Bacharelado Ciência da Computação

Trabalho de Banco de Dados

Grupo 17

Grupo 17

Vinícius Brandão Crepschi - 743601

Pedro Coelho - 743585

Natham Corracini - 743582

16/10/2018

Bancos de Dados

Profª Drª Sahudy Montenegro González

Sumário

1 Objetivos e especificação do problema

Um grande fã de música, deseja armazenar toda sua coleção de álbuns de maneira digital. Para cada álbum deseja-se armazenar: seu artista, título, ano de lançamento, gênero e ordem cronológica seguindo os álbuns já lançados por este artista (exemplo: este é o primeiro álbum de determinado artista), para este tipo-entidade, adotaremos também um ID referente ao álbum como chave primária. Um álbum deve estar relacionado a apenas um artista (este sistema não considera álbuns colaborativos). Além disso, um álbum deve ser composto por ao menos uma música.

Esse grande entusiasta também deseja armazenar as seguintes informações para cada música: título, duração, número da faixa no álbum, seu artista e um ID único referente a música que funcionará como chave primária. Cada música deve estar relacionada a um e somente um álbum, não há preocupação sobre o formato em que os álbuns foram gravados (CD, fita, disco de vinil, etc). Dois artistas distintos podem gravar uma mesma música, já que duas versões de uma mesma música terão diversos campos divergentes (exemplo: artista, álbum, número da faixa, etc).

Além do mais, deseja-se manter as seguintes informações sobre os artistas de seus respectivos álbuns e músicas: nome e país de origem, neste caso definimos seu nome como chave primária. Um artista pode se relacionar com diversos álbuns e músicas, porém uma música ou álbum só podem estar relacionados a um único artista.

Outra funcionalidade desejada por nosso entusiasta seria o armazenamento de Playlists, que seria uma espécie de “coletânea” de diversas músicas, sem que haja a necessidade destas fazerem parte de um mesmo álbum ou serem compostas por um mesmo artista. Além disso, uma mesma música pode fazer parte de diversas playlists e uma playlist deve possuir ao menos uma música.

O sistema deve permitir operações como: listar todas as informações de determinado álbum, listar todas as músicas de um certo álbum, listar o nome de artista dado uma música, consultar o número da faixa de uma música, listar todos os álbuns presentes no sistema, listar todas a músicas presentes no sistema, listar todas as músicas de determinado artista, listar todos os álbuns de um certo artista, listar títulos de álbuns dado um certo ano de lançamento, listar todos os álbuns dado um gênero musical e listar o número total de álbuns, mostrar a média do número de músicas por álbum e músicas presentes no sistema. Além de operações básicas como: inserção, remoção e alteração de registros.

2 Diagrama MER

Na figura 1, apresenta-se o diagrama do Modelo Entidade-Relacionamento referente ao minimundo especificado. Nele explicitam-se as relações entre os tipo-entidades, bem como seus atributos:

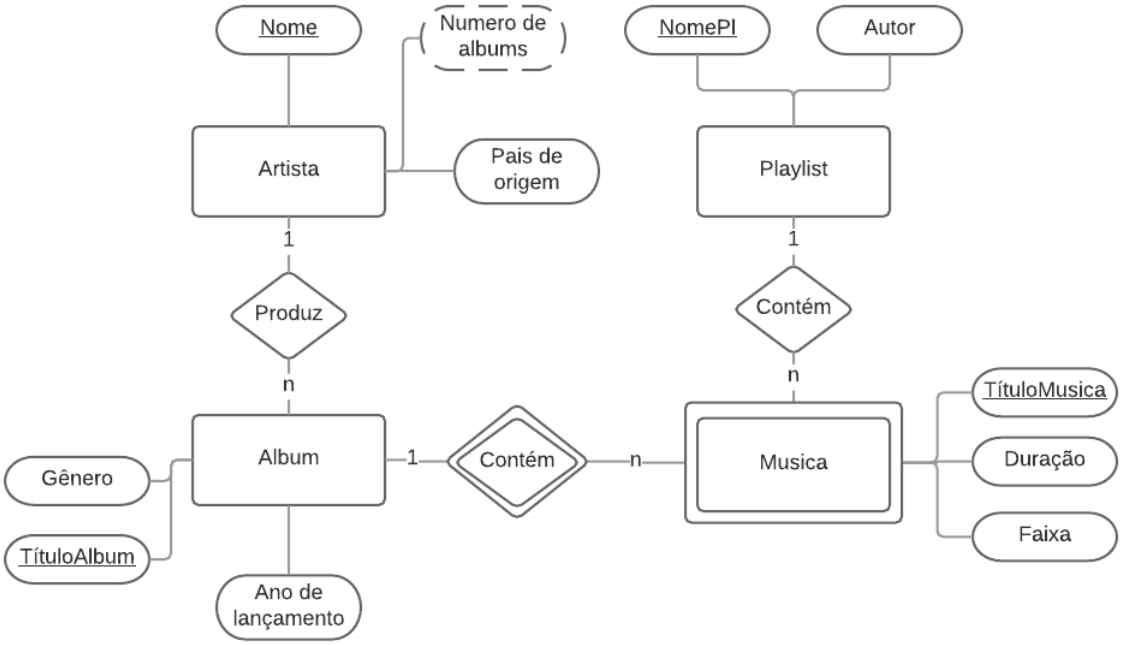

Figura 1 – Modelagem Entidade-Relacionamento do projeto

3 Tabela de Metadados

Na figura 2, apresenta-se a tabela de metadados referente ao projeto.

Tipo-Entidade	Atributo	Tipo	Restrição
Artista	Nome País de origem	Chave Monovalorado	Obrigatório Obrigatório
Álbum	Ano de lançamento Gênero TítuloÁlbum	Monovalorado Monovalorado Chave	Obrigatório, 1500 <= Ano Obrigatório Obrigatório
Música	TítuloMúsica Duração Faixa	Chave Monovalorado Monovalorado	Obrigatório Obrigatório, 10s <= Duração Obrigatório
Playlist	Autor NomePl	Monovalorado Chave	Obrigatório Obrigatório

Figura 2 – Metadados referentes ao projeto

4 Requisitos de Dados

1. Listar títulos de álbuns dado um certo gênero musical.
2. Mostrar a média do número de músicas por álbum.
3. Listar todas as músicas dado um certo álbum.
4. Listar nome do artista dado uma música.
5. Consultar o número da faixa de uma música.
6. Listar títulos de álbuns dado um certo ano de lançamento.
7. Listar todos os álbuns de um certo artista.
8. Listar todas as músicas de um certo artista.
9. Listar artistas com albuns lançados em determinado ano
10. Contagem de álbuns cadastrados.
11. Contagem de músicas cadastradas.
12. Listar todos os álbuns cadastrados.
13. Listar todas as músicas cadastradas.
14. Listar todas as músicas de uma determinada playlist.
15. Contagem de músicas por playlist.
16. Contagem de álbuns por artista (Campo calculado).

5 Modelo Relacional

Este modelo é uma representação do mini-mundo, de maneira mais próxima a forma como a implementação será realizada. Este modelo parte do princípio de que todos os dados a são armazenados em tabelas. O modelo abaixo foi obtido através da conversão do nosso diagrama MER (Figura 1) para um Modelo Relacional utilizando o BrModelo.

Artista(Nome, PaisDeOrigem)

Na tabela Artista, apenas incluimos seus atributos, não sendo necessário incluir as relações referentes a ele, pois isso incluiria campos multivalorados ao mapear o relacionamento entre Artista e Álbum através de um atributo inserido na tabela de artista, dado que um artista pode ter um ou mais álbuns (por isso decidimos representar esta relação como atributo em Álbum)

Álbum(TituloAlbum, NomeArtista, OrdemCronologica, Genero, AnoDeLancamento)

Na tabela Álbum incluímos além de seus atributos, o nome do artista como chave estrangeira (advindo do relacionamento entre Álbum e Artista) que faz parte da chave primária, o que apesar de aumentar o número de chaves primárias, garante que não seja necessário a criação de uma nova tabela e evita campos opcionais, já que todo álbum está relacionado a um artista.

Já sobre a relação entre Álbum e Música, decidimos representá-la como um atributo na tabela Música, para evitar atributos multivalorados, já que um álbum pode conter de 1 a n músicas.

Playlist(NomePl, Autor)

Na tabela Playlist, apenas incluímos seus atributos. Optamos por não incluir as relações referentes a ela, pois isso incluiria campos multivalorados ao mapear o relacionamento entre Playlist e Música, já que uma playlist pode ter uma ou mais músicas (por isso decidimos representar esta relação como atributo em Música).

Música(TituloMusica, TituloAlbum, NomePl, Duracao, Faixa)

Na tabela Música incluímos além de seus atributos, duas chaves estrangeiras, sendo uma delas parte da chave primária, assim como o título da música. Sendo esta o título do álbum ao qual ela pertence, representando a relação entre música e álbum. Decidimos representá-la desta forma para evitar junções e campos multivalorados. Decidimos colocá-la como parte da chave primária de Música, para que fosse possível uma música de mesmo nome, fazer parte de diferentes álbuns (por exemplo, caso uma música de mesmo nome fosse produzida por diferentes artistas que por consequência possuem álbuns de nomes diferentes).

A segunda chave estrangeira é o nome da playlist. Optamos por incluí-la na tabela

Música, para evitar junções e campos multivalorados (já que uma playlist pode ter diversas músicas). Optamos por não incluí-la como parte da chave primária, para que fosse possível que uma mesma música fizesse parte de diferentes playlists, sem ocasionar dependência funcional parcial.

Na figura 3, apresenta-se o diagrama gerado no BrModelo referente ao Modelo Relacional do minimundo especificado. Verifica-se sua corretude de acordo com a 2FN visto que não se encontram dependencias funcionais parciais, visto que na tabela Album, por exemplo, todos os atributos não-chave dependem da chave primária inteira e não parte dela. Tendo apenas o nome do album não deduz o ano de lançamento, gênero ou a ordem cronológica. Tendo apenas o nome do artista também não deduz os atributos não-chave. O mesmo ocorre na tabela música, onde a chave estrangeira NomePl não foi incluída como chave primária pois ocasionaria dependência funcional parcial, visto que a partir do TituloMusica e do TituloAlbum, deduz-se a duração da música e a faixa no álbum.

Já a 3FN, pode ser verificada no fato de que não há dependências entre atributos não-chave. Em todas as tabelas, os atributos não-chave sempre dependem da chave primária.

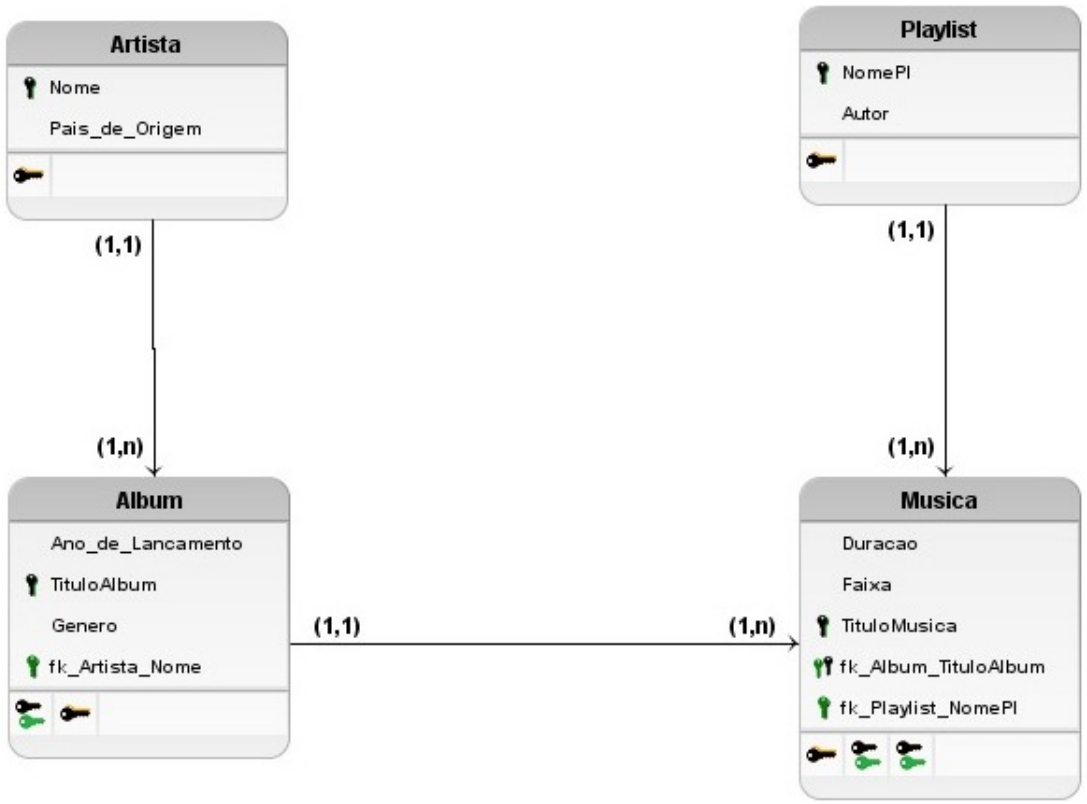

Figura 3 – Modelo Relacional gerado pelo BrModelo, onde a chave preta significa Chave Primaria e a chave verde representa a Chave Estrangeira

6 Projeto Físico do Banco de Dados

Para a criação do BD foram criadas 4 tabelas, conforme o modelo relacional, sendo elas: Artista, Album, Musica e Playlist. Os scripts para criação das tabelas se encontram no arquivo criacaotabelas.sql

Já a adoção de políticas de integridade no projeto consiste no uso das cláusulas: **NOT NULL**, **NOT NULL**, **CASCADE**, **PRIMARY KEY** e **FOREIGN KEY**.

Na tabela Artista, temos o atributo Nome como **PRIMARY KEY** e o PaisDeOrigem como **NOT NULL**, visto que o Artista é sempre identificado unicamente pelo nome e seu país de origem precisa ser inserido.

Na tabela Álbum, o atributo AnoDeLancamento é **NOT NULL**, assim como o atributo genero. Já o atributo TituloAlbum é **PRIMARY KEY** e ainda possui um atributo fkArtistaNome que é **FOREIGN KEY** referenciando o nome do artista na tabela Artista. Além disso a chave estrangeira fkArtistanome possui **CASCADE** que garante que quando seja deletado um album, os dados que dependem dele também sejam deletados. Por fim, o atributo AnoDeLancamento possui a restrição de que o ano deve ser maior ou igual que 1500.

A tabela Música, possui os atributos Duração e Faixa como **NOT NULL**, TituloMusica como **PRIMARY KEY**, juntamente com o TituloAlbum que é também **PRIMARY KEY** e **FOREIGN KEY**, de forma que uma Musica é identificada pelo seu nome e também pelo album em que se encontra, visto que podem existir músicas de mesmo nome em albuns diferentes. Além disso, as chaves estrangeiras possuem propriedade **CASCADE** para que ao deletar um Album, sejam deletados também as músicas dentro dele

Por fim, a tabela Playlist, possui como **PRIMARY KEY** o NomePl e o nome do Autor como **NOT NULL**.

Para povoar o banco de dados, foi criado um script chamado insercaotuplas.sql

6.1 Consultas

Consulta 1: Listar títulos de álbuns dado um certo gênero musical.

AR:

$$\sigma_{genero=<genero>}(Album)$$

SQL:

```
SELECT * FROM ALBUM WHERE genero = <genero>;
```

Consulta 2: Mostrar a média do número de músicas por álbum.

AR:

$$\Pi_{media}(AVG(COUNT(\sigma_{musica.tituloalbum}(MUSICA))))$$

SQL:

```
SELECT AVG(cnt) FROM (SELECT COUNT(*)
AS cnt, tituloalbum FROM musica GROUP BY tituloalbum) AS medias;
```

Consulta 3: Listar todas as músicas dado um certo álbum.

AR:

$$\sigma_{tituloalbum=<album>}(Musica)$$

SQL:

```
SELECT * FROM MUSICA WHERE tituloalbum = <album>;
```

Consulta 4: Listar nome do artista dado uma música.

AR:

$$\Pi_{nomeartista}(musica.tituloalbum = album.tituloalbum \cup musica.titulomusica =< musica >)(Album \cup Musica)$$

SQL:

```
SELECT ALBUM.nomeartista FROM ALBUM, MUSICA
WHERE MUSICA.tituloalbum = ALBUM.tituloalbum
AND MUSICA.titulomusica = <musica>;
```

Consulta 5: Consultar o número da faixa de uma música.

AR:

$$\Pi_{faixa}(\sigma_{titulomusica=<musica>}(musica))$$

SQL:

```
SELECT faixa FROM MUSICA WHERE titulomusica = <musica>;
```

Consulta 6: Listar todos os álbuns dado um certo ano de lançamento.

AR:

$$\sigma_{anodelancamento=<ano>}(album)$$

SQL:

```
SELECT * FROM ALBUM WHERE anodelancamento = <ano>;
```

Consulta 7: Listar todos os álbuns de um certo artista.

AR:

$\sigma_{nomedeartista=<nome>}(album)$

SQL:

```
SELECT * FROM ALBUM WHERE nomeartista = <nome>;
```

Consulta 8: Listar todas as músicas de um certo artista.

AR:

$\sigma_{musica.tituloalbum=album.tituloalbum \cup album.nomeartista=<nome>}(Album \cup Musica \cup Artista)$

SQL:

```
SELECT DISTINCT * FROM MUSICA, ALBUM, ARTISTA
WHERE MUSICA.tituloalbum = ALBUM.tituloalbum
AND ALBUM.nomeartista = <nome>;
```

Consulta 9: Listar artistas com albuns lançados em determinado ano.

AR:

$\Pi_{nomeartista}(\sigma_{artista.nome=album.nomeartista \cup album.ano=<ano>})(Album, Artista)$

SQL:

```
SELECT ARTISTA.nome FROM ALBUM, ARTISTA
WHERE ARTISTA.nome = ALBUM.nomeartista
AND ALBUM.anodelancamento = <ano>;
```

Consulta 10: Contagem de álbuns cadastrados.

AR:

$\sigma_{(COUNT(Album))}$

SQL:

```
SELECT COUNT(*) FROM ALBUM;
```

Consulta 11: Contagem de musicas cadastrados.

AR:

$\sigma_{(COUNT(Musica))}$

SQL:

```
SELECT COUNT(*) FROM ALBUM;
```

Consulta 12: Listar todos os álbuns cadastrados.

AR:

$$\Pi_{tituloalbum,nomeartista,genero,anodelancamento}(Album)$$

SQL:

```
SELECT(*) FROM ALBUM;
```

Consulta 13: Listar todos as músicas cadastradas.

AR:

$$\Pi_{titulomusica,tituloalbum,nomepl,duracao,faixa}(Musica)$$

SQL:

```
SELECT(*) FROM MUSICA;
```

Consulta 14: Listar todas as músicas de uma determinada playlist.

AR:

$$\Pi_{titulomusica,tituloalbum,nomepl,duracao,faixa}(\sigma_{nomepl=<playlist>}(Playlist))$$

SQL:

```
SELECT * FROM MUSICA WHERE nomepl = <playlist>;
```

Consulta 15: Contagem de músicas por playlist..

AR:

$$\Pi_{nomeplρ(COUNT(σmusica.nomepl=playlist.nomepl(Playlist∪Musica)))}$$

SQL:

```
SELECT PLAYLIST.nomepl AS Playlist, COUNT(*) AS Numero_Musicas
FROM MUSICA, PLAYLIST WHERE MUSICA.nomepl = PLAYLIST.nomepl
GROUP BY PLAYLIST.nomepl;
```

Consulta 16: Contar o numero de albuns de determinado artista, utilizando a view.

AR:

$$\Pi_{nomeartista,numeroalbuns}(NumeroAlbuns)$$

SQL:

```
SELECT * from NumeroAlbuns;
```

6.2 Triggers

Foram criadas três triggers para a operação de **DELETE**. O primeiro trigger serve para remover um Album automaticamente ao detectar que foi removida a ultima musica relacionada a ele. O segundo trigger serve para remover um Artista ao detectar que foi removido o ultimo álbum relacionado a ele. Por fim, o terceiro trigger deleta uma Playlist caso detecte que foi removida a ultima música relacionada a ela.

Trigger 1: Verifica se a tupla deletada de Muisca era a ultima referenciando o album e, caso isso aconteça, deleta também o album correspondente na tabela Album.

```
  CREATE OR REPLACE FUNCTION verifica_musica() RETURNS trigger
  AS $verifica_musica$
    BEGIN
      IF (TG_OP = 'DELETE') THEN
           IF NOT EXISTS (SELECT 1 FROM MUSICA
            WHERE tituloalbum = OLD.tituloalbum) THEN DELETE FROM ALBUM
            WHERE ALBUM.tituloalbum = OLD.tituloalbum;
            RETURN OLD;
              END IF;
        END IF;
        RETURN NULL;
    END
$verifica_musica$ LANGUAGE plpgsql;

CREATE TRIGGER verifica_musica AFTER DELETE ON MUSICA
   FOR EACH ROW EXECUTE PROCEDURE verifica_musica();
```

Trigger 2: Verifica se a tupla deletada de Album era a ultima referenciando o artista e, caso isso aconteça, deleta também o artista correspondente na tabela Artista.

```
  CREATE OR REPLACE FUNCTION verifica_artista() RETURNS trigger
  AS $verifica_artista$
  BEGIN
    IF (TG_OP = 'DELETE') THEN
      IF NOT EXISTS (SELECT 1 FROM ALBUM WHERE nomeartista = OLD.nomeartista)
        THEN DELETE FROM ARTISTA
            WHERE ARTISTA.nome = OLD.nomeartista;
        RETURN OLD;
      END IF;
    END IF;
    RETURN NULL;
  END
$verifica_artista$ LANGUAGE plpgsql;

CREATE TRIGGER verifica_artista AFTER DELETE ON ALBUM
   FOR EACH ROW EXECUTE PROCEDURE verifica_artista();
```

Trigger 3: Verifica se a tupla deletada de Muisca era a ultima referenciando a playlist e, caso isso aconteça, deleta também a playlist correspondente na tabela Playlist.

```
  CREATE OR REPLACE FUNCTION verifica_playlist() RETURNS trigger
  AS $verifica_playlist$
    BEGIN
        IF (TG_OP = 'DELETE') THEN
            IF NOT EXISTS (SELECT 1 FROM MUSICA WHERE nomepl = OLD.nomepl)
              THEN DELETE FROM PLAYLIST
                  WHERE PLAYLIST.nomepl = OLD.nomepl;
                  RETURN OLD;
              END IF;
          END IF;
          RETURN NULL;
      END
$verifica_playlist$ LANGUAGE plpgsql;

CREATE TRIGGER verifica_playlist AFTER DELETE ON MUSICA
   FOR EACH ROW EXECUTE PROCEDURE verifica_playlist();
```

6.3 View

Durante este projeto também foi desenvolvida uma View, para que fosse possível a obtenção do campo calculado na tabela Artista (Número de Álbuns). Esta view possibilita a consulta da quantidade de álbuns que os artistas possuem a qualquer momento, de maneira que basta fazer uma simples consulta da view, para que se obtenha a quantidade de álbuns pertencentes a um artista específico. Esta também possibilita uma listagem de todos os artistas com seu número de albuns cadastrados. Além disso, esta view cumpre todas as características necessárias para que ela esteja sempre atualizada, este foi um dos grandes motivos pelos quais foi optado pela view no lugar de uma tabela temporária.

Implementação da View:

```
CREATE VIEW Numero_albuns AS
      SELECT ARTISTA.Nome as ARTISTA, count(*) as NUMERO_ALBUNS
      FROM ALBUM, ARTISTA
      WHERE ARTISTA.Nome = ALBUM.NomeArtista
      GROUP BY ARTISTA.Nome;
```

7 Considerações Finais

Durante o trabalho pudemos ter um contato maior com a linguagem de consultas SQL utilizando o SGDB PostgreSQL. Através de um trabalho prático pudemos compreender melhor na prática o funcionamento de um BD, bem como o projeto do mesmo em todas suas etapas, desde a elaboração do modelo entidade-relacionamento até a implementação em SQL. A maior dificuldade encontrada foi na elaboração do MER, onde não tinhamos total certeza da corretude do mesmo. Com a ajuda da professora, pudemos então corrigir, a cada fase, os erros cometidos, feedback que foi essencial para a progressão no trabalho.