows=206 width=6) (actual time=4.323..4.323 rows=1668 loops=1)"
"                          Buckets: 2048 (originally 1024)  Batches: 1 (originally 1)  Memory Usage: 80kB"
"                          ->  Hash Join  (cost=18.75..137.44 rows=206 width=6) (actual time=1.082..3.451 rows=1668 loops=1)"
"                                Hash Cond: (m.co_microrregiao = m2.co_microrregiao)"
"                                ->  Seq Scan on municipio m  (cost=0.00..95.70 rows=5570 width=13) (actual time=0.004..0.984 rows=5570 loops=1)"
"                                ->  Hash  (cost=18.49..18.49 rows=21 width=7) (actual time=0.412..0.412 rows=160 loops=1)"
"                                      Buckets: 1024  Batches: 1  Memory Usage: 15kB"
"                                      ->  Nested Loop  (cost=5.35..18.49 rows=21 width=7) (actual time=0.156..0.366 rows=160 loops=1)"
"                                            ->  Seq Scan on regiao r  (cost=0.00..1.06 rows=1 width=10) (actual time=0.005..0.006 rows=1 loops=1)"
"                                                  Filter: (co_regiao = '3'::numeric)"
"                                                  Rows Removed by Filter: 4"
"                                            ->  Hash Join  (cost=5.35..17.22 rows=21 width=17) (actual time=0.137..0.326 rows=160 loops=1)"
"                                                  Hash Cond: (m2.co_mesorregiao = m3.co_mesorregiao)"
"                                                  ->  Seq Scan on microrregiao m2  (cost=0.00..9.58 rows=558 width=12) (actual time=0.004..0.055 rows=558 loops=1)"
"                                                  ->  Hash  (cost=5.28..5.28 rows=5 width=15) (actual time=0.108..0.108 rows=37 loops=1)"
"                                                        Buckets: 1024  Batches: 1  Memory Usage: 10kB"
"                                                        ->  Hash Join  (cost=1.35..5.28 rows=5 width=15) (actual time=0.049..0.094 rows=37 loops=1)"
"                                                              Hash Cond: (m3.co_uf = u.co_uf)"
"                                                              ->  Seq Scan on mesorregiao m3  (cost=0.00..3.37 rows=137 width=10) (actual time=0.003..0.015 rows=137 loops=1)"
"                                                              ->  Hash  (cost=1.34..1.34 rows=1 width=22) (actual time=0.020..0.020 rows=4 loops=1)"
"                                                                    Buckets: 1024  Batches: 1  Memory Usage: 9kB"
"                                                                    ->  Seq Scan on uf u  (cost=0.00..1.34 rows=1 width=22) (actual time=0.003..0.009 rows=4 loops=1)"
"                                                                          Filter: (co_regiao = '3'::numeric)"
"                                                                          Rows Removed by Filter: 23"
"Planning time: 2.101 ms"
"Execution time: 548.666 ms"
```

Figura 6: Consulta 2 - Original

Podemos observar que o plano de execução foi realizado baseado em agregação, ou seja, foi realizada uma agregação do retorno de cada subconsulta correlata com a junção

entre todas as tabelas para recuperar os dados da região especificada. O retorno de cada subconsulta correlata foi armazenado em uma variável, como se cada uma destas tivesse seu próprio plano de execução (InitPlan 1, 2, 3 e 4).

Apesar de todas as junções e agregações, o tempo de execução foi relativamente rápido, apenas 2 segundos e 522 milissegundos.

4.2.2 COM IN

Com intuíto de diminuir o tempo de execução modificamos a consulta para a utilizarmos o operador `IN` ao invés do operador `JOIN`, embora este já fosse bem pequeno.

CONSULTA SQL - IN

```
SELECT count(e.co_escola) as qtd_escolas, (
     SELECT count(e2.co_escola)
     FROM escola e2
     WHERE e2.dependencia_adm = 'Federal'
  ) as qtd_federal, (
     SELECT count(e3.co_escola)
     FROM escola e3
     WHERE e3.dependencia_adm = 'Estadual'
  ) as qtd_estadual, (
     SELECT count(e4.co_escola)
     FROM escola e4
     WHERE e4.dependencia_adm = 'Municipal'
  ) as qtd_municipal, (
     SELECT count(e5.co_escola)
     FROM escola e5
     WHERE e5.dependencia_adm = 'Privada'
  ) as qtd_privada
FROM escola e
WHERE e.co_distrito IN (
  SELECT d.co_distrito
  FROM distrito d
  WHERE d.co_municipio IN (
     SELECT m.co_municipio
     FROM municipio m
     WHERE m.co_microrregiao IN (
        SELECT mi.co_microrregiao
        FROM microrregiao mi
        WHERE mi.co_mesorregiao IN (
           SELECT me.co_mesorregiao
           FROM mesorregiao me
           WHERE me.co_uf IN (
              SELECT u.co_uf
              FROM uf u
              WHERE u.co_regiao = 3
```

```
            )
        )
    )
  )
);
```

TEMPO DE EXECUÇÃO

1 secs 679 msec.
1 rows affected.

PLANO DE EXECUÇÃO

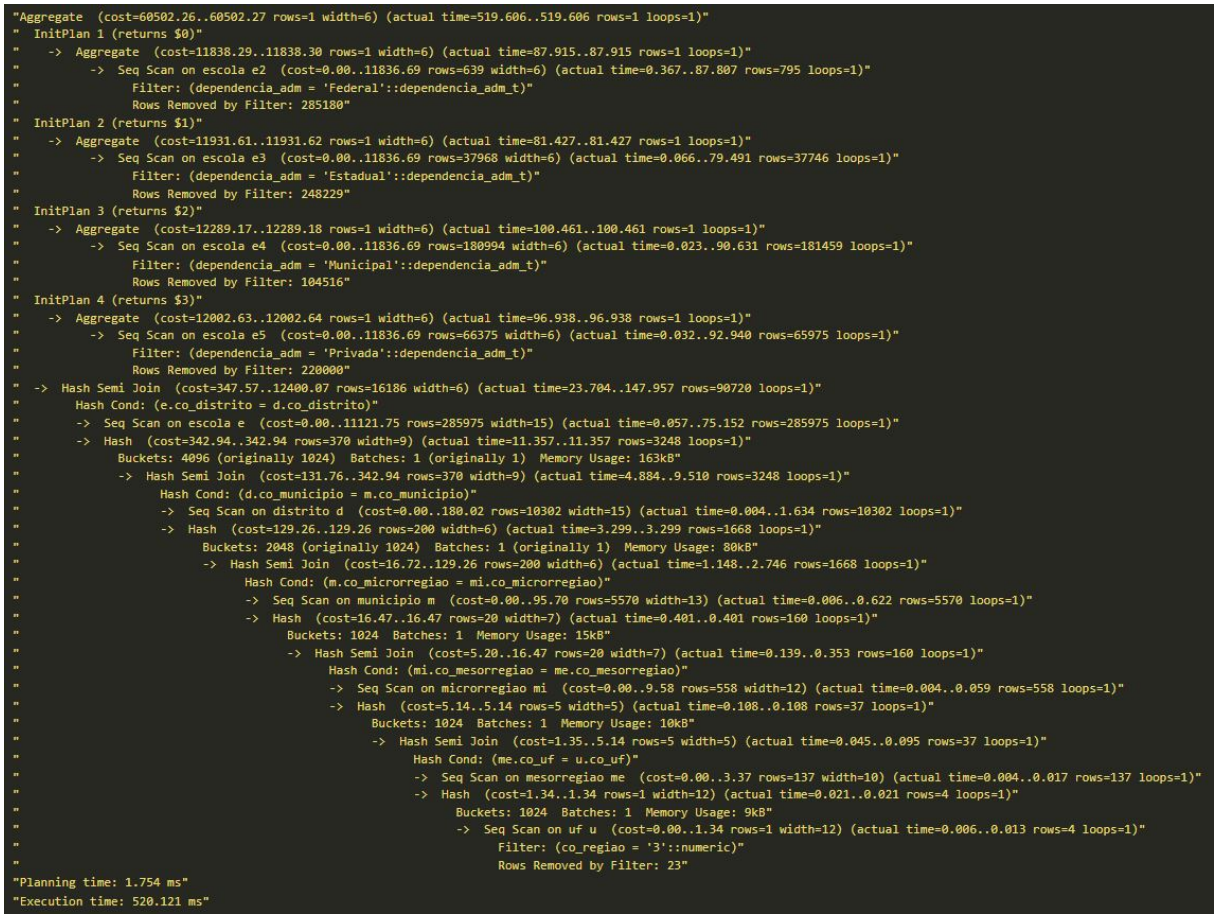

Figura 7: Consulta 2 - Operador IN

Podemos observar que o tempo de execução sofreu uma redução de aproximadamente 1 segundo. Queda significava se observarmos que este valor representa aproximadamente 33% do original.

Na Tabela 2 podemos observar as comparações entre os tempos de execução de cada um dos testes realizados para a consulta 2.

Tabela 2: Comparação Consulta 2

Testes	**Tempo (ms)**	**Diferença com Anterior (%)**	**Diferença com Original (%)**
Consulta Inicial (JOIN)	2522	-	-
IN	1679	33,55	33,55