

Esse material foi produzido por Maria Luiza Mondelli e está licenciado com a licença Attribution-NonCommercial 4.0 International (CC BY-NC 4.0)

Para mais informações sobre a licença, acesse: <https://br.creativecommons.org/licencas/>



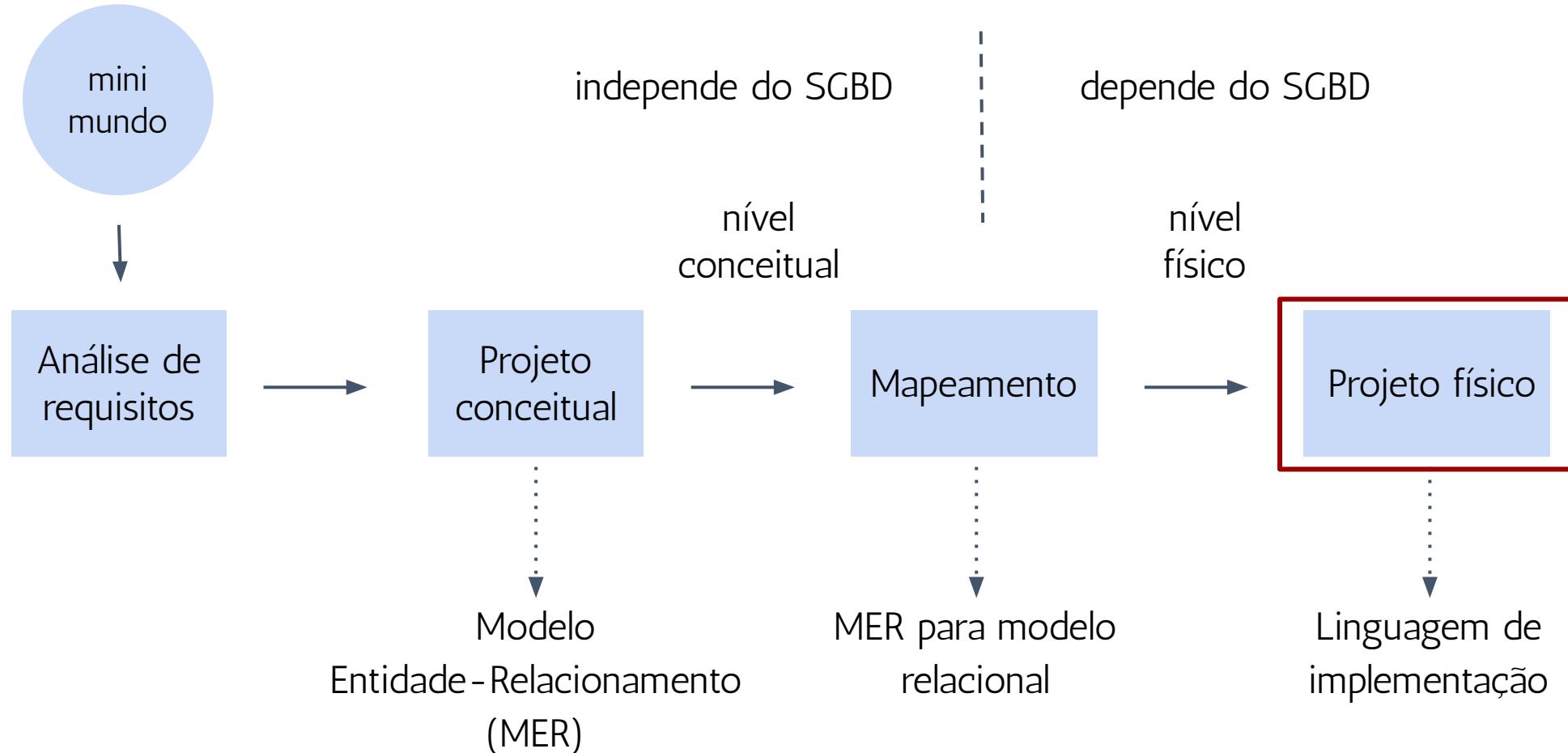
Atribuição-NãoComercial

CC BY-NC

Esta licença permite que outros remixem, adaptem e criem a partir do seu trabalho para fins não comerciais, e embora os novos trabalhos tenham de lhe atribuir o devido crédito e não possam ser usados para fins comerciais, os usuários não têm de licenciar esses trabalhos derivados sob os mesmos termos.



Projeto de Banco de Dados



SQL

Structured Query language

- É uma linguagem de pesquisa **declarativa** padrão para banco de dados relacionais

- Tipo de paradigma onde o **foco** não está no "como" e sim **no "que"**

Mais detalhes sobre as diferenças de paradigmas [aqui](#).

- Criada no início dos anos 70 com o objetivo de demonstrar que era viável implementar o modelo relacional

- Proposto por E. F. Codd, membro do laboratório de pesquisa da IBM em San Jose, CA

- Padrão utilizado pelos sistemas de banco de dados relacionais

- Pode ser dividida nas seguintes categorias:

- DQL - Data Query Language
 - DDL - Data Definition Language
 - DML - Data Manipulation Language
 - DCL - Data Control Language
 - DTL - Data Transaction Language



SQL

Resumindo o que vamos ver

- **DDL**

CREATE TABLE
CREATE INDEX
CREATE VIEW
ALTER TABLE
ALTER INDEX
DROP INDEX
DROP VIEW

- **DML**

INSERT
UPDATE
DELETE

- **DQL**

SELECT

- **Cláusulas**
FROM, WHERE
GROUP BY, HAVING
ORDER BY, DISTINCT
UNION

- **Operadores lógicos e relacionais**

AND, OR, NOT
>, >=, <, <=, ==><>
BETWEEN, IN, LIKE

- **Funções de agregação**

MAX, MIN
AVG, SUM, COUNT

- **Junções**

INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL OUTER JOIN

- **DCL**

GRANT
REVOKE

- **Backup e Restauração**



Volte aqui sempre que estiver agarrado num erro de SQL

(aqui vai um checklist de verificações, pode ser que esse slide dê uma luz)

- Olhe a mensagem de erro. Ela dá alguma dica?
- Execute uma declaração SQL por vez
 - seu cursor está na linha correta a ser executada?
 - tente selecionar apenas bloco de linhas que precisa ser executado, e execute.
 - pode ser uma boa adicionar um ';' ao final da sua consulta.
- Você está conectado à base de dados correta?
- Erros de digitação são mais comuns do que a gente pensa
 - os nomes das tabelas/colunas que escreveu estão de acordo com a estrutura de tabelas/colunas que de fato existem no banco de dados?
 - as declarações SQL estão escritas corretamente?
- A sua consulta SQL está escrita seguindo a estrutura/ordem correta?



SQL

DDL - Data Definition Language

Tabelas

CREATE TABLE

Utilizado para criar novas tabelas no banco de dados. É necessário definir o nome da tabela, assim como o nome e o tipo de dados de cada coluna.

```
CREATE TABLE nome_tabela (coluna_1 datatype, coluna_2 datatype, coluna_3 datatype);
```

ALTER TABLE

Permite que a estrutura da tabela seja modificada, adicionando ou excluindo novas colunas ou modificando o tipo de dados de alguma coluna, por exemplo.

```
ALTER TABLE nome_tabela ADD nome_coluna datatype;  
ALTER TABLE nome_tabela DROP COLUMN nome_coluna;  
ALTER TABLE nome_tabela MODIFY COLUMN nome_coluna datatype;
```



SQL

DDL - Data Definition Language Tabelas

DROP TABLE

Permite a exclusão de tabelas. Se existirem registros na tabela, os mesmos também serão excluídos. Deve-se tomar atenção no que diz respeito às violação das restrições de integridade (chaves estrangeiras, por exemplo)

```
DROP TABLE nome_tabela;  
DROP DATABASE nome_database;
```



SQL

DDL - Data Definition Language

Constraints

Usadas para criar regras para os atributos nas tabelas do banco de dados

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column when no value is specified
- **INDEX** - Used to create and retrieve data from the database very quickly

Mais detalhes [aqui](#).



SQL

DDL - Data Definition Language Índices

- Os índices são usados para recuperar dados mais rapidamente. Os usuários não podem ver os índices, eles são usados apenas para acelerar consultas.

CREATE INDEX

Permite a criação de índices a partir de uma ou mais colunas.

```
CREATE INDEX nome_indice ON nome_tabela (nome_coluna);
CREATE UNIQUE INDEX nome_indice ON nome_tabela (nome_coluna1, nome_coluna2);
```



SQL

DDL - Data Definition Language Índices

ALTER

INDEX

(extensão do Postgres, pode não ser o mesmo para outros SGBDs)

Permite alterar a definição de um índice.

```
ALTER INDEX nome indice RENAME TO novo nome;
```

DROP

INDEX

Permite a exclusão de índices. As colunas continuarão existindo, apenas o índice associado será removido.

```
DROP
```

```
INDEX
```

```
nome indice;
```



SQL

DDL - Data Definition Language

Views

- Views são tabelas virtuais formadas a partir de consultas SQL, contendo linhas e colunas como se fosse uma tabela do modelo do banco de dados. Também podem ser entendidas como consultas armazenadas.

CREATE VIEW

Permite a criação de views a partir da definição de uma consulta.

Detalhes sobre consultas serão apresentados em slides mais à frente..

```
CREATE VIEW nome_view AS  
SELECT coluna1, coluna2  
FROM nome_tabela  
WHERE condição;
```



SQL

DDL - Data Definition Language Views

CREATE OR REPLACE VIEW

Permite a atualização de views a partir da definição de uma consulta.

```
CREATE OR REPLACE VIEW nome_view AS
  SELECT coluna1, coluna2
    FROM nome_tabela
   WHERE condição;
```

DROP

VIEW

Permite a exclusão de views. As tabelas utilizadas na criação da view não são removidas..

DROP

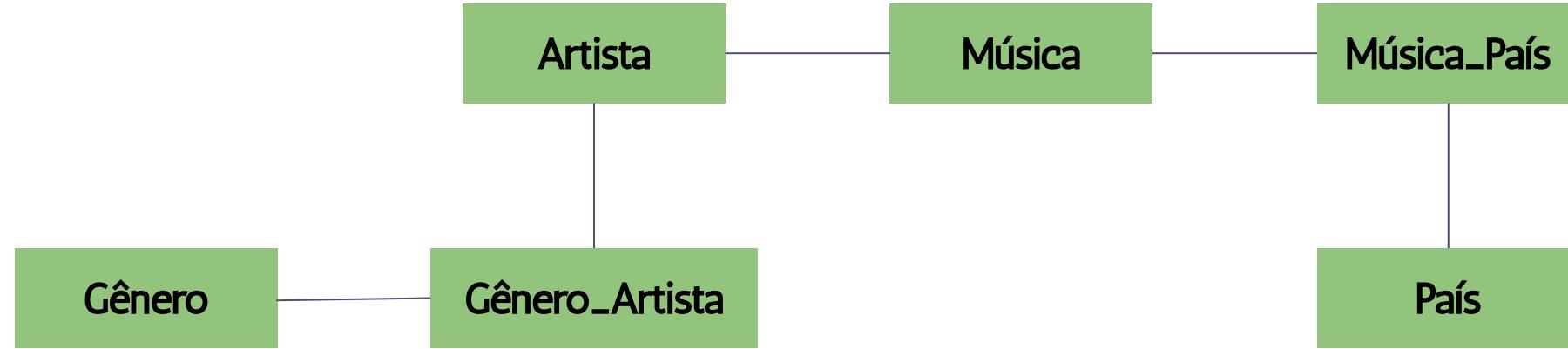
VIEW

nome_view;



Criação do Modelo

Exemplo - Spotify



SQL

DML - Data Manipulation Language

INSERT

Utilizado na inserção de novos registros em uma tabela.

```
INSERT INTO nome_tabela (coluna_1,coluna_2,coluna_3)
VALUES (valor_1, 'valor_2', valor_3);
```

UPDATE

Permite a edição/alteração de registros em uma tabela.

```
UPDATE
SET                 alguma_coluna
WHERE alguma_coluna = algum_valor;
```

DELETE

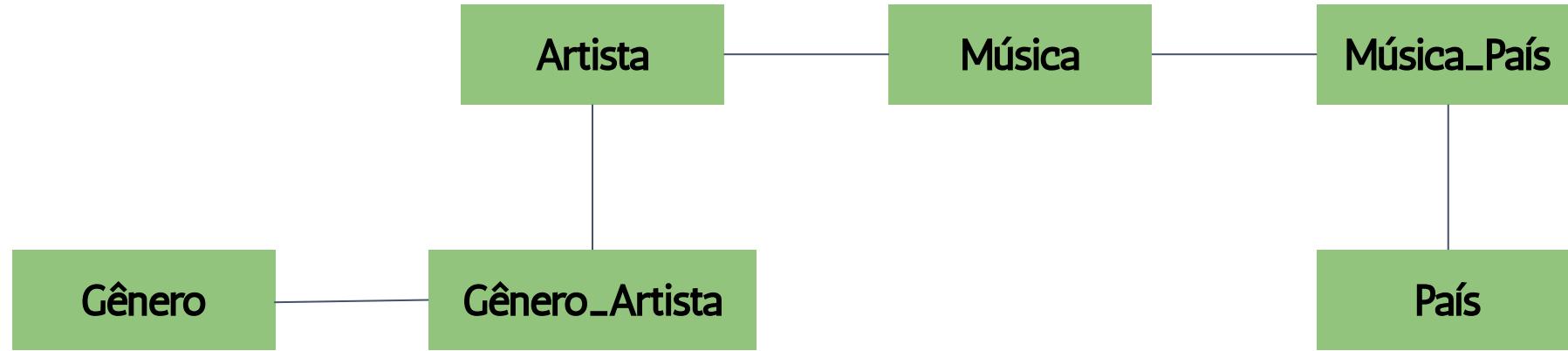
Permite que registros sejam deletados de uma tabela.

```
DELETE FROM nome_tabela    nome_tabela
wHERE alguma_coluna = algum_valor;lor
```



Inserção de Dados

Exemplo - Spotify



SQL

DQL - Data Query Language

- Compreende o comando responsável por realizar consultas ao banco de dados: **SELECT**
- Pode ser combinado com diferentes **cláusulas** e **operadores** para a construção de consultas tanto simples quanto mais complexas

Exemplo (mais simples possível, sem cláusulas/operadores)

`SELECT 1 as id;`



Neste exemplo estamos selecionando o *número 1* com o 'apelido' *id*.

No entanto, o ideal e mais comum é utilizarmos as cláusulas para indicar qual será a *origem* das informações que estamos consultando/buscando no banco de dados.



SQL

Cláusulas

aluno

| matricula | nome | idade |
|-----------|---------------|-------|
| 123456789 | João Silveira | 22 |

- Condições de modificação que definem/restringem os dados que serão selecionados ou modificados na consulta

FROM

Especifica a fonte (tabela) dos dados a serem recuperadas, podendo ser apenas uma ou várias.
É utilizada no comando SELECT e forma a base de qualquer consulta SQL.

```
SELECT
  FROM nome_tabela;
```



```
SELECT matricula, nome
  FROM aluno;
```

WHERE

Restringe os dados através de operações que testam se cada registro satisfaz ou não a condição estabelecida.

```
SELECT
  FROM nome_tabela
 WHERE nome_coluna operador valor;
```



```
SELECT matricula, nome
  FROM aluno
 WHERE idade > 18;
```

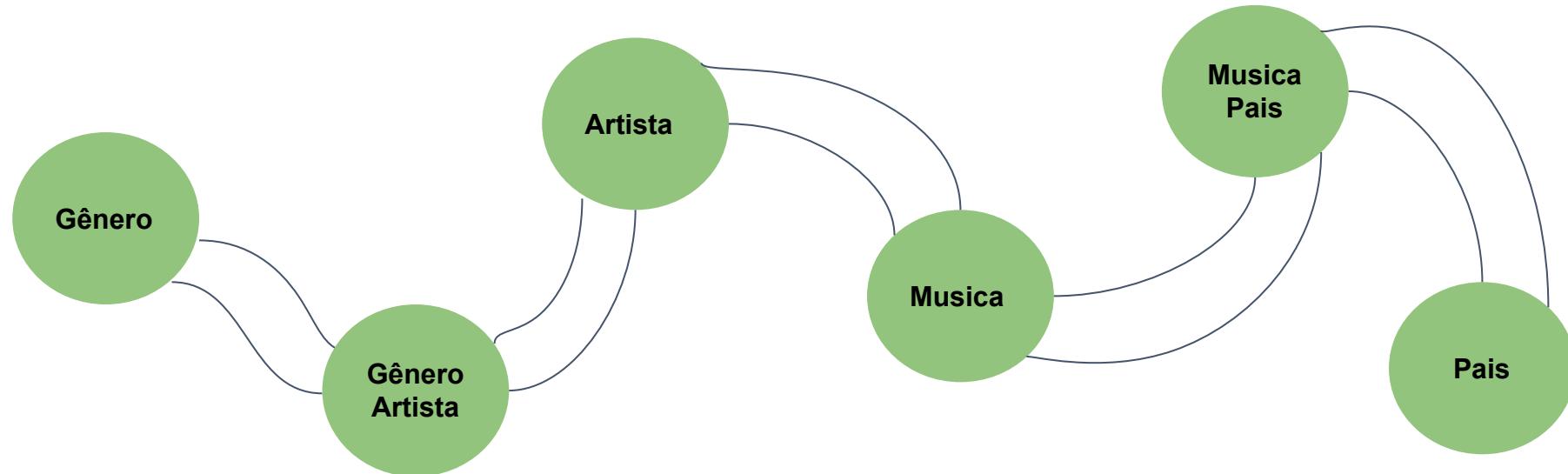


SQL

DQL - Data Query Language

Como pensar nas consultas? **Um caminho a ser percorrido.**

Esse caminho determina as **junções** de tabelas que precisam ser feitas.

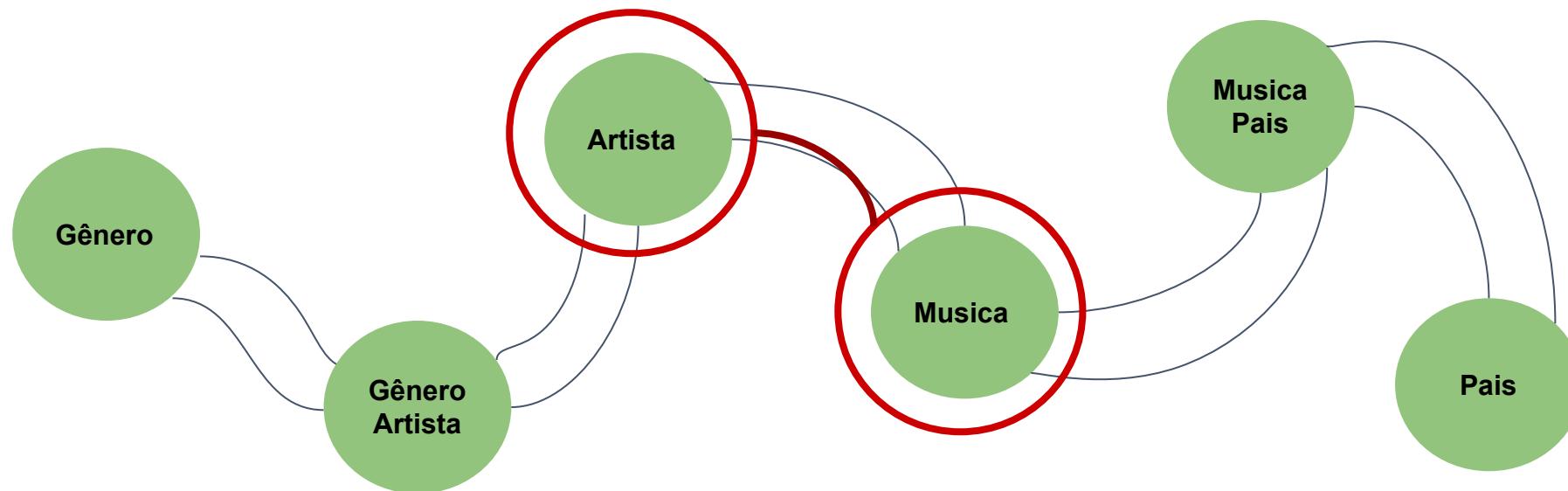


SQL

DQL - Data Query Language

O que precisa ser selecionado? **De onde** precisa ser selecionado?

Exemplo: Selecionar nome do artista e nome da música.

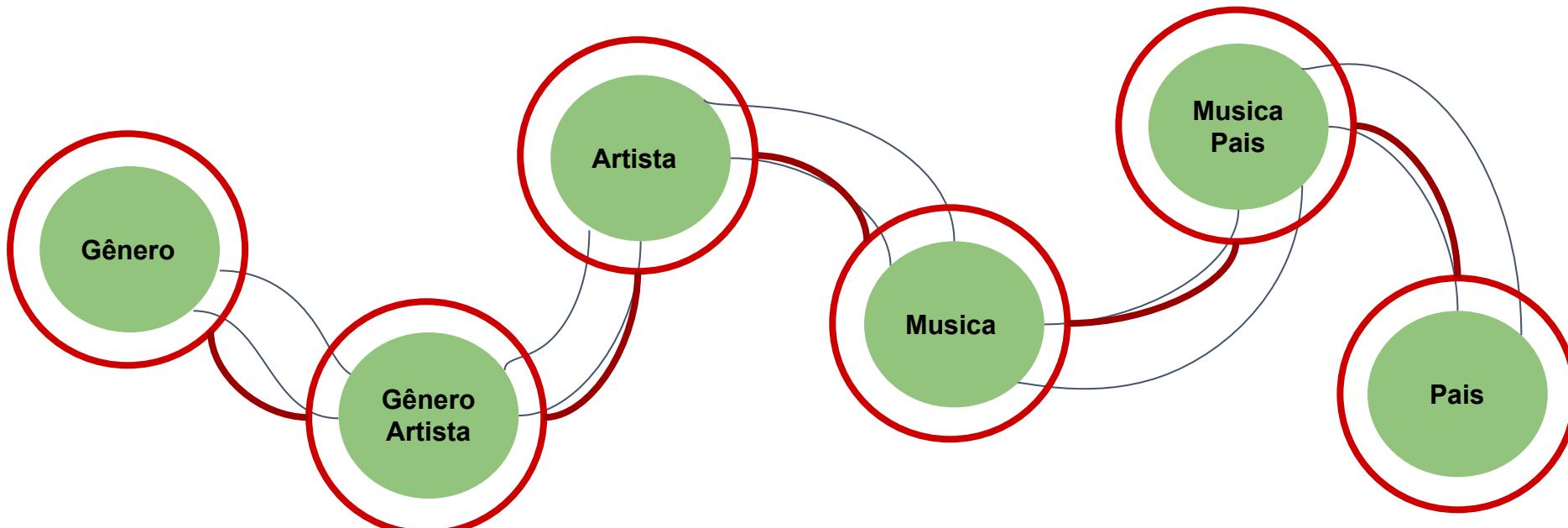


SQL

DQL - Data Query Language

O que precisa ser selecionado? **De onde** precisa ser selecionado?

Exemplo: Selecionar os países que tocam o gênero *latino*
ou ainda, selecionar a listagem de gêneros que tocam em cada país



SQL

Cláusulas

GROUP BY

Utilizada em conjunto com funções de agregação para o agrupamento do resultado por uma ou mais colunas.

```
SELECT COUNT(*)  
      FROM nome_tabela  
     GROUP BY nome_coluna;
```

HAVING

Utilizada apenas quando aplicada a cláusula GROUP BY, a fim de limitar os dados recuperados de acordo com alguma restrição.

```
SELECT nome_coluna, função_agregação(nome_coluna)  
      FROM nome_tabela  
     WHERE nome_coluna operador valor  
     GROUP BY nome_coluna  
    HAVING função_agregação(nome_coluna) operador valor;
```



SQL

Cláusulas

ORDER BY

Utilizada para ordenar os registros, levando em consideração uma ou mais colunas

```
SELECT  
FROM  
ORDER BY nome_coluna ASC|DESC;
```

nome_coluna
nome_tabela

DISTINCT

Utilizada para selecionar dados sem que haja repetição.

```
SELECT DISTINCT nome_coluna  
FROM nome_tabela;
```

LIMIT

Especifica um valor máximo de linhas que será retornado.

```
SELECT  
FROM  
LIMIT valor;
```

nome_coluna(s)
nome_tabela



SQL

Operadores lógicos e relacionais

- **AND:** **E** lógico.
Avalia as condições e retorna as colunas se todas as condições forem verdadeiras.
- **OR:** **OU** lógico.
Avalia as condições e retorna as colunas se alguma das condições for verdadeira..
- **NOT:** Negação lógica.
Retorna as colunas o valor contrário da expressão seja verdadeiro.

```
SELECT nome_coluna(s)
FROM nome_tabela
WHERE nome_coluna_1 = valor_1
AND|OR nome_coluna_2 = valor_2;
```

```
SELECT nome_coluna(s)
FROM nome_tabela
WHERE NOT nome_coluna = valor;
```



SQL

Operadores lógicos e relacionais

| Operador | Função |
|----------|----------------|
| < | Menor |
| > | Maior |
| <= | Menor ou igual |
| >= | Maior ou igual |
| == | Igual |
| <> | Diferente |

| Operador | Função |
|----------|---|
| BETWEEN | <code>WHERE nome_coluna BETWEEN valor_1 AND valor_2;</code> |
| LIKE | <code>WHERE nome_coluna LIKE padrão;</code> |
| IN | <code>WHERE nome_coluna IN (valor_1,valor_2,..., valor_n);</code> |



SQL

Funções de agregação

AVG

Retorna a média dos valores de uma determinada coluna.

```
SELECT      AVG(nome_coluna)
FROM nome_tabela;
```

SUM

Retorna a soma dos valores de uma determinada coluna

```
SELECT      SUM(nome_coluna)
FROM nome_tabela;
```

COUNT

Retorna o número de linhas onde a coluna possui valor diferente de NULL.

```
SELECT      COUNT(nome_coluna)
FROM nome_tabela;
```

MIN E MAX

Retorna o menor ou maior valor encontrado na coluna passada como argumento.

```
SELECT MIN(nome_coluna)
FROM nome_tabela;
```



Consultas Básicas

Exemplo - Spotify

- Consultar nome dos artistas
- Consultar nome dos artistas sem repetição
- Quantidade total de gêneros distintos
- Quantidade total de músicas distintas
- Listar nome dos artistas e popularidade, ordenando pela popularidade (asc)

Ideias?

