UNIVERSIDADE FEDERAL DE SÃO CARLOS
CAMPUS SOROCABA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

SISTEMAS DE BANCO DE DADOS
Prof. Sahudy Montenegro González

PROJETO INTEGRADO
Fase Final

BASE DE DADOS BRASILEIRA
TEMA 4 - Cadastro Brasileiro de Escolas

Bianca Gomes Rodrigues - 743512
Pietro Zuntini Bonfim - 743588

Sorocaba-SP
12 de Junho 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.

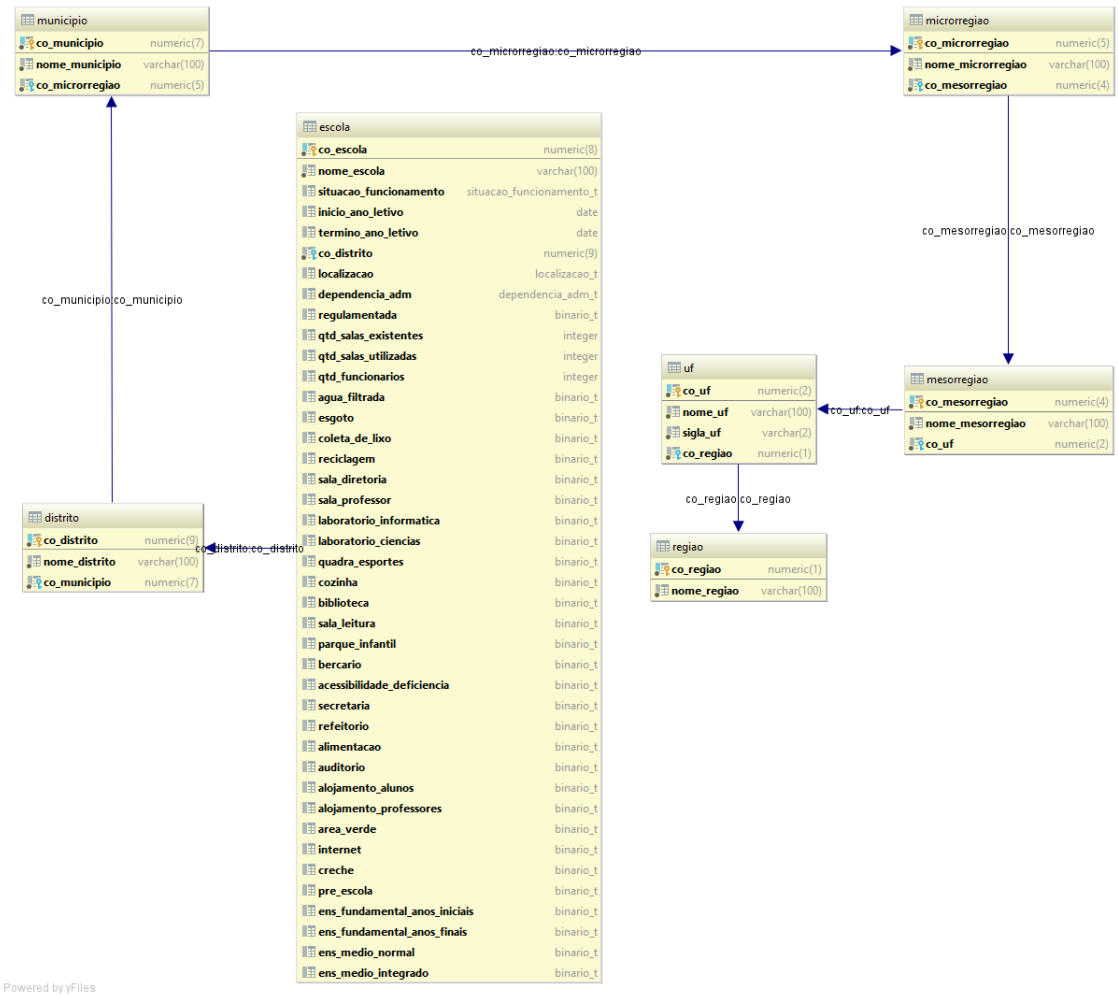

Figura 1: Diagrama do BD

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 a 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 de uma região com cada tipo de situação de funcionamento, dependência administrativa e oferta de matrícula.

   Campos de busca: Código da Região.

   Campos de visualização do resultado: Quantidade de Escolas na Região, e desta região a quantidade de escolas Em Atividade, Paralisada, Extinta, Federal, Estadual, Municipal, Privada e com cada tipo de oferta de matrícula.

   Operadores das condições: Código da Região (=).

   SQL:

```
SELECT
  GROUPING(e.bercario) g_bercario, GROUPING(e.creche) g_creche,
  GROUPING(e.pre_escola) g_pe,
  GROUPING(e.ens_fundamental_anos_iniciais) g_efi,
  GROUPING(e.ens_fundamental_anos_finais) g_efii,
  GROUPING(e.ens_medio_normal) g_emn,
  GROUPING(e.ens_medio_integrado) g_emi,
  GROUPING(e.situacao_funcionamento) g_situacao,
  GROUPING(e.dependencia_adm) g_dep,
```

```
    GROUPING(e.localizacao) g_localizacao,
    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,e.situacao_funcionamento, e.dependencia_adm,
    e.localizacao, count(e.co_escola) as qtd_escolas
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>;
GROUP BY GROUPING SETS (
    (e.situacao_funcionamento),
    (e.dependencia_adm),
    (e.localizacao),
    (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),
    ()
);
```

4 POPULANDO O BD

Nesta seção será apresentado o modo como foi realizada a adição de dados ao banco de dados, bem como algumas informações técnicas de cada tabela. Vamos começar falando sobre de onde os dados foram retirados. Os dados do banco de escolas brasileiras foram retirados do Microdados do INEP do Censo Escolar 2018 (http://inep.gov.br/microdados . Como os dados vêm em uma tabela no excel, foi criado um script em PHP percorrendo a tabela e inserindo os dados no banco. Foram criados outros scripts também sobre outras tabelas auxiliares para a extração das informações dos Distritos, Municípios, Microrregiões, Mesorregiões, Estados e Regiões.

Quanto ao volume de dados final do banco, ou seja, o número de tuplas e o total em bytes de cada tabela, podemos visualizar na tabela abaixo:

Tabela 1: Informações Tabelas

Nome da Tabela	Nº de Tuplas	Tamanho
Escola	285975	58 MB
Distrito	10302	976 kB
Municipio	5570	744 kB
Microrregiao	558	88 kB
Mesorregiao	137	56 kB
Uf	27	24 kB
Regiao	5	24 kB
Total	302574	59,86 MB

5 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. As consultas foram executadas cinco vezes.

5.1 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')

5.1.1 Operador 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`

5.1.2 Operador 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.
96 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.

5.1.3 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 execução:

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.

5.1.4 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

```
"Sort  (cost=12171.45..12171.48 rows=12 width=76) (actual time=155.081..155.088 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=114.982..154.966 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=0.116..145.545 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=2.824..2.824 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=2.325..2.818 rows=1 loops=1)"
"                    ->  Seq Scan on distrito d  (cost=0.00..205.78 rows=2 width=15) (actual time=2.101..2.594 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=0.222..0.222 rows=1 loops=1)"
"                          ->  Hash Semi Join  (cost=16.67..24.69 rows=1 width=6) (actual time=0.213..0.213 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=0.013..0.013 rows=1 loops=1)"
"                                      Index Cond: (co_municipio = '3552205'::numeric)"
"                                ->  Hash  (cost=15.62..15.62 rows=61 width=7) (actual time=0.181..0.181 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=0.074..0.168 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=0.008..0.040 rows=558 loops=1)"
"                                            ->  Hash  (cost=3.71..3.71 rows=15 width=5) (actual time=0.046..0.046 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=0.016..0.034 rows=15 loops=1)"
"                                                        Filter: (co_uf = '35'::numeric)"
"                                                        Rows Removed by Filter: 122"
"Planning time: 1.029 ms"
"Execution time: 155.371 ms"
```

Figura 5: Consulta 1 - Operador LIKE

Podemos notar que ao realizarmos a mudança dos operadores, houve uma queda aproximadamente 14 segundos 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.

5.1.5 Materialized Views

Outra tentativa de otimização foi realizar a criação de views materializadas, ou seja, que ficam em armazenadas em disco. Deste modo, uma MATERIALIZED VIEW foi criada para cada estado da tabela Estado.

Para conseguir fazer isto, foi criado um script na linguagem PHP para criar, para cada estado, uma view contendo os códigos dos distritos daquele estado. Foi dado o nome distrito<codigo_estado>.

Criadas as views, basta realizarmos uma seleção dos códigos dos distritos presentes na view específica do estado desejado, não precisando mais percorrer todos os IN's da consulta anterior.

CONSULTA SQL - COM MATERIALIZED VIEW

```
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 co_distrito
   FROM distritos35 d
   WHERE d.co_municipio = 3552205
)
AND e.nome_escola LIKE '%EDU%'
ORDER BY e.qtd_funcionarios;
```

TEMPO DE EXECUÇÃO

1 secs 613 msec.

PLANO DE EXECUÇÃO

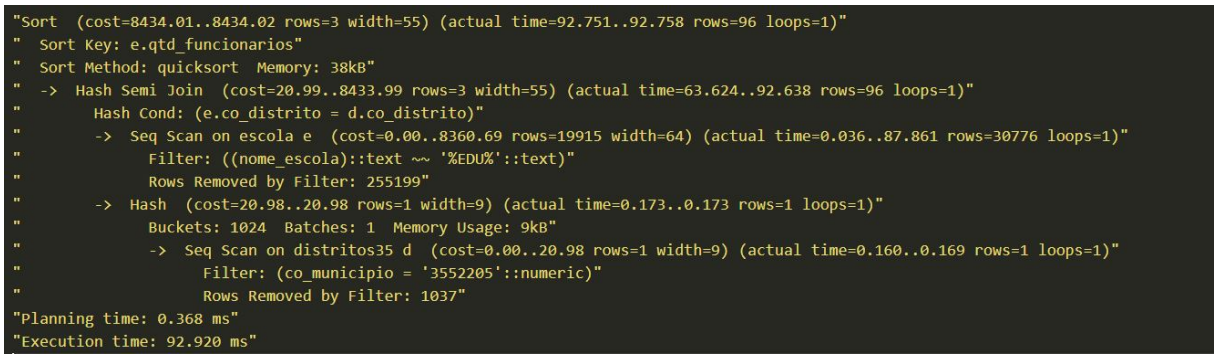

Figura 6: Consulta 1 - com MATERIALIZED VIEW

Podemos observar que o tempo de execução caiu, porém não significativamente, apenas 395ms. Pelo plano de consulta podemos verificar que o otimizador realizou uma busca sequencial no código do distrito na view de distritos e comparou com o código do distrito na tabela escola. Por fim, realizou uma ordenação com quicksort pela quantidade de funcionários.

5.1.6 Criação do Índice em Escola(Código Distrito)

A última otimização realizada foi a criação de um índice na coluna `co_distrito` da tabela `Escola`. A SQL utilizada para a criação do índice e o tempo de execução podem ser observados a seguir:

```
CREATE INDEX e_co_distrito_index
ON escola(co_distrito)
3 secs 607 msec.
```

Realizando novamente a mesma consulta do item anterior, observamos os seguintes resultados:

TEMPO DE EXECUÇÃO

482 msec.

PLANO DE EXECUÇÃO

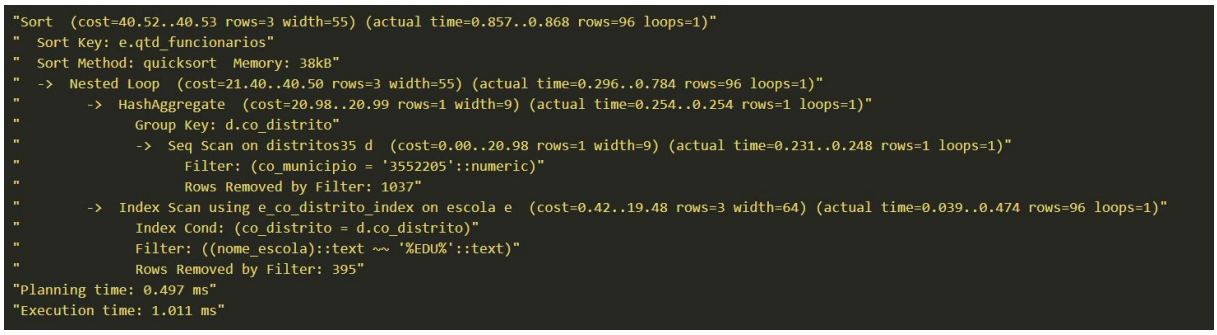

Figura 7: Consulta 1 - com Índice no Código do Distrito

Como podemos observar, o tempo de execução caiu para apenas 482 ms! Uma diminuição de 70% do valor anterior, que ainda não continha o índice. Fica clara esta diferença ao visualizar o plano de execução, uma vez que o otimizador utilizou o novo índice criado.

5.1.7 Conclusão - Consulta 1

Na Tabela 1 podemos observar as diferenças, em porcentagem, dos tempos de execução de cada um dos testes realizados para a Consulta 1, e observar que conseguimos uma melhora final de 98,4% em relação ao tempo inicial:

Tabela 2: 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
MATERIALIZED VIEW	1613	19,67	94,66
ÍNDICE (co_distrito)	482	70,11	98,4

5.2 Consulta 2 - Quantidade de Escolas de cada Dependência Administrativa

A segunda consulta também foi inicialmente realizada com o operador `JOIN`. Nesta consulta foi utilizada a cláusula `GROUP BY GROUPING SETS`, a qual realiza o agrupamento dos resultados de acordo com o específicado (no caso por situação de funcionamento, dependência administrativa e cada tipo de oferta de matrícula). O parênteses vazio na última linha do `GROUPING SETS` serve para recuperar o total de escolas. Para fins de teste, utilizaremos o seguinte dado como parâmetro para realização das consultas:

- codigo_regiao = 3 (Sudeste)

5.2.1 Operador JOIN

O primeiro teste foi realizado com o operador `JOIN`.

CONSULTA SQL - JOIN

```
SELECT
   GROUPING(e.bercario) g_bercario, GROUPING(e.creche) g_creche,
   GROUPING(e.pre_escola) g_pe,
   GROUPING(e.ens_fundamental_anos_iniciais) g_efi,
   GROUPING(e.ens_fundamental_anos_finais) g_efii,
   GROUPING(e.ens_medio_normal) g_emn,
   GROUPING(e.ens_medio_integrado) g_emi,
   GROUPING(e.situacao_funcionamento) g_situacao,
   GROUPING(e.dependencia_adm) g_dep,
   GROUPING(e.localizacao) g_localizacao,
   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,e.situacao_funcionamento, e.dependencia_adm,
   e.localizacao, count(e.co_escola) as qtd_escolas
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;
GROUP BY GROUPING SETS (
   (e.situacao_funcionamento),
   (e.dependencia_adm),
   (e.localizacao),
   (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),
   ()
);
```

TEMPO DE EXECUÇÃO

4 secs 206 msec.

PLANO DE EXECUÇÃO

Figura 8: Consulta 2 - Original

Podemos observar no plano de execução que o índice criado no código do `Distrito` (`e_co_distrito_index` para otimizar a consulta um, também foi utilizado por esta consulta. Logo, este índice também foi responsável pelo baixo tempo de execução inicial desta consulta.

5.2.2 Operador IN

Com intuíto de diminuir o tempo de execução modificamos a consulta para a utilizarmos o operador `IN` ao invés do operador `JOIN`.

CONSULTA SQL - IN

```
SELECT
   GROUPING(e.bercario) g_bercario, GROUPING(e.creche) g_creche,
   GROUPING(e.pre_escola) g_pe,
   GROUPING(e.ens_fundamental_anos_iniciais) g_efi,
   GROUPING(e.ens_fundamental_anos_finais) g_efii,
   GROUPING(e.ens_medio_normal) g_emn,
   GROUPING(e.ens_medio_integrado) g_emi,
   GROUPING(e.situacao_funcionamento) g_situacao,
```

```
  GROUPING(e.dependencia_adm) g_dep,
  GROUPING(e.localizacao) g_localizacao,
  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,e.situacao_funcionamento, e.dependencia_adm,
  e.localizacao, count(e.co_escola) as qtd_escolas
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;
        )
      )
    )
  )
)
GROUP BY GROUPING SETS (
  (e.situacao_funcionamento),
  (e.dependencia_adm),
  (e.localizacao),
  (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),
  ()
);
```

TEMPO DE EXECUÇÃO

2 secs 476 msec.

PLANO DE EXECUÇÃO

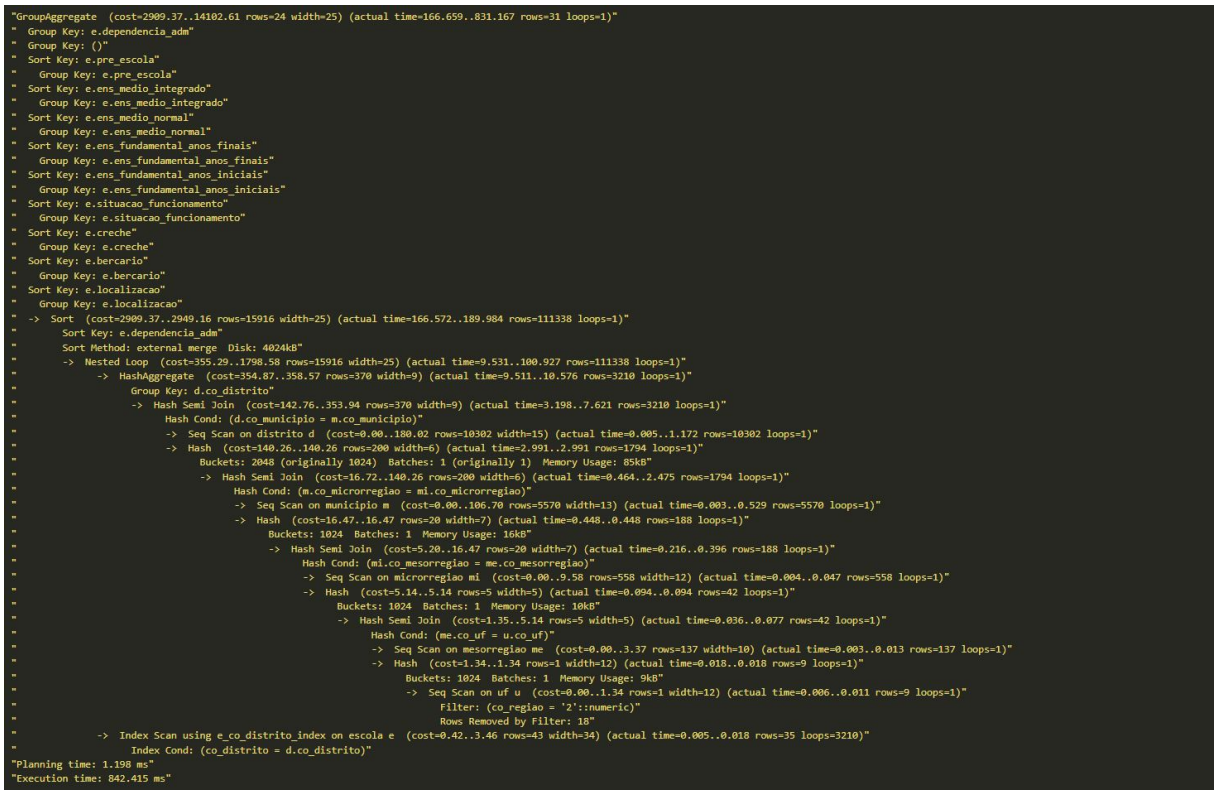

Figura 9: Consulta 2 - Operador IN

Podemos observar que o tempo de execução sofreu uma redução de 1 sec 730 msec. Queda significava se observarmos que este valor representa uma redução de aproximadamente 41% em relação ao tempo anterior.

5.2.3 Materialized View

Observamos que a criação de views com os códigos dos distritos de cada estado foram responsáveis por uma melhoria significativa no tempo de execução da consulta 1. Logo, a última otimização realizada para a segunda consulta foi a criação de `Materialized Views` contendo os distritos (código e nome do distrito, além do código de seu município) de cada região. O intuito desta alteração foi reduzir ainda mais o tempo da consulta, mesmo que já pequeno, visto que para a aplicação quanto mais rápido melhor.

CONSULTA SQL - View

```
SELECT
SELECT
   GROUPING(e.bercario) g_bercario, GROUPING(e.creche) g_creche,
   GROUPING(e.pre_escola) g_pe,
   GROUPING(e.ens_fundamental_anos_iniciais) g_efi,
   GROUPING(e.ens_fundamental_anos_finais) g_efii,
   GROUPING(e.ens_medio_normal) g_emn,
   GROUPING(e.ens_medio_integrado) g_emi,
```

```
  GROUPING(e.situacao_funcionamento) g_situacao,
  GROUPING(e.dependencia_adm) g_dep,
  GROUPING(e.localizacao) g_localizacao,
  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,e.situacao_funcionamento, e.dependencia_adm,
  e.localizacao, count(e.co_escola) as qtd_escolas
FROM escola e
WHERE e.co_distrito IN (
  SELECT d.co_distrito
  FROM distritos_regiao3 d
)
GROUP BY GROUPING SETS (
  (e.situacao_funcionamento),
  (e.dependencia_adm),
  (e.localizacao),
  (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),
  ()
);
```

TEMPO DE EXECUÇÃO

1 secs 644 msec.

PLANO DE EXECUÇÃO

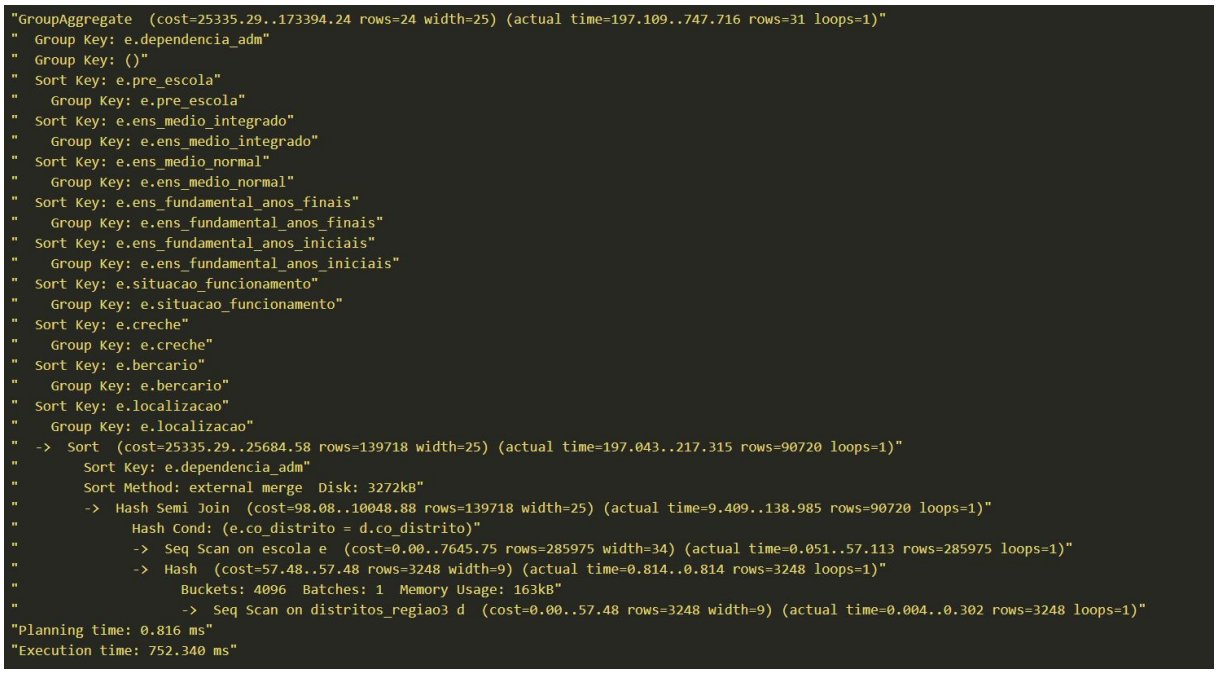

Figura 10: Consulta 2 - View

Podemos observar pelo plano de execução que este ficou bem mais simples em relação aos outros, visto que agora bastou uma busca sequencial na view da região especificada, sem ter que realizar todos os joins com as outras tabelas.

5.2.4 Conclusão - Consulta 2

Com as otimizações realizadas, conseguimos uma melhora total de 60,91% no tempo de execução da consulta. Na Tabela 2 podemos observar as comparações entre os tempos de execução de cada um dos testes realizados para a consulta 2:

Tabela 3: Comparação - Consulta 2

Testes	**Tempo (ms)**	**Diferença com Anterior (%)**	**Diferença com Original (%)**
Consulta Inicial (JOIN)	4206	-	-
IN	2476	41,13	41,13
MATERIALIZED VIEW	1644	33,6	60,91

6 Segurança e Controle de Acesso

Nesta seção iremos mostrar e analisar as técnicas utilizadas no banco para segurança e controle de acesso aos dados. Iremos citar as técnicas utilizadas para prevenção de SQL Injection e os usuários criados para o controle das operações permitidas sobre o banco das escolas.

6.1 Prevenção de SQL Injection

Para a prevenção de SQL Injection, foram utilizadas basicamente duas técnicas: a utilização de `stored procedures` e de `prepared statements` para tratamento das consultas SQL.

PROCEDURES

A primeira *stored procedure* foi criada para utilização da consulta 1, que recupera as informações das escolas. Essa consulta também envolve, na aplicação, paginação, deslocamento e ordenação. O procedimento pode ser visualizado a seguir:

```
01 | CREATE OR REPLACE FUNCTION recuperarEscolas( campoBusca varchar(100),
       codigoEstado varchar(2), limite int, deslocamento int, ordColuna int,
       ordDirecao varchar(4) )
02 | RETURNS TABLE (
03 |     co_escola escola.co_escola%TYPE, nome_escola escola.nome_escola%TYPE,
04 |     situacao_funcionamento escola.situacao_funcionamento%TYPE,
05 |     dependencia_adm escola.dependencia_adm%TYPE,
06 |     bercario escola.bercario%TYPE,
07 |     creche escola.creche%TYPE, pre_escola escola.pre_escola%TYPE,
08 |     ens_fundamental_anos_iniciais escola.ens_fundamental_anos_iniciais%TYPE,
09 |     ens_fundamental_anos_finais escola.ens_fundamental_anos_finais%TYPE,
10 |     ens_medio_normal escola.ens_medio_normal%TYPE,
11 |     ens_medio_integrado escola.ens_medio_integrado%TYPE
12 | ) AS $$
13 | DECLARE
14 |     sql text := 'SELECT
15 |         e.co_escola, e.nome_escola, e.situacao_funcionamento, e.
       dependencia_adm, e.bercario, e.creche, e.pre_escola,
16 |         e.ens_fundamental_anos_iniciais, e.ens_fundamental_anos_finais, e.
       ens_medio_normal, e.ens_medio_integrado
17 |     FROM escola e ';
18 | BEGIN
19 |
20 |     sql := sql || 'WHERE true ';
21 |
22 |     IF codigoEstado IS NOT NULL THEN
23 |         -- Filtra apenas os distritos do estado especificado
24 |         sql := sql || 'AND co_distrito IN (
25 |             SELECT co_distrito
26 |             FROM distritos' || codigoEstado || ' d)';
27 |     END IF;
28 |
29 |     -- Campo de Busca
30 |     sql := sql || '
31 |         AND e.nome_escola LIKE ' || quote_literal('%' || campoBusca || '%') ||
       '
32 |         ORDER BY ';
33 |
34 |     -- Ordenacao
35 |     CASE ordColuna
36 |         WHEN 0 THEN
37 |             sql := sql || 'e.co_escola ';
38 |         WHEN 1 THEN
```

```
39 |             sql := sql || 'e.nome_escola ';
40 |         WHEN 2 THEN
41 |             sql := sql || 'e.situacao_funcionamento ';
42 |         WHEN 3 THEN
43 |             sql := sql || 'e.dependencia_adm ';
44 |     END CASE;
45 |
46 |     -- Limite e Deslocamento
47 |     sql := sql || ordDirecao || '
48 |         LIMIT ' || limite || ' OFFSET ' || deslocamento;
49 |
50 |     -- Retorna a Query
51 |     RETURN QUERY EXECUTE sql;
52 | END
53 | $$ LANGUAGE plpgsql;
```

Os exemplos de chamada da função seriam:

```
01 | -- Exemplos de chamada
02 | SELECT * FROM recuperarEscolas('UNIDADE', NULL, 100, 0, 1, 'asc');  -- Brasil
03 | SELECT * FROM recuperarEscolas('UNIDADE', '35', 10, 100, 1, 'asc'); -- SP
```

A segunda *stored procedure* criada foi para a consulta 2, ou seja, consulta para recuperar as estatísicas. Criamos uma procedure capaz de lidar com todos os casos possíveis de consulta na aplicação: estatíticas de todo o Brasil, de uma região, de um estado, ou de um município específicos. A procedure pode ser visualizada no código a seguir:

```
01 | CREATE OR REPLACE FUNCTION recuperarEstatisticas( brasil boolean, codigoRegiao
       varchar(1), codigoEstado varchar(2), codigoMunicipio varchar(7) )
02 | RETURNS TABLE (
03 |     g_bercario integer, g_creche integer, g_pe integer, g_efi integer,
04 |     g_efii integer, g_emn integer, g_emi integer, g_situacao integer,
05 |     g_dep integer, g_localizacao integer, bercario escola.bercario%TYPE,
06 |     creche escola.creche%TYPE, pre_escola escola.pre_escola%TYPE,
07 |     ens_fundamental_anos_iniciais escola.ens_fundamental_anos_iniciais%TYPE,
08 |     ens_fundamental_anos_finais escola.ens_fundamental_anos_finais%TYPE,
09 |     ens_medio_normal escola.ens_medio_normal%TYPE,
10 |     ens_medio_integrado escola.ens_medio_integrado%TYPE,
11 |     situacao_funcionamento escola.situacao_funcionamento%TYPE,
12 |     dependencia_adm escola.dependencia_adm%TYPE, localizacao escola.
       localizacao%TYPE,
13 |     qtd_escolas bigint
14 | ) AS $$
15 | DECLARE
16 |     sql text := 'SELECT
17 |         GROUPING(e.bercario) g_bercario, GROUPING(e.creche) g_creche,
18 |         GROUPING(e.pre_escola) g_pe,
19 |         GROUPING(e.ens_fundamental_anos_iniciais) g_efi,
20 |         GROUPING(e.ens_fundamental_anos_finais) g_efii,
21 |         GROUPING(e.ens_medio_normal) g_emn,
22 |         GROUPING(e.ens_medio_integrado) g_emi,
23 |         GROUPING(e.situacao_funcionamento) g_situacao,
24 |         GROUPING(e.dependencia_adm) g_dep,
```

```
25 |          GROUPING(e.localizacao) g_localizacao,
26 |          e.bercario, e.creche, e.pre_escola, e.ens_fundamental_anos_iniciais,
27 |          e.ens_fundamental_anos_finais, e.ens_medio_normal,
28 |          e.ens_medio_integrado,e.situacao_funcionamento,
29 |          e.dependencia_adm, e.localizacao, count(e.co_escola) as qtd_escolas
30 |      FROM escola e';
31 |      g_text text := ' GROUP BY GROUPING SETS (
32 |          (e.situacao_funcionamento),
33 |          (e.dependencia_adm),
34 |          (e.localizacao),
35 |          (e.bercario),
36 |          (e.creche),
37 |          (e.pre_escola),
38 |          (e.ens_fundamental_anos_iniciais),
39 |          (e.ens_fundamental_anos_finais),
40 |          (e.ens_medio_normal),
41 |          (e.ens_medio_integrado),
42 |          ()
43 |      )';
44 |  BEGIN
45 |      IF brasil IS NULL THEN
46 |
47 |          IF codigoRegiao IS NOT NULL THEN
48 |
49 |              -- SQL Regiao
50 |              sql := sql || ' WHERE e.co_distrito IN (
51 |                  SELECT d.co_distrito
52 |                  FROM distritos_regiao' || $2 || ' d)';
53 |
54 |              RETURN QUERY EXECUTE sql || g_text
55 |              USING codigoRegiao;
56 |
57 |          ELSIF codigoEstado IS NOT NULL THEN
58 |
59 |              -- SQL Estado
60 |              sql := sql || ' WHERE e.co_distrito IN (
61 |                      SELECT d.co_distrito
62 |                      FROM distritos' || $3 || ' d)';
63 |
64 |              RETURN QUERY EXECUTE sql || g_text
65 |              USING codigoEstado;
66 |
67 |
68 |          ELSIF codigoMunicipio IS NOT NULL THEN
69 |
70 |              -- SQL Municipio
71 |              sql := sql || ' WHERE e.co_distrito IN (
72 |                      SELECT d.co_distrito
73 |                      FROM distrito d
74 |                      WHERE d.co_municipio = ' || $4 || ')';
75 |
76 |              RETURN QUERY EXECUTE sql || g_text
77 |              USING codigoMunicipio;
78 |
79 |          END IF;
80 |
81 |      ELSE
```

```
82 |         -- SQL Brasil
83 |         RETURN QUERY EXECUTE sql || g_text;
84 |     END IF;
85 |
86 | END
87 | $$ LANGUAGE plpgsql;
```

Os exemplos de chamada da função seriam:

```
01 | -- Exemplos de chamada
02 | SELECT * FROM recuperarEstatisticas(true, NULL, NULL, NULL);      -- Brasil
03 | SELECT * FROM recuperarEstatisticas(NULL, '4', NULL, NULL);       -- Regiao
04 | SELECT * FROM recuperarEstatisticas(NULL, NULL, '35', NULL);      -- Estado
05 | SELECT * FROM recuperarEstatisticas(NULL, NULL, NULL, '3552205'); -- Municipio
```

6.2 Usuários para Controle de Acesso

Foram criados três usuários para realizar o controle de acesso ao bando de dados. Os usuários criados foram:

- `admin`
- `gerente`
- `diretor`
- `cliente`

O `admin` possui todos os privilégios, ou seja, os privilégios de seleção, inserção, atualização e dele.ão sobre todas as tabelas d Abancolém disso, foi çespecifio cado o privilégio `CREATEROLE`, assim o `admin` pode criar outros usuários. Também foi especificado `GRANT OPTION`, deste modo, ele recebe o privilégio de conceder privilégios a outros usuários.

À partir do `admin` foram criados os usuários `gerente`, `diretor` e `cliente`. Ao gerente foram dados os privilégios de seleção e atualização sobre todas as tabelas. Ao `diretor` foram concedidos os privilégios de seleção e inserção sobre todas as tabelas.

Por fim, ao `cliente` foi apenas dado o privilégio de seleção sobre as tabelas. É importante ressaltar que a todos os três usuários criados foi especificada a opção `NOINHERIT`, ou seja, não herdam nenhum dos privilégios de cima.

Para melhor visualizar os usuários criados, bem como cada privilégio concedido a cada um, podemos observar a tabela abaixo:

Além disso, foi criado também o diagrama de concessões, que nos permite visualizar melhor ainda os privilégios de cada usuário, bem como quem os concedeu:

Tabela 4: Privilégios Usuários

	admin	**diretor**	**gerente**	**cliente**
escola	S,I,U,D	S,I	S,U	S
distrito	S,I,U,D	S,I	S,U	S
municipio	S,I,U,D	S,I	S,U	S
microrregiao	S,I,U,D	S,I	S,U	S
mesorregiao	S,I,U,D	S,I	S,U	S
uf	S,I,U,D	S,I	S,U	S
regiao	S,I,U,D	S,I	S,U	S

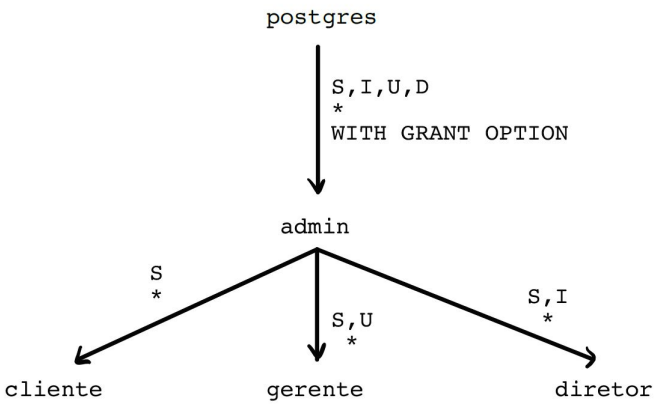

Figura 11: Diagrama de Concessões