Universidade Federal de São Carlos
Campus Sorocaba

Projeto Integrado
Laboratório de Banco de Dados

Professora: Sahudy Montenegro González

Grupo 9 - Tema 5 (Museus Brasileiros)

Daniel Davoli - 610372

Renato Araujo - 587788

Vitoria Rodrigues Silva - 726598

Maio
2019

Sumário

1 Introdução

O projeto integrado tem como principal objetivo o desenvolvimento de um sistema para a compreensão e construção de técnicas eficientes de manipulação de dados. O sistema de banco de dados tem como tema o registro de museus brasileiros.

2 Minimundo

O Brasil possui museus em todas as suas regiões, criados em diversos anos de sua história, seus acervos são de diferentes tipologias (antropologia e etnografia, arqueologia, artes visuais, ciências naturais, história, imagem e som, arquivístico, biblioteconômico, documental, entre outras) e podem ser administrados tanto em âmbito público, privado ou misto. Todo museu possui diferentes tipos de serviços e instalações oferecidas para facilitar acessibilidade e mobilidade para o público em geral e para portadores de deficiência física, visual e auditiva. Cada museu também possui um plano de segurança projetado para os casos de emergência ou desastre.

A partir do armazenamento dessas características em um banco de dados relacional, é possível obter informações relevantes sobre o que há de melhor ou pior nos museus registrados, o que torna possível até mesmo propor soluções e melhorias para o segundo caso.

3 Modelo relacional

O modelo relacional descreve as tabelas geradas a partir do minimundo e a relação entre elas. A partir dele, serão realizadas as consultas. O modelo criado para o cadastro de meseus brasileiros está descrito abaixo:

Museu (nome_museu, ano_criacao, endereco, estado, esfera, telefone, site, email, cobranca_ingresso, horarios)

TipologiaDoAcervo (nome_tipologia, descricao)

Tipologia_Museu (nome_tipologia, nome_museu)

nome_museu referencia Museu(nome_museu)

nome_tipologia referencia TipologiaDoAcervo(nome_tipologia)

Acessibilidade (nome_acessibilidade, descricao)

Acessibilidade_Museu (nome_acessibilidade, nome_museu)

nome_museu referencia Museu(nome_museu)

nome_acessibilidade referencia Acessibilidade(nome_acessibilidade)

PlanoDeSeguranca (nome_plano, descricao)

Seguranca_Museu (nome_plano, nome_museu)

nome_museu referencia nome_museu(Museu)

nome_plano referencia PlanoDeSeguranca(nome_plano)

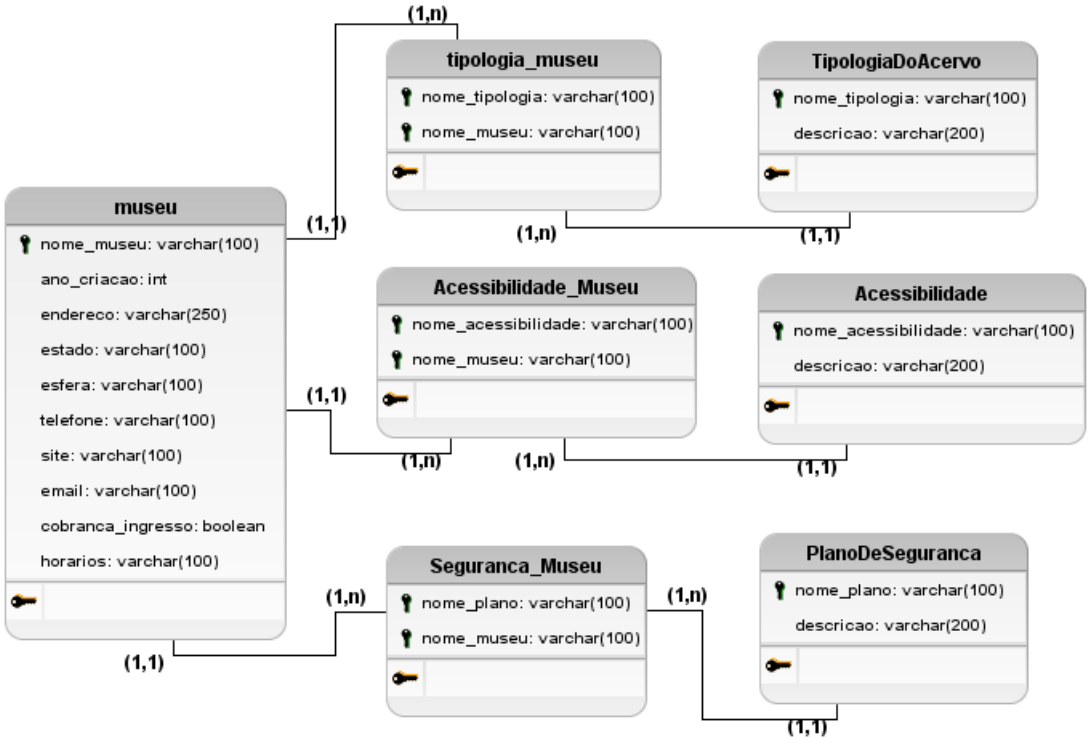

Figura 1: Diagrama do Modelo Relacional

4 Definição das Consultas em SQL

A consultas a seguir são baseadas no modelo relacional apresentado na secção anterior.

4.1 Consulta 1

Listar a quantidade de serviços e instalações de acessibilidade dos museus por ordem decrescente, dado uma tipologia.

SQL:

```
WITH acess AS (
    SELECT nome_museu, count(*) AS qntidade_acessibilidade
    FROM acessibilidade_museu GROUP BY nome_museu
), segur AS (
    SELECT nome_museu, count(*) AS qntidade_seguranca FROM
    seguranca_museu GROUP BY nome_museu LIMIT(50000);
), tipo AS (
    SELECT nome_museu, nome_tipologia FROM tipologia_museu
    WHERE nome_tipologia = 'História'
)
SELECT museus.nome_museu, tipo.nome_tipologia, esfera, ano_criacao,
qntidade_acessibilidade, qntidade_seguranca FROM
museu
NATURAL JOIN tipo
NATURAL JOIN acess
NATURAL JOIN segur
ORDER BY (qntidade_acessibilidade, qntidade_seguranca) DESC
```

Campos de visualização do resultado:

nome_museu, nome_tipologia,
esfera, ano_criacao, qtd_acessibilidade, qntidade_seguranca

Campos de busca (ou das condições):

nome_tipologia (absoluta).

Operadores das condições:

nome_tipologia (=)

4.2 Consulta 2

Listar os museus dados um estado, uma quantidade mínima de planos de segurança e uma quantidade mínima de serviços de acessibilidade.

SQL:

WITH acess AS (

SELECT nome_museu, count(*) AS qntidade_acessibilidade
FROM acessibilidade_museu GROUP BY nome_museu

), segur AS (

SELECT nome_museu, count(*) AS qntidade_seguranca FROM
seguranca_museu GROUP BY nome_museu LIMIT(50000);

) SELECT nome_museu, estado, esfera, qntidade_acessibilidade, qntidade_seguranca FROM
museu
NATURAL JOIN acess
NATURAL JOIN segur
WHERE qntidade_acessibilidade > <acess_minima> AND qntidade_seguranca > <seguranca_minima> AND estado ILIKE <sigla_estado>

Campos de visualização do resultado:

nome_museu, estado, esfera,
qtd_seguranca, qtd_acessibilidade

Campos de busca (ou das condições):

estado (relativo),
acess_minima (minimo),
seguranca_minima (minimo)

Operadores das condições:

estado (ILIKE), qntidade_seguranca (>),
qntidade_acessibilidade(>)

5 Populando o Banco de Dados

Para a geração das tuplas que foram inseridas no banco de dados, um *script* próprio foi criado. O *script* de geração dados (`gerador-dados.html`) foi desenvolvido com uma série de funções em javascript que geram dados aleatórios para cada campo das tabelas descritas no esquema do BD (descrito em detalhes na seção 3 deste relatório). O *backup* do BD está disponível no arquivo `bkp.backup`.

O quadro abaixo contém informações sobre as tabelas do banco de dados. As funções `pg_relation_size` e `pg_size_pretty` foram utilizadas para obter o tamanho de cada tabela.

NOME DA TABELA	NÚMERO DE REGISTROS	TAMANHO (em bytes)	CONSULTA (Postgresql)
museu	100.000	23MB	SELECT pg_size_pretty (pg_relation_size('museu'));
acessibilidade	10	8192 bytes	SELECT pg_size_pretty (pg_relation_size('acessibilidade'));
acessibilidade_museu	409.151	49 MB	SELECT pg_size_pretty (pg_relation_size('acessibilidade_museu'));
planodeseguranca	5	8192 bytes	SELECT pg_size_pretty (pg_relation_size('planodeseguranca'));
seguranca_museu	167.582	22 MB	SELECT pg_size_pretty (pg_relation_size('seguranca_museu'));
tipologia_museu	407.660	44 MB	SELECT pg_size_pretty (pg_relation_size('tipologia_museu'));
tipologiadoacervo	10	8192 bytes	SELECT pg_size_pretty (pg_relation_size('tipologiadoacervo'));

Figura 2: Quantidade de registros e tamanho de cada tabela

O gerador de dados contém 13 funções responsáveis por gerar os dados: as 10 primeiras geram dados para cada campo da tabela `museu` (100.000 tuplas), que, neste caso, é a tabela principal do esquema e as outras 3 são responsáveis por relacionar os dados da tabela `museu` com as tabelas `acessibilidade`, `planodeseguranca` e `tipologiadoacervo`, cujos resultados são inseridos, respectivamente, nas tabelas `acessibilidade_museu`, `seguranca_museu` e `tipologia_museu`. Como

os relacionamentos possuem cardinalidade `n:m`, a quantidade de tuplas geradas nessas tabelas pode ser maior do que a quantidade existente na tabela `museu`, assim como podemos observar na `Figura 2`.

6 Técnicas de Acesso Eficiente ao Banco de Dados

6.1 Especificação da versão

Para a realização dos testes em ambas as consultas foi utilizado um computador com a seguinte versão do PostgreSQL instalada:

> PostgreSQL 9.6.12 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.4.7 20120313 (Red Hat 4.4.7-16), 64-bit

6.2 Consulta 1

6.2.1 Execução

Para otimizar a consulta, foram construídas duas MATERIALIZED VIEW para diminuir o tempo que a consulta não otimizada levava para executar as querys em tempo de execução. Dessa forma, o NATURAL JOIN que é executado na consulta já faz a junção com essas tabelas instânciadas. Neste caso, a criação de Views com armazenamento físico é benéfico, pois o minimundo da aplicação não envolve atualizações frequentes. Como a cláusula MATERIALIZED VIEW gera o armazenamento do resultado da consulta e o atualiza no caso de alguma mudança, ela pode não ser vantajosa nos casos em que ocorram muitas atualizações nas tabelas do banco da dados.

```
CREATE MATERIALIZED VIEW MV_qtdseguranca AS (

    SELECT nome_museu, count(*) AS qntidade_seguranca FROM
    seguranca_museu GROUP BY nome_museu;

CREATE MATERIALIZED VIEW MV_qtdacess AS (

    SELECT nome_museu, count(*) AS qntidade_acessibilidade
    FROM acessibilidade_museu GROUP BY nome_museu;
```

Também foram criados os índices conforme mostrado abaixo, com objetivo de melhorar os tempos de condição para junção de tabelas.

```
CREATE INDEX idx_acc_museu_by_nome_museu ON (

    acessibilidade_museu(nome_museu)

CREATE INDEX idx_seg_museu_by_nome_museu ON (

    seguranca_museu(nome_museu)
```

Abaixo encontra-se a tabela comparativa contendo 5 amostragens de tempo de execução da primeira consulta e o percentual da diferença entre a consulta inicial e consulta otimizada.

	Consulta Original (1)	Com Index (2)	Diferença (%) Entre 1 e 2
Amostra 1	7697.759 ms	1232.146 ms	0,839
Amostra 2	6752.442 ms	1078.298 ms	0,840
Amostra 3	6724.002 ms	1018.758 ms	0,848
Amostra 4	6882.653 ms	1016.343 ms	0,852
Amostra 5	6732.763 ms	986.325 ms	0,853
Médias	6957,923 ms	1066,374 ms	0,846

Figura 3: Tabela Comparativa - Consulta 1

6.2.2 Plano de Consulta

Para testar a primeira consulta, vamos supor que a tipologia selecionada pelo usuário é "História". Por meio da cláusula `EXPLAIN ANALYSE`, é possível ter acesso à descrição do plano da consulta.

```
22  -> Hash Join  (cost=1374.79..2957.45 rows=1 width=325) (actual time=3511.746..3511.746 rows=0 loops=1)
23      Hash Cond: ((acess.nome_museu)::text = (museu.nome_museu)::text)
24      -> CTE Scan on acess  (cost=0.00..1332.76 rows=66638 width=226) (actual time=2459.975..2459.975 rows=1 loops=1)
25      -> Hash  (cost=1374.78..1374.78 rows=1 width=753) (actual time=1051.754..1051.754 rows=0 loops=1)
26          Buckets: 1024  Batches: 1  Memory Usage: 8kB
27          -> Hash Join  (cost=8.48..1374.78 rows=1 width=753) (actual time=1051.754..1051.754 rows=0 loops=1)
28              Hash Cond: ((segur.nome_museu)::text = (museu.nome_museu)::text)
29              -> CTE Scan on segur  (cost=0.00..1150.56 rows=57528 width=226) (actual time=1021.358..1021.358 rows=1 loops=1)
30              -> Hash  (cost=8.46..8.46 rows=1 width=527) (actual time=30.379..30.380 rows=0 loops=1)
31                  Buckets: 1024  Batches: 1  Memory Usage: 8kB
32                  -> Nested Loop  (cost=0.42..8.46 rows=1 width=527) (actual time=30.377..30.377 rows=0 loops=1)
33                      -> CTE Scan on tipo  (cost=0.00..0.02 rows=1 width=436) (actual time=30.376..30.376 rows=0 loops=1)
34                      -> Index Scan using museu_pkey on museu  (cost=0.42..8.44 rows=1 width=91) (never executed)
35                          Index Cond: ((nome_museu)::text = (tipo.nome_museu)::text)
36  Planning time: 29.901 ms
37  Execution time: 3520.274 ms
```

Figura 4: Consulta 1 - Plano de Consulta

A partir deste recurso, podemos perceber que é realizada uma leitura sequencial na tabela `acessibilidade_museu` durante a formação da primeira Common Table Expressions (CTE), a qual é utilizada posteriormente nas junções com `tipo` e `segur`, as quais, por sua vez, também são CTEs. O mesmo ocorre com relação à tabela `segurança_museu`. Neste caso, os índices secundários são úteis para diminuir o impacto das buscas sequenciais. Portanto, foram criados dois índices: `idx_seg_museu_by_nome_museu` e `idx_acc_museu_by_nome_museu`.

Após a criação dos índices secundários importantes para a otimização desta consulta, foi possível obter o plano de consulta apresentado na Figura 5.

Logo, podemos concluir que a melhora no desempenho ocorreu devido à não necessidade da busca sequencial nas tabelas mencionadas, sendo realizada apenas o carregamento das páginas selecionadas pelos índices secunários criados.

```
1  Sort  (cost=77694.90..77694.91 rows=1 width=325) (actual time=0.187..0.187 rows=0 loops=1)
2    Sort Key: acess.qntidade_acessibilidade, segur.qntidade_seguranca DESC
3    Sort Method: quicksort  Memory: 25kB
4    CTE acess
5      ->  GroupAggregate  (cost=0.55..52013.94 rows=66638 width=72) (actual time=0.071..0.071 rows=1 loops=1)
6            Group Key: acessibilidade_museu.nome_museu
7            ->  Index Only Scan using idx_acc_museu_by_nome_museu on acessibilidade_museu  (cost=0.55..49301.81 rows=409151 width=6..
8                  Heap Fetches: 7
9    CTE segur
10     ->  GroupAggregate  (cost=0.42..22715.14 rows=57528 width=72) (actual time=0.041..0.041 rows=1 loops=1)
11           Group Key: seguranca_museu.nome_museu
12           ->  Index Only Scan using idx_seg_museu_by_nome_museu on seguranca_museu  (cost=0.42..21301.95 rows=167582 width=64) (...
13                 Heap Fetches: 4
14   CTE tipo
15     ->  Index Scan using idx_tip_museu_by_nome_tipologia on tipologia_museu  (cost=0.42..8.36 rows=1 width=79) (actual time=0.040..0...
16           Index Cond: ((nome_tipologia)::text = 'Historia'::text)
17   ->  Hash Join  (cost=1374.79..2957.45 rows=1 width=325) (actual time=0.167..0.167 rows=0 loops=1)
```

Figura 5: Consulta 1 - Execução após a criação dos índices

6.3 Consulta 2

6.3.1 Execução

Abaixo encontra-se a tabela comparativa contendo 5 amostragens de tempo de execução da segunda consulta e o percentual da diferença entre a consulta inicial e consulta otimizada. Nas amostras abaixo é possível verificar as mudanças que ocorreram nos tempos de execução de ambos os casos.

Foi adicionado o índice abaixo para melhorar a busca na tabela `tipologia_museu`

```
CREATE INDEX idx_tip_museu_by_nome_tipologia ON tipologia_museu
(nome_tipologia);
```

As MATERIALIZED VIEWs apresentadas e comentadas na seção anterior também foram utilizadas nesta consulta, pois, ela também realiza junções naturais entre ambas as tabelas geradas por elas. Assim como os índices sobre as tabelas `acessibilidade_museu`, `seguranca_museu` e `tipologia_museu`.

	Consulta Original (1)	Com Index (2)	Diferença (%) Entre 1 e 2
Amostra 1	6604.225 ms	75.965 ms	0,988
Amostra 2	5366.651 ms	70.853 ms	0,986
Amostra 3	6209.702 ms	76.033 ms	0,987
Amostra 4	6179.703 ms	68.617 ms	0,988
Amostra 5	6422.447 ms	68.928 ms	0,989
Médias	6156,5456 ms	72,0792 ms	0,988

Figura 6: Consulta 2 - Tabela Comparativa

6.3.2 Plano de Consulta

Por meio da cláusula `EXPLAIN ANALYSE`, é possível ter acesso à descrição do plano da consulta. Primeiramente, para definir as CTEs segur e acess, os mesmos passos são utilizados: leitura sequencial nas tabelas acessibilidade_museu (409.151 linhas) e suguranca_museu (167.782 linhas) como é possível observar na `Figura 7`.

	QUERY PLAN text
1	Hash Join (cost=100323.96..101908.10 rows=149 width=106) (actual time=4187.727..6105.005 rows=392 loops=1)
2	Hash Cond: ((acess.nome_museu)::text = (museu.nome_museu)::text)
3	CTE acess
4	-> GroupAggregate (cost=63831.03..67566.05 rows=66638 width=72) (actual time=2403.899..4206.455 rows=81902 loops=1)
5	Group Key: acessibilidade_museu.nome_museu
6	-> Sort (cost=63831.03..64853.91 rows=409151 width=64) (actual time=2403.863..4067.427 rows=409151 loops=1)
7	Sort Key: acessibilidade_museu.nome_museu
8	Sort Method: external merge Disk: 29816kB
9	-> Seq Scan on acessibilidade_museu (cost=0.00..10307.51 rows=409151 width=64) (actual time=13.394..241.923 rows=409151 l...
10	CTE segur
11	-> GroupAggregate (cost=25346.84..27178.98 rows=57528 width=72) (actual time=1125.661..1694.343 rows=66773 loops=1)
12	Group Key: seguranca_museu.nome_museu
13	-> Sort (cost=25346.84..25765.79 rows=167582 width=64) (actual time=1125.637..1632.078 rows=167582 loops=1)
14	Sort Key: seguranca_museu.nome_museu
15	Sort Method: external merge Disk: 12184kB
16	-> Seq Scan on seguranca_museu (cost=0.00..4501.82 rows=167582 width=64) (actual time=9.326..172.011 rows=167582 loops=1)

Figura 7: Plano de consulta inicial (sem otimização)

A partir da análise da consulta por meio das cláusulas EXPLAIN ANALYSE, é possível verificar que, nos níveis mais baixos do plano de execução, toda a tabela museu é percorrida sequencialmente para a seleção do estado requisitado, o que resulta em 3589 linhas. Em seguida, é realizada uma leitura na CTE segur para selecionar apenas as tuplas que possuem a quantidade de segurança maior que o valor estipulado na consulta.

```
17  -> CTE Scan on acess  (cost=0.00..1499.36 rows=22213 width=226) (actual time=2403.911..4292.688 rows=63604 loops=1)
18      Filter: (qntidade_acessibilidade > 2)
19      Rows Removed by Filter: 18298
20  -> Hash  (cost=5570.57..5570.57 rows=669 width=316) (actual time=1783.437..1783.437 rows=598 loops=1)
21      Buckets: 1024  Batches: 1  Memory Usage: 125kB
22      -> Hash Join  (cost=4197.59..5570.57 rows=669 width=316) (actual time=1151.737..1782.567 rows=598 loops=1)
23          Hash Cond: ((segur.nome_museu)::text = (museu.nome_museu)::text)
24          -> CTE Scan on segur  (cost=0.00..1294.38 rows=19176 width=226) (actual time=1125.770..1746.920 rows=16855 loops=1)
25              Filter: (qntidade_seguranca > 3)
26              Rows Removed by Filter: 49918
27          -> Hash  (cost=4154.00..4154.00 rows=3487 width=90) (actual time=25.920..25.920 rows=3589 loops=1)
28              Buckets: 4096  Batches: 1  Memory Usage: 466kB
29              -> Seq Scan on museu  (cost=0.00..4154.00 rows=3487 width=90) (actual time=0.032..24.281 rows=3589 loops=1)
30                  Filter: ((estado)::text = 'MT'::text)
31                  Rows Removed by Filter: 96411
```

Figura 8: Continuação do Plano de consulta inicial (sem otimização)

Após a inclusão dos índices, houve uma redução significativa do tempo de execução, como destacado na seção anterior. A `Figura 9` destaca a parte do plano de consulta em que há a utilização dos índices criados. Nesta imagem, podemos verificar que os índices estão sendo utilizados durante a execução da consulta.

```
1  Hash Join  (cost=67140.68..68508.08 rows=111 width=106) (actual time=366.589..635.579 rows=227 loops=1)
2    Hash Cond: ((segur.nome_museu)::text = (museu.nome_museu)::text)
3    CTE acess
4      ->  Limit  (cost=0.55..39027.38 rows=50000 width=72) (actual time=0.012..210.078 rows=50000 loops=1)
5            ->  GroupAggregate  (cost=0.55..52013.94 rows=66638 width=72) (actual time=0.012..204.622 rows=50000 loops=1)
6                  Group Key: acessibilidade_museu.nome_museu
7                  ->  Index Only Scan using idx_acc_museu_by_nome_museu on acessibilidade_museu  (cost=0.55..49301.81 rows=409151 width=64) (actual ti...
8                        Heap Fetches: 249586
9    CTE segur
10     ->  GroupAggregate  (cost=0.42..22715.14 rows=57528 width=72) (actual time=0.018..210.122 rows=66773 loops=1)
11           Group Key: seguranca_museu.nome_museu
12           ->  Index Only Scan using idx_seg_museu_by_nome_museu on seguranca_museu  (cost=0.42..21301.95 rows=167582 width=64) (actual time=0....
13                 Heap Fetches: 167582
14   ->  CTE Scan on segur  (cost=0.00..1294.38 rows=19176 width=226) (actual time=0.045..260.783 rows=16855 loops=1)
15         Filter: (qntidade_seguranca > 3)
16         Rows Removed by Filter: 49918
```

Figura 9: Consulta 2 - Execução após a criação dos índices secundários

Essa alteração no desempenho ocorre devido ao fato de que não há mais necessidade de percorrer todas as tuplas, pois apenas as páginas realmente relevantes são carregadas para a leitura e execução.

7 Programação com Banco de Dados

Nas subseções que se seguem abaixo, serão descritas as funções que implementam cada consulta. Neste caso, ambas as funções retornam tabelas, pois executam querys que geram a visualização do resultado das consultas.

7.1 Consulta 1

Assim como a primeira consulta definida na seção 4.1, esta *Store Procedure* retorna as quantidades de serviços de acessibilidade e de segurança a partir de uma tipologia recebida como parâmetro. O parâmetro é recebido como uma expressão regular que constitui a busca relativa da aplicação. Deste modo, o parâmetro possui o tipo varchar. Foi adicionado a cláusula SECURITY DEFINER que será abordada na próxima seção. A query de criação é mostrada a seguir:

```
CREATE OR REPLACE FUNCTION consulta_um(p_tipologia VARCHAR(15))
    RETURNS table( nome_museu VARCHAR(100), nome_tipologia VARCHAR(100), esfera VARCHAR(100), anot_criacao integer, qntidade_acessibilidade bigint , qntidade_seguranca bigint ) as $$
        BEGIN
        RETURN QUERY
        CONSULTA 1
        END;
$$ LANGUAGE plpgsql SECURITY DEFINER;
```

7.2 Consulta 2

Com relação à segunda *Store Procedure*, uma expressão exata é recebida como parâmetro, a sigla do estado no qual o museu está localizado. Mas, nesse caso, a busca pelo estado é uma pesquisa. Foi adicionado a cláusula SECURITY DEFINER que será abordada na próxima seção. A query de criação é mostrada a seguir:

```
CREATE OR REPLACE FUNCTION consulta_dois(p_estado VARCHAR(15), p_acc int, p_seg int)
    RETURNS table( nome_museu VARCHAR(100), estado VARCHAR(100), esfera VARCHAR(100), qntidade_acessibilidade bigint , qntidade_seguranca bigint ) as $$
```

```
        BEGIN
        RETURN QUERY
        CONSULTA 2
        END;
$$ LANGUAGE plpgsql SECURITY DEFINER;
```

8 Controle de acesso de usuários

É perceptível que o foco deste banco de dados, cujo minimundo foi descrito na seção 2, não possui um número significativo de atualizações. E, caso elas aconteçam, não serão realizadas diretamente por um usuário, mas sim por um administrador responsável pelo gerenciamento das informações sobre os museus a serem disponibilizadas. O usuário comum, neste caso, é autorizado a essencialmente a visualizar as informações sobre os museus.

Neste contexto foram criados dois tipo de papel de usuários: diretor e visitante, foi estabelecido quais consultas cada tipo de usuário tem acesso pela cláusula GRANT EXECUTE e por fim criado um usuário para cada papel. Em nosso projeto, o papel diretor tem acesso as duas consultas e o papel visitante acesso apenas a consulta 2. Foi adicionado a cláusula SECURITY DEFINER na produção da *Store Procedure* que pode ser visto na seção anterior para poder definir privilégios desses usuários. As querys de todas essas definições estão abaixo:

```
CREATE ROLE diretor;
CREATE ROLE visitante;

GRANT EXECUTE ON FUNCTION consulta_um(VARCHAR)
TO diretor;
GRANT EXECUTE ON FUNCTION consulta_dois(VARCHAR, int, int)
TO diretor;
GRANT EXECUTE ON FUNCTION consulta_dois(VARCHAR, int, int)
TO visitante;

CREATE ROLE dir LOGIN PASSWORD '111' IN ROLE diretor;
CREATE ROLE vis LOGIN PASSWORD '111' IN ROLE visitante;
```

Na Figura 10, é possível o analisar a quais tabelas e fuções cada tipo de usuário tem acesso.

	Diretor	Visitante
Consulta 1	SELECT	Sem acesso
Consulta 2	SELECT	SELECT

Figura 10:

9 Outras Informações

Nas subseções que seguem abaixo, serão descritas os detalhes de implementação do bando de dados, assim como a aplicação produzida para testar o resultado das consultas, exibição dos dados e outros detalhes específicos do projeto.

9.1 Detalhes de implementação

Para produção do bando de dados, utilizamos o SGBD PostgreSQL justamente com pgAdmin que é uma interface para trabalhar com administração e desenvolvimento de banco de dados.

9.2 Interface da aplicação

Para exibir os dados de forma mais amigável e visível, foi produzido uma aplicação Java que realiza a conexão com o servidor local do Postgre onde se encontra o banco produzido neste projeto. Para isso foi feita configuração de um driver JDBC que é disponibilizado pela organização que cuida do Postgre.

Foi utilizado como base uma aplicação já existente mostrada na aula de Laboratórios de Banco de Dados e feitas todas as mudanças necessárias para que possa funcionar com nosso projeto, assim como adaptar a interface para o resultado ao qual irá exibir. Na Figura 11 é possível ver a interface visual da aplicação. É uma interface simples que conta com duas opções, a primera é para escolher qual usuário deseja realizar a consulta e a segunda é qual consulta deseja realizar com tal usuário.

O contexto dos campos de busca é modificado de acordo com a consulta escolhida. A aplicação leva em consideração as configurações feitas no SGBD de controle de acesso de usuários como detalhado na seção 8.

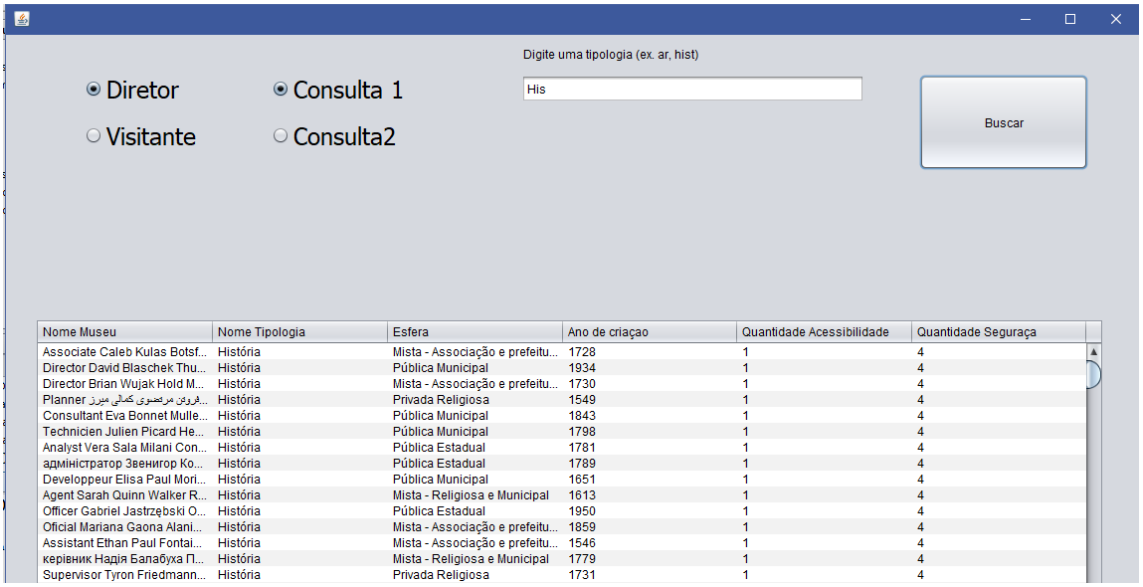
Digite uma tipologia (ex. ar, hist)
Diretor
Visitante
Consulta 1
Consulta2
His
Buscar
Nome Museu
Nome Tipologia
Esfera
Ano de criaçao
Quantidade Acessibilidade
Quantidade Seguraça
Associate Caleb Kulas Botsf... História Mista - Associação e prefeitu... 1728 1 4
Director David Blaschek Thu... História Pública Municipal 1934 1 4
Director Brian Wujak Hold M... História Mista - Associação e prefeitu... 1730 1 4
Planner فروتن مرتضوی کمالی میرز... História Privada Religiosa 1549 1 4
Consultant Eva Bonnet Mulle... História Pública Municipal 1843 1 4
Technicien Julien Picard He... História Pública Municipal 1798 1 4
Analyst Vera Sala Milani Con... História Pública Estadual 1781 1 4
адміністратор Звенигор Ко... História Pública Estadual 1789 1 4
Developpeur Elisa Paul Mori... História Pública Municipal 1651 1 4
Agent Sarah Quinn Walker R... História Mista - Religiosa e Municipal 1613 1 4
Officer Gabriel Jastrzębski O... História Pública Estadual 1950 1 4
Oficial Mariana Gaona Alani... História Mista - Associação e prefeitu... 1859 1 4
Assistant Ethan Paul Fontai... História Mista - Associação e prefeitu... 1546 1 4
керівник Надія Балабуха П... História Mista - Religiosa e Municipal 1779 1 4
Supervisor Tyron Friedmann... História Privada Religiosa 1731 1 4

Figura 11:

10 Considerações Finais

O projeto detalhado ao longo deste documento é o resultado final das várias iterações realizadas durante um semestre da disciplina de Laboratórios de Banco de dados e que se iniciou a partir apenas de um tema: Museus Brasileiros. Durante o processo, pudemos ter acesso as diversas abordagens para se produzir um bando de dados, como produzir consultas que trazem os dados que queremos, como otimiza-la para que retorne os dados rápidamente, como controlar que tipo de usuários tem acesso a tabelas e também como exibir esses dados em uma interface visual.

As primeiras fases de elaboração conceitual do banco, das consultas e a produção dos dados sintéticos, não tivemos dificuldades já que é um conhecimento que adquirimos na disciplina de Banco de Dados. Para otimização das consultas, foi necessário mais estudos e testes, pois para conseguir tempos menores tivemos que realizar algumas modificações nas querys e também produzir alguns "agilizadores"em forma de MATERIALIZED VIEWS e INDEX para chegar a um resultado de tempo melhor que as primeiras versões das consultas. Inicialmente, a otimizações feitas por nós nesta fase do projeto não estavam de acordo com as exigências e também podiam retornar resultados erroneos por se basear unicamente em limitar a quantidade de dados retornados. Toda a seção 6 foi alterada e o resultado final das otimizações é a que se encontra neste documento.

Em relação a programação do banco, não houve problemas assim como o controle de acesso de usuários, onde bastou elaborar conceitualmente quais usuários existiriam no contexto do projeto e produzir as querys que realizam essa configuração no Postgre.

Quanto a aplicação e interface para exibir os dados, utilizamos como base uma já existente e que possuía a estrutura que necessitávamos em nosso projeto. Fizemos a modificação na conexão com nosso banco e usuários específicos, criamos as classes de acordo com nosso contexto e modificamos a interface para se adequar a nossas consulta e exibição de nossos dados.

Ao chegar neste resultado final, reconhecemos que muito ainda poderia ter sido feito mas acreditamos que utilizamos todo o conhecimento adquirido durante o semestre e conseguimos aplicar com sucesso neste projeto.

Bibliografia

Guia dos Museus Brasileiros
Link: https://www.museus.gov.br/wp-content/uploads/2011/05/gmb_sudeste.pdf
Museus em números - Dados institucionais
http://www.museus.gov.br/wp-content/uploads/2011/11/museus_em_numeros_volume1.pdf