**Universidade Federal de São Carlos – Campus Sorocaba**

**Bacharelado em Ciência da Computação**

**Sistemas de Bancos de Dados**

Profa. Sahudy Montenegro

**Projeto Integrado – Bases de dados brasileiras**

Tema 6 – Cadastro brasileiro de cursos universitários

Grupo 4:

José Vitor Novaes dos Santos RA: 743556

Marcus Vinícius Natrielli Garcia RA: 743578

Victor Fernandes de Oliveira Brayner RA: 743600

William Giacometti Dutra de Oliveira RA: 743606

**Fase Final: 12/06/2019**

Sumário

[1. Descrição do minimundo 3](#_Toc11005212)

[2. Definição de esquema de BD 3](#_Toc11005213)

[3. Especificação do banco e das consultas em SQL 4](#_Toc11005214)

[3.1 Código SQL do banco de dados: 4](#_Toc11005215)

[3.2 Código SQL das consultas (sem otimização): 5](#_Toc11005216)

[4. Amostragem das consultas sem otimização 7](#_Toc11005217)

[5. Populando o Banco de Dados 7](#_Toc11005218)

[6. Indexação e otimização das consultas 8](#_Toc11005219)

[7. Programação com Banco de Dados 12](#_Toc11005220)

[8. Controle de acesso de usuários 12](#_Toc11005221)

[9. Informações sobre o projeto 14](#_Toc11005222)

[10. Considerações finais 14](#_Toc11005223)

1. Descrição do minimundo

São muitos os cursos universitários existentes no Brasil, e, a partir disso, torna-se necessária a existência de um sistema para armazenar e organizar eles e suas informações. As Instituições de ensino contam com uma sigla identificadora única, seu nome completo e a sua natureza administrativa (pública ou privada) e sua nota no índice Geral de Cursos (IGC, mas informalmente chamado de “Nota do Mec”) pelo Ministério da Educação como atributos.

Os campi são identificados pela sigla de sua instituição e pelo seu nome, e possuem como informações o estado e a cidade em que se localizam. Por fim, os cursos possuem nome, área do conhecimento (ciências exatas e da terra, ciências biológicas, engenharias, ciências da saúde, ciências agrárias, ciências sociais aplicadas, ciências humanas, linguística, letras e artes, e outros), nota no ENADE, grau do curso (bacharelado, licenciatura, mestrado, doutorado ou tecnólogo), o turno (matutino, noturno, integral ou vespertino), sua duração em semestres, o ano em que foi criado.

As notas IGC e ENADE variam de 0 a 5. Um curso é unicamente identificado a partir da sigla da instituição e do nome do campus ao qual pertence e do seu nome, de maneira que o campus de uma instituição só pode possuir um curso com um determinado nome. O ano de criação do curso deve ser maior que 1980, e não pode ultrapassar o ano atual.

Uma instituição pode ter vários campi, porém um campus só pertence a uma instituição. Além disso, um campus pode ter vários cursos, e de um campus para o outro, um certo curso pode ter informações diferentes.

1. Definição de esquema de BD

Modelo relacional do esquema do banco de dados proposto, baseado no código SQL presente no apêndice:

instituicao(sigla, nome\_instituicao, natureza\_administrativa, nota\_igc)

campus(nome\_campus, estado, cidade, sigla)

sigla referencia instituição (sigla)

curso(nome\_curso, area, nota\_enade, grau, turno, duracao, ano\_criacao, campus, sigla)

campus referencia campus(nome\_campus)

sigla referencia campus (sigla)

1. Especificação do banco e das consultas em SQL

Nesta seção é especificado o código SQL do banco de dados e das consultas, estas que também contam com seus respectivos enunciados, campos de busca, operadores de comparação e campos de visualização dos resultados.

* 1. Código SQL do banco de dados:

CREATE TABLE instituicao (

sigla varchar(10),

nome\_instituicao varchar(55) NOT NULL,

natureza\_administrativa varchar(7) NOT NULL CHECK (natureza\_administrativa IN ('Pública', 'Privada')),

nota\_IGC smallint NOT NULL CHECK (nota\_IGC >= 0 AND nota\_IGC <= 5),

PRIMARY KEY(sigla)

);

CREATE TABLE campus(

nome\_campus varchar(55) NOT NULL,

estado char(2) NOT NULL CHECK (estado IN ('AC', 'AL', 'AP', 'AM', 'BA', 'CE', 'DF', 'ES', 'GO', 'MA', 'MT', 'MS', 'MG', 'PA', 'PB', 'PR', 'PE', 'PI', 'RJ', 'RN', 'RS', 'RO', 'RR', 'SC', 'SP', 'SE', 'TO')),

cidade varchar(55) NOT NULL,

sigla varchar(10),

PRIMARY KEY(sigla, nome\_campus),

FOREIGN KEY(sigla) REFERENCES instituicao(sigla) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE curso(

nome\_curso varchar(55) NOT NULL,

area varchar(30) NOT NULL CHECK (area IN ('Ciências Exatas e da Terra', 'Ciências Biológicas', 'Engenharias', 'Ciências da Saúde', 'Ciências Agrárias', 'Ciências Sociais Aplicadas', 'Ciências Humanas', 'Linguística, Letras e Artes', 'Outros')),

nota\_enade smallint NOT NULL CHECK (nota\_enade >= 0 AND nota\_enade <= 5),

grau varchar(12) NOT NULL CHECK (grau IN ('Bacharelado', 'Licenciatura', 'Mestrado', 'Doutorado', 'Tecnólogo')),

turno varchar(10) NOT NULL CHECK (turno IN ('Matutino', 'Integral', 'Noturno', 'Vespertino')),

duracao smallint NOT NULL CHECK (duracao > 0 AND duracao <= 12),

ano\_criacao smallint NOT NULL CHECK (ano\_criacao <= DATE\_PART('YEAR', CURRENT\_DATE) AND ano\_criacao > 1980),

campus varchar(55) NOT NULL,

sigla varchar(10) NOT NULL,

PRIMARY KEY(sigla, campus, nome\_curso),

FOREIGN KEY(sigla, campus) REFERENCES campus(sigla, nome\_campus) ON DELETE CASCADE ON UPDATE CASCADE

);

## Código SQL das consultas (sem otimização):

1ª Consulta: Listar sigla da instituição, nome do campus, estado, cidade, nome, grau de especialização, turno e duração em semestres e nota no ENADE de cursos em instituições de uma certa natureza administrativa e que possuam um determinado termo no nome.

Campos de visualização do resultado: curso.sigla, campus, estado, cidade nome\_curso, grau, turno, duracao e nota\_enade.

Campos de busca (ou das condições): nome\_curso (relativa) e natureza\_administrativa (absoluta).

Operadores de condições: nome\_curso (LIKE), natureza\_administrativa (=).

SQL:

SELECT curso.sigla, campus, estado, cidade, nome\_curso, grau, turno, CONCAT(duracao, ' semestre(s)') AS duracao, nota\_enade

FROM instituicao, campus, curso

WHERE instituicao.sigla = campus.sigla AND campus.nome\_campus = curso.campus AND campus.sigla = curso.sigla AND nome\_curso LIKE '%<termo\_fornecido>%' AND natureza\_administrativa = '<natureza\_fornecida>'

ORDER BY curso.sigla, estado, cidade;

2ª Consulta: Ranquear campi que criaram mais cursos entre dois determinados anos, fornecendo a sigla, o nome e a natureza administrativa da instituição atrelada a eles, seus nomes, estados, cidades e quantidade de cursos.

Campos de visualização do resultado: campus.sigla, nome\_instituicao, natureza\_administrativa, nome\_campus, estado, cidade e quantidade\_de\_cursos.

Campos de busca (ou das condições): ano\_criacao (absoluta).

Operadores de condições: ano\_criacao (<=, >=).

SQL:

SELECT campus.sigla, nome\_instituicao, natureza\_administrativa, a.nome\_campus, estado, cidade, cont

FROM

(SELECT campus.sigla, campus.nome\_campus, COUNT(\*) AS cont

FROM campus, curso

WHERE campus.sigla = curso.sigla AND campus.nome\_campus = curso.campus AND ano\_criacao <= <ano\_fornecido1> AND ano\_criacao >= <ano\_fornecido2>

GROUP BY campus.sigla, campus.nome\_campus) AS a,

instituicao, campus

WHERE instituicao.sigla = campus.sigla AND a.sigla = campus.sigla AND a.nome\_campus = campus.nome\_campus

ORDER BY cont DESC, sigla;

# Amostragem das consultas sem otimização

Versão do PostgreSQL, utilizando o comando select version(): "PostgreSQL 9.6.13, compiled by Visual C++ build 1800, 64-bit".

Dadas as duas consultas propostas, seguem abaixo as tabelas 1 e 2 com a execução delas sem utilizar técnicas de indexação e otimização.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1ª exec. | 2ª exec. | 3ª exec. | 4ª exec. | 5ª execução |
| Tempo utilizado | 6.2 segs. | 5.8 segs. | 5.7 segs. | 5.7 segs. | 6.1 segundos |

*Tabela 1. Apresentação de resultados de execução da primeira consulta*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1ª exec. | 2ª exec. | 3ª exec. | 4ª exec. | 5ª execução |
| Tempo utilizado | 10.7 segs. | 10.5 segs. | 10.6 segs. | 10.4 segs. | 10.5 segundos |

*Tabela 2. Apresentação de resultados de execução da segunda consulta*

Para a primeira consulta, na Tabela 1, foi buscado o termo “Curso” no nome dos cursos e a natureza administrativa escolhida foi “Pública”. Para a segunda consulta, na Tabela 2, o intervalo de tempo, em anos, utilizado foi de 1980 até 2019, abrangendo todo o período de criação de cursos universitários.

As médias de tempo de execução das consultas são:

1ª Consulta: 5.9 segundos

2ª Consulta: 10.54 segundos

# Populando o Banco de Dados

Foram enviados os arquivos de Backup do banco de dados, com os dados das tabelas propostas para o trabalho integrado, junto da tabela registro\_backup, que possui as informações solicitadas sobre quantidade de dados em cada tabela e seus tamanhos em bytes; e o script, de autoria própria do grupo, que gerou a grande maioria dos dados do banco.

Os dados com nomes reais e formatados para serem apresentados posteriormente na aplicação a ser desenvolvida foram criados sem uso de scripts, usando a inserção manual. Os dados genéricos foram gerados por um script desenvolvido na linguagem C++.

# Indexação e otimização das consultas

Para a otimização das consultas, o grupo utilizou duas técnicas principais: a indexação (criação de índices com árvore B) e a sintonização das consultas, reescrevendo as mesmas para evitar operações com muitos acessos a disco.

Para a primeira consulta, reescrevendo a mesma, obtivemos:

SELECT curso.sigla, estado, cidade, campus, nome\_curso, grau, turno, CONCAT(duracao, ' semestre(s)') AS duracao, nota\_enade

FROM instituicao JOIN campus ON (natureza\_administrativa = 'Pública' AND instituicao.sigla = campus.sigla) JOIN curso ON (nome\_curso LIKE 'C%' AND campus.nome\_campus = curso.campus AND campus.sigla = curso.sigla)

ORDER BY curso.sigla, estado, cidade;

A segunda consulta, após ser reescrita, ficou:

SELECT campus.sigla, nome\_instituicao, natureza\_administrativa, a.nome\_campus, estado, cidade, cont

FROM

(SELECT campus.sigla, campus.nome\_campus, COUNT(\*) AS cont

FROM campus JOIN curso ON (campus.sigla = curso.sigla AND campus.nome\_campus = curso.campus AND ano\_criacao <= 2019 AND ano\_criacao >= 1980)

GROUP BY campus.sigla, campus.nome\_campus) AS a JOIN

campus ON (a.sigla = campus.sigla AND a.nome\_campus = campus.nome\_campus) JOIN instituicao ON (instituicao.sigla = campus.sigla)

ORDER BY cont DESC, sigla;

Os índices criados para ambas as consultas foram:

CREATE EXTENSION pg\_trgm;

CREATE UNIQUE INDEX natureza\_insti ON instituicao (sigla, natureza\_administrativa);

CREATE INDEX gin\_nome\_curso ON curso USING gin(sigla gin\_trgm\_ops, campus gin\_trgm\_ops, nome\_curso gin\_trgm\_ops);

CREATE INDEX fk\_curso ON curso(sigla, campus);

CREATE INDEX fk\_campus ON campus(sigla);

CREATE INDEX ano ON curso (sigla, campus, ano\_criacao);

Sendo que pg\_trgm não é um índice e sim uma opção pra ativar uma estrutura de dados pro índice gin.

Apesar do tempo de execução das "Queries" ter sido significativamente diferente nas diferentes máquinas (a consulta 1, por exemplo, aumentou em 6 segundos em outra versão do PgAdmin), a criação de índices e a otimização baseada na sintonização das consultas não fez diferença no tempo de execução.

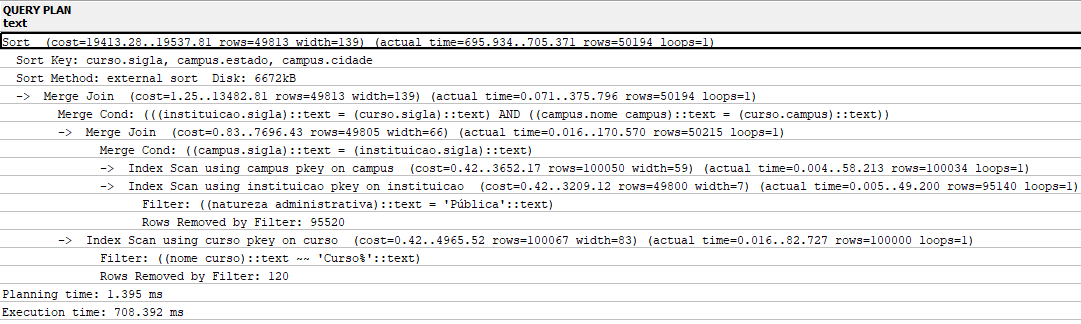
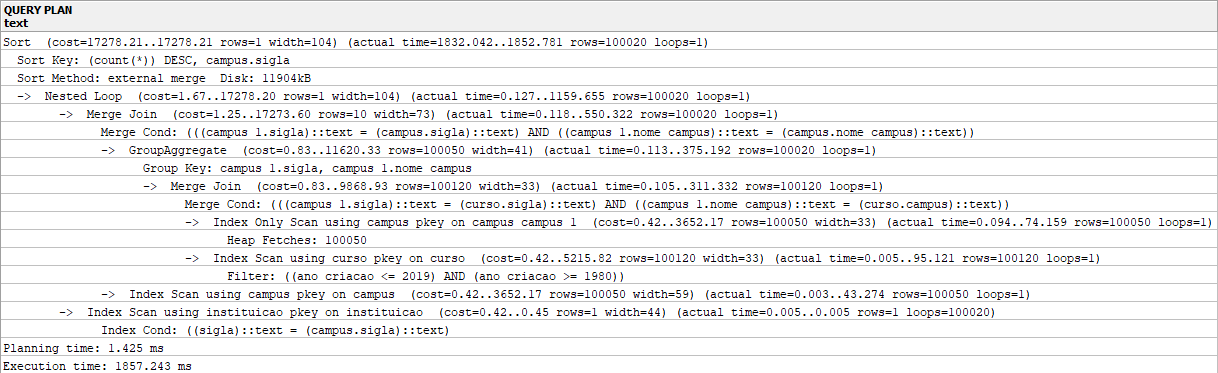


Figura 2. Explain analyse da segunda consulta sem índices

Figura 1. Explain analyse da primeira consulta sem índices

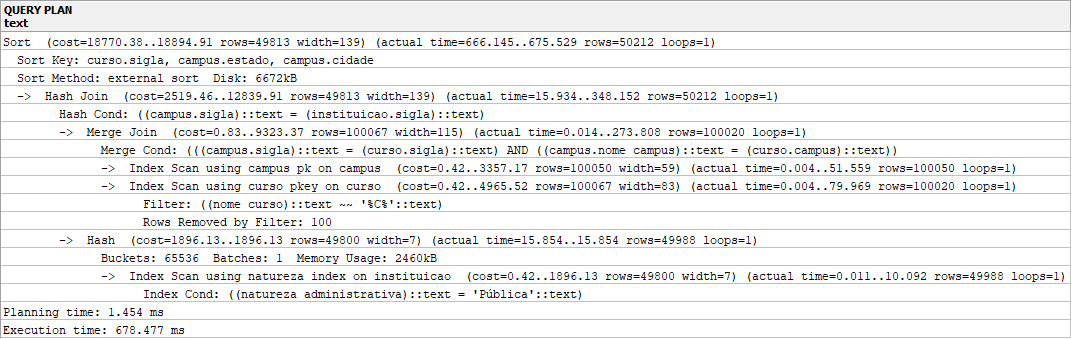
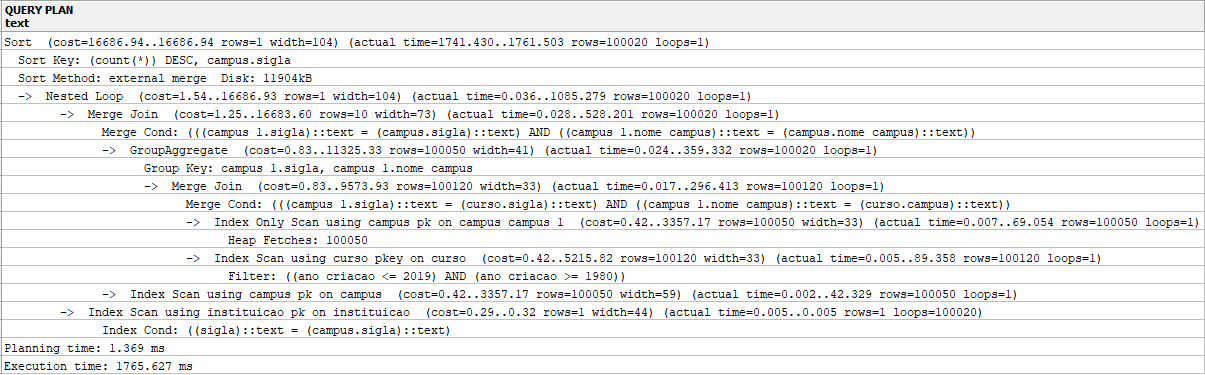
A execução das consultas otimizadas é expressa pelas tabelas 3 e 4:

Figura 4. Explain analyse da segunda consulta com índices

Figura 3. Explain analyse da primeira consulta com índices

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1ª exec. | 2ª exec. | 3ª exec. | 4ª exec. | 5ª execução |
| Tempo utilizado | 6 segs. | 6 segs. | 6 segs. | 6 segs. | 6 segundos |

*Tabela 3. Apresentação de resultados de execução da primeira consulta otimizada*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1ª exec. | 2ª exec. | 3ª exec. | 4ª exec. | 5ª execução |
| Tempo utilizado | 10.3 segs. | 10.3 segs. | 10.3 segs. | 10.4 segs. | 10.4 segundos |

*Tabela 4. Apresentação de resultados de execução da segunda consulta otimizada*

As médias de tempo de execução das consultas são:

1ª Consulta: 6 segundos

2ª Consulta: 10.34 segundos

As tabelas 5 e 6 comparam a média de execução entre as consultas:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Consulta inicial | Consulta otimizada | Diferença (%) |
| Tempo de execução | 6.1 segundos | 6.0 segundos | 1,6 |

*Tabela 5. Apresentação de resultados da primeira consulta*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Consulta inicial | Consulta otimizada | Diferença (%) |
| Tempo de execução | 10.54 segundos | 10.34 segundos | 1,9 |

*Tabela 6. Apresentação de resultados da segunda consulta*

Buscamos utilizar os métodos de indexação e de sintonização de consultas para otimizar a execução das consultas pois a criação de índices é a ferramenta mais conhecida na criação de banco de dados para aperfeiçoar o tempo de execução e busca de dados de uma consulta; enquanto a sintonização permite que uma mesma consulta seja executada de forma mais eficiente dependendo do contexto no qual ela é utilizada, utilizando operadores menos custosos e realizando menos acessos a disco.

A principal transformação realizada nas consultas durante o processo de sintonização foi a troca da cláusula WHERE pela operação JOIN, visando diminuir a junção de tuplas por produto cartesiano.

1. Programação com Banco de Dados
2. Controle de acesso de usuários

Essa seção descrever as diferentes visões de usuários presentes no banco de dados e suas relações com as tabelas, como mostrado abaixo na tabela 7.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Administrador | Gerente | Usuário |
| Campus | S, I, U, D | S, U | S |
| Curso | S, I, U, D | S, U | S |
| Instituição | S, I, U, D | S, U | S |

*Tabela 7. Perfis de usuários, seu papel e privilégios*

*(SELECT (S), INSERT (I), DELETE (D) e UPDATE (U))*

Script:  
--Cria um usuário com login e senha, que pode criar outros usuários e bancos de dados

CREATE USER administrador WITH NOINHERIT PASSWORD 'admin' CREATEDB CREATEROLE;

--Dá todos os privilégios nas 3 tabelas especificadas ao usuário administrador e permite que ele dê esses privilégios a usuários que ele criar

GRANT ALL PRIVILEGES ON campus, curso, instituicao TO administrador WITH GRANT OPTION;

--Da todos os privilégios do usuário postgres para o usuário administrador

GRANT postgres TO administrador;

--Seleciona o usuário administrador

SET ROLE administrador;

--Com o usuário administrador selecionado cria o usuário gerente com login e senha, que pode criar apenas outros usuários

CREATE USER gerente WITH NOINHERIT PASSWORD '12345678' CREATEROLE;

--Dá privilégios de leitura e atualização ao usuário gerente nas 3 tabelas especificadas e permite que ele dê esses privilégios a usuários que ele criar

GRANT SELECT, UPDATE ON campus, curso, instituicao TO gerente WITH GRANT OPTION;

--Seleciona o usuário gerente

SET ROLE gerente;

--Cria um papel chamado usuario

CREATE ROLE usuario WITH NOINHERIT;

--Garante privilégios de apenas leitura ao papel usuario

GRANT SELECT ON campus, curso, instituicao TO usuario;

A aplicação utiliza apenas o papel usuario para as consultas e para conectar com o banco de dados foi criada uma conexão com o JDBC em uma classe Java, seguindo o Padrão de projeto DAO. A partir daí, os métodos dessa classe representam as consultas que são utilizadas na aplicação.

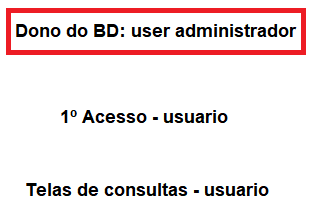


Figura 5. Conexão utilizando o perfil de usuário

1. Informações sobre o projeto
2. Considerações finais