

Introdução a Banco de Dados – Modelo Físico

Bruno Costa

Sumário

- Projeto Físico (3ª Fase)
- MySQL e SQL
- DDL e DML
- Tipos de Dados
- Comandos DDL (Create, Drop e Alter)
- Comando DML (Select, Insert, Update e Delete)

SQL

- Hoje é expandido para *Structured Query Language* ou seja Linguagem de Consulta Estruturada.
- Foi criada e implementada pela IBM
- É a linguagem padrão para SGBDs relacionais

SQL

- É uma linguagem de banco de dados abrangente tem instruções para definição de dados, consultas e atualizações.
- É dividida em: DDL E DML.
- **DDL – Linguagem de Definição de Dados:** fornece comandos para definições de esquemas de relação, criação/remoção de tabelas, criação de índices e modificação de esquemas
- **DML – Linguagem de Manipulação de Dados:** inclui uma linguagem de consulta baseada na álgebra relacional e cálculo relacional de tupla. Compreende comandos para inserir, consultar, remover e modificar tuplas num Banco de Dados.

SQL

- A SQL usa os termos **tabela**, **linha** e **coluna** para os termos do modelo relacional formal *relação*, *tupla* e *atributo*, respectivamente.
- Obs.: A linguagem SQL não é case sensitive ou seja não diferencia letras maiúsculas de minúsculas.

Tipos de Dados

- Alguns tipos de Dados mais comuns são:
 - CHAR,
 - VARCHAR,
 - TEXT
 - INT,
 - BIGINT
 - DECIMAL,
 - DATE,
 - DATETIME,
 - TIME.

Tipos de Dados

- Caracteres:
- **CHARACTER(X)** ou **CHAR** – Representa um String de tamanho x.
- **CHARACTER VARYING(X)** ou **VARCHAR** – representa um String de tamanho x. Pode conter dados de textos de até 255 caracteres de tamanho.
- **TEXT** – é utilizado quando se deseja armazenar uma quantidade grande de texto, pois não tem um tamanho definido.

Tipos de Dados

- Números Exatos:
- **INTEGER** ou **INT** – utilizado para armazenar números inteiros.
- **BIGINT** – é utilizado quando se deseja armazenar valor 'numéricos extensos.
- **DECIMAL**– utilizado para armazenar números decimais, tem uma precisão e uma escala (números de dígitos na parte fracionária). Muito usado para representar dinheiro.
- Sintaxe: DECIMAL(5,2).
- *os números mostram quantos dígitos o banco de dados deve esperar em frente dos decimais, e quantos depois.*
- *Exe.: 19892.23 – cinco antes e 2 depois.*

Tipos de Dados

- Datetimes:
- **DATE**: armazena ano(4 dígitos), mês (2dígitos) e dia (2 dígitos). O Tipo de dado dever ser informados no padrão internacional.
- *Exemplo: 1989-06-01*
- **TIME**: armazena hora (2 dígitos), minuto (2 dígitos) e segundo (2 dígitos);
- **DATETIME** – armazena data e hora.

Comandos DDL

SQL – COMANDOS BÁSICOS

- **Comando da Linguagem de definição de dados (DDL)**
- CREATE – criar banco de dados ou tabela
- DROP – apagar banco de dados ou tabela
- ALTER – alterar tabelas(adicionar colunas, renomear colunas ou tabelas, alterar tipos de dados de colunas)

COMANDO CREATE

- É utilizado para basicamente criar banco de dados como as tabelas que farão parte dele.
- Sintaxe para criação do banco

```
mysql> CREATE DATABASE nome_do_banco;
```

É o comando

É o nome do banco

A instrução é encerrada com um ponto de vírgula (;)

```
mysql> CREATE DATABASE nome_do_banco;  
Query OK, 1 row affected (0.00 sec)
```

Esta ultima linha será o resultado do SGBD informando que uma *linha foi afetada ou modificada, construída etc.*

CREATE DATABASE

- Depois que o banco de dados é criado é preciso informar ao SGBD que você irá utilizá-lo.
- Através do comando **USE**.

```
mysql> use nome_do_banco;  
Database changed  
mysql>
```

- O SGBD sempre dará uma resposta caso esteja correta ou não!
- Neste ele informou que o “banco de dados mudou/foi alterado” agora podemos trabalhar em cima do banco.

Comando Create Table

- CREATE TABLE nome_da_tabela
- (- nome_da_coluna1 tipo_de_dado,
- Nome_da_coluna2 tipo_de_dado
-);

Comando Create Table

- Exemplo:
- CREATE TABLE Autor
(
 cod_autor INT(12) PRIMARY KEY,
 nome varchar(30) NOT NULL,
 nascimento date NOT NULL,
);

CREATE TABLE

- Exemplo:
- CREATE TABLE LIVRO (
 cod_livro INT(12) PRIMARY KEY,
 nome VARCHAR(30) NOT NULL,
 fk_cod_autor INT(12) NOT NULL,
 FOREIGN KEY (cod_autor) REFERENCES Autor(cod_autor)
);

COMANDO ALTER

- As definições de uma tabela básica ou de outros elementos do esquema poderão ser alterados pelo comando **ALTER**.
- **Alter Table**
 - Permite alterar ou adicionar atributos de uma determinada tabela.
 - Utilizado na alteração ou exclusão das restrições da tabela.

Alter Table

- **Sintaxe para adicionar uma nova coluna**

- ALTER TABLE nome_tabela ADD nome_coluna *tipo de dado*;

- **Sintaxe para modificar uma coluna de uma tabela**

- ALTER TABLE nome_tabela MODIFY nome_coluna *tipo de dado*;

- **Sintaxe para remover uma coluna de uma tabela**

- ALTER TABLE nome_tabela DROP nome_coluna;

- **Sintaxe para alterar atributo(coluna) de uma tabela**

- ALTER TABLE nome_tabela CHANGE nome_coluna nome_coluna_nova [tipo_de_dado];

Alter Table

- **Sintaxe para adicionar uma restrição a uma tabela**
 - ALTER TABLE nome_tabela ADD CONSTRAINT nome_restrição *nome_coluna*;
- **Sintaxe para remover restrição**
 - ALTER TABLE nome_tabela DROP nome_restrição;
- **Sintaxe para remover vínculo de chave estrangeira**
 - ALTER TABLE nome_tabela DROP FOREIGN KEY <nome_coluna_fk>;

Alter Table

- ALTER TABLE livro ADD idioma varchar(15)
 - Adiciona a coluna idioma de tamanho 15 na tabela livro
- ALTER TABLE livro MODIFY idioma varchar(30)
 - Altera a coluna idioma para o tamanho 30
- ALTER TABLE livro DROP idioma
 - Exclui a coluna idioma
- ALTER TABLE livro DROP PRIMARY KEY (coluna)
 - Exclui a chave primária da tabela Livro
- ALTER TABLE livro ADD CONSTRAINT PRIMARY KEY (coluna)
 - Adiciona chave primária
- ALTER TABLE livro ADD CONSTRAINT FOREIGN KEY (coluna) REFERENCES nome_da_tabela (coluna)
 - Adiciona chave estrangeira

Chave Estrangeira com ON UPDATE e ON DELETE

- Existem algumas opções aplicáveis às chaves estrangeiras que auxilia a manter a integridade dos dados nas tabelas do banco de dados;
- **Exemplo de criação de chave estrangeira:**
 - **...CONSTRAINT FOREIGN KEY** (*nome_coluna_chave_estrangeira*) **REFERENCES** *nome_tabela_pai* (*nomes_colunas_tabela_pai*) **ON DELETE** [*ação referencial*] **ON UPDATE** [*ação referencial*];
 - **ON DELETE** significa que a ação referencial será executada quando um registro for excluído da tabela pai.
 - **ON UPDATE** significa que a ação referencial será executada quando um registro for alterado da tabela pai.

Chave Estrangeira com ON UPDATE e ON DELETE

- A principais opções para as ações referenciais são:
- **CASCADE:** A opção CASCADE permite excluir ou atualizar os registros relacionados presentes na tabela filha automaticamente, quando um registro da tabela pai for atualizado (**ON UPDATE**) ou excluído (**ON DELETE**). É a opção mais comum aplicada.
- **RESTRICT:** Impede que ocorra a exclusão ou a atualização de um registro da tabela pai, caso ainda hajam registros na tabela filha. Uma exceção de violação de chave estrangeira é retornada. A verificação de integridade referencial é realizada antes de tentar executar a instrução UPDATE ou DELETE; Essa é a ação padrão ao se criar chave estrangeira
- **SET NULL:** Esta opção é usada para definir com o valor NULL o campo na tabela filha quando um registro da tabela pai for atualizado ou excluído.

ON UPDATE CASCADE e ON DELETE CASCADE

- Vamos criar 2 tabelas:
- **CREATE TABLE Pai** (id_Pai **INT PRIMARY KEY**, nome_Pai **VARCHAR(50)**);
- **CREATE TABLE Filho** (id_Filho **INT AUTO_INCREMENT PRIMARY KEY**, nome_Filho **VARCHAR(50)**, id_Pai **INT**, **CONSTRAINT FOREIGN KEY** (id_Pai) **REFERENCES** Pai(id_Pai) **ON DELETE CASCADE ON UPDATE CASCADE**);
- As tabelas ficariam assim, depois de preenchidas

| id_Pai | nome_Pai |
|--------|----------|
| 1 | João |
| 2 | Mário |
| 3 | Renato |
| 4 | Emerson |
| 5 | José |

| id_Filho | nome_Filho | id_Pai |
|----------|------------|--------|
| 1 | Maria | 1 |
| 2 | José | 1 |
| 3 | Carla | 2 |
| 4 | Tiago | 3 |
| 5 | Julia | 5 |

ON UPDATE CASCADE e ON DELETE CASCADE

- Excluindo registros da tabela pai:
- **DELETE FROM** Pai **WHERE** id_Pai = 1;
- Após isso a tabela filho ficaria assim:

| id_Filho | nome_Filho | id_Pai |
|----------|------------|--------|
| 3 | Carla | 2 |
| 4 | Tiago | 3 |
| 5 | Julia | 5 |

- Todo registro que tinha o id_Pai = 1 foi removido de forma cascadeada

ON UPDATE SET NULL e ON DELETE SET NULL

- Vamos criar 2 tabelas:
- **CREATE TABLE Pai** (id_Pai **INT PRIMARY KEY**, nome_Pai **VARCHAR(50)**);
- **CREATE TABLE Filho** (id_Filho **INT AUTO_INCREMENT PRIMARY KEY**, nome_Filho **VARCHAR(50)**, id_Pai **INT**, **CONSTRAINT FOREIGN KEY** (id_Pai) **REFERENCES** Pai(id_Pai) **ON DELETE SET NULL ON UPDATE SET NULL**);
- As tabelas ficariam assim, depois de preenchidas

| id_Pai | nome_Pai |
|--------|----------|
| 1 | João |
| 2 | Mário |
| 3 | Renato |
| 4 | Emerson |
| 5 | José |

| id_Filho | nome_Filho | id_Pai |
|----------|------------|--------|
| 1 | Maria | 1 |
| 2 | José | 1 |
| 3 | Carla | 2 |
| 4 | Tiago | 3 |
| 5 | Julia | 5 |

ON UPDATE SET NULL e ON DELETE SET NULL

- Excluindo registros da tabela pai:
- **DELETE FROM** Pai **WHERE** id_Pai = 1;
- Após isso a tabela filho ficaria assim:

| id_Filho | nome_Filho | id_Pai |
|----------|------------|--------|
| 1 | Maria | NULL |
| 2 | José | NULL |
| 3 | Carla | 2 |
| 4 | Tiago | 3 |
| 5 | Julia | 5 |

- Todo registro que tinha o id_Pai = 1 a coluna com id_Pai mostra o valor NULL obnde houve a exclusão.

ON UPDATE e ON DELETE – Dica para testar os comandos sem apagar a tabela

- Para não precisar apagar e criar as tabelas podemos remover a chave estrangeira.
- 1º execute o comando **SHOW CREATE TABLE** Filho; será mostrado algo como:

```
CREATE TABLE `Filho` (  
  `id_Filho` int(11) NOT NULL AUTO_INCREMENT,  
  `nome_Filho` varchar(50) DEFAULT NULL,  
  `id_Pai` int(11) DEFAULT NULL,  
  PRIMARY KEY (`id_Filho`),  
  KEY `id_Pai` (`id_Pai`),  
  CONSTRAINT `Filho_ibfk_1` FOREIGN KEY (`id_Pai`) REFERENCES `Pai` (`id_Pai`)  
  ON DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

- Aí é só copiar o nome da chave estrangeira: 'Filho_ibfk_1' é ela que iremos remover.
- Em seguida executamos o código: **ALTER TABLE** curso **DROP FOREIGN KEY** Filho_ibfk_1
- E a chave estrangeira está removida.

ON UPDATE e ON DELETE - DICA

- Para não precisar apagar e criar as tabelas podemos remover a chave estrangeira.
- Basta realizar a adição da chave estrangeira com a ação referencial desejada, exemplo:
- **ALTER TABLE** Filho **ADD CONSTRAINT FOREIGN KEY**(id_Pai) **REFERENCES** Pai(id_Pai)
ON DELETE SET NULL
ON UPDATE SET NULL;

Comandos DML

SQL – COMANDOS BÁSICOS

- Comando da Linguagem de Manipulação de Dados(DML)
- Permite a visualização e alteração do conteúdo dos dados de tabelas básicas.
- INSERT – Inclusão de uma ou várias tuplas em uma tabelas
- SELECT – Seleção de atributos de uma tabela
- UPDATE – Atualização de valores de atributos
- DELETE – Remoção de uma ou várias tuplas de uma tabela

Comando INSERT

- INSERT INTO nome_tabela (n_coluna1, n_coluna2, ...) VALUES ('valor1', 'value', ...);
- INSERT INTO – a palavra chave *inicia a declaração*
- Logo depois o nome_tabela a qual os dados serão inseridos
- Entre parênteses o nome das colunas que compoem esta tabela
- VALUES indica que valores serão inseridos
- ('value1' , 'value2' , ...) os valores serão inseridos de acordo com as colunas.
- **Obs: os valores precisam estar na mesma ordem que os nome das colunas**
- As aspas simples são utilizadas para inserir caracteres, textos
- Os números são escritos sem aspas

COMANDO SELECT

- SELECT – especifica as colunas da tabela
- FROM – especifica as tabelas;
- WHERE – especifica as linhas;
- *O Comando **SELECT** elimina os resultados para nós e exibe somente as linhas que são compatíveis com a condição estabelecida.*
- *A cláusula **WHERE** permite que o sistema busca algo específico*

COMANDO SELECT - Sintaxe

A. *Selecionando Colunas Específicas da Tabela*

- SELECT <nome(s) da(s) coluna(s)
 - FROM <tabela>;
-
- Exe.: Selecionar o valor e a descrição de cada produto.
 - **SELECT** valor_produto, descricao_produto **FROM** produto;

COMANDO SELECT - Sintaxe

B. Seleccionando todas as colunas da tabela

- `SELECT *FROM <nome_da_tabela>;`
- O **(asterisco)* indica que todas as colunas serão selecionadas.
- *Ex.: Selecionar todas as colunas da tabela cliente*
- **`SELECT *FROM cliente;`**

Comando SELECT + CLAUSULA WHERE

C. SELECIONANDO APENAS ALGUMAS LINHAS DA TABELA

- A clausula WHERE em um comando SELECT especifica quais linhas queremos obter, baseada em condições de seleção.
- Sintaxe básica.
- SELECT <nome(s) da(s) coluna(s)>
- FROM <nome da tabela>
- WHERE <condições de seleção>;

Comparações na Cláusula – WHERE

- **SELECIONANDO APENAS ALGUMAS LINHAS DA TABELA**
- WHERE <nome da coluna> <operador> <valor>;
 - Operadores de Comparação

| OPERADOR | VALOR |
|----------|-----------------------|
| = | Igual |
| <> Ou != | Diferente |
| < | Menor que |
| > | Maior que |
| >= | Maior ou igual do que |
| <= | Menor ou igual do que |
| !> | Não maior |
| <! | Não menor |

Comparações na Cláusula - WHERE

- WHERE <nome da coluna> <operador> <valor>;
- Exe1.:
 - Listar o número do pedido, o código do produto e a quantidade dos itens de pedido com a quantidade igual a 35 da tabela item do pedido.
 - SELECT num_pedido, codigo_produto, quantidade
 - FROM item_de_pedido
 - WHERE quantidade = 35;

Comparações na Cláusula - WHERE

Exe2.: Litar todos os clientes que moram em Niterói

```
SELECT nome_cliente  
FROM cliente  
WHERE cidade = 'niteroi';
```

Obs: as aspas simples sempre tem que ser colocadas para trazer os resultados de caracteres; esquecer de colocas pode não trazer os resultados ou não trazer nada.

CLÁUSULA - WHERE

D. Operadores Lógicos

| OPERADOR | VALOR |
|----------|-------------|
| AND | E – Lógico |
| OR | OU – lógico |
| NOT | negação |

Exe.: Listar os produtos que tenham unidade igual a 'M' e valor unitário igual a R\$ 1,05 da tabela produto.

```
SELECT descricao_produto  
FROM produto  
WHERE unidade = 'M' AND valor_do_produto = 1.05;
```

Clausula - WHERE

- ***Operadores Lógicos***
- Exe3.: listar os clientes e seus respectivos endereços, que moram em 'SÃO PAULO' ou estejam na faixa de CEP entre 30077000 e 30079000. classificado por nome de ordem.
- **SELECT** nome_cliente, endereco
- **FROM** cliente
- **WHERE** (cep >= 30077000 AND cep <= 30079000) OR cidade = 'São Paulo';

Clausula - WHERE

- ***Operadores Lógicos***

- Exe4.: listar os pedidos que não tenham prazo de entrega igual a 15 dias

- **SELECT** codigo_do_pedido

- **FROM** pedido

- **WHERE NOT** (prazo_entrega = 15);

- Ou poderia utilizar o operador de comparação(diferente <> ou !=)

- **SELECT** num_pedido

- **FROM** pedido

- **WHERE** (prazo_entrega != 15);

CLÁUSULA - WHERE

D. Ordenando os dados selecionados

Sintaxe básica:

SELECT <nome da(s) coluna(s)>

FROM <tabela>

WHERE <condição(ões)>

ORDER BY <nome da(s) colunas(s)> | **ASC**
DESC

- Ou
- **ORDER BY** <número da coluna>
- **Obs.:** quando a palavras **ASC** OU **DESC** forem omitidas o valor da consulta por padrão será **ASC**.

CLÁUSULA - WHERE

- ***Ordenando os dados selecionados***
- Exe5.: listar em ordem alfabética todos os seus vendedores e seus respectivos salários
- **SELECT** nome_vendedor, salario_fixo
- **FROM** vendedor
- **ORDER BY** nome_vendedor;

CLÁUSULA - WHERE

- ***Ordenando os dados selecionados***

- Exe6.: listar os nomes, cidades e estados de todos os clientes, ordenados por estado e estado de forma descendente
- **SELECT** nome, cidade, uf
- **FROM** cliente
- **ORDER BY** UF **DESC**, cidade **DESC**;
- Exe7.: Mostrar a descrição e o valor unitário de todos os produtos que tenham a unidade 'KG', em ordem de valor unitário ascendente.
- **SELECT** descricao, valor_produto
- **FROM** produto
- **WHERE** unidade = 'KG'
- **ORDER BY** valor_produto **ASC**;

CLÁUSULA - WHERE

E. Manipulando dados numéricos

Operadores aritméticos podem ser usados sobre qualquer coluna numérica, incluindo colunas de tipo de dado, int, dec, float e real.

Os operadores aritméticos são:

| SÍMBOLO | OPERAÇÃO | TIPO DE DADO |
|---------|---------------|------------------------|
| + | Adição | Int, dec, float e real |
| - | Subtração | Int, dec, float e real |
| / | Divisão | Int, dec, float e real |
| * | Multiplicação | Int, dec, float e real |
| % | Módulo | int |

Clausula - WHERE

- **Manipulando dados numéricos**
- **Operado adição (+)**
- *Exe8.: selecionar todos os vendedores com salário fixo e somar R\$ 100,00 de gratificação para cada um deles.*
- **SELECT** nome_vendedor, salario_fixo = (salario_fixo + 100)
- **FROM** vendedor;

Clausula - WHERE

- **Manipulando dados numéricos**
- **Operado subtração (-)**
- *Exe9.: selecionar todos os vendedores com salário fixo e descontar R\$ 100,00 por faltas no trabalho.*
- **SELECT** nome_vendedor, salario_fixo = (salario_fixo - 100)
- **FROM** vendedor;

Clausula – WHERE com LIKE

- Comando **LIKE** é utilizado quando precisamos buscar um determinado texto dentro de um campo com valores textuais. Seria uma espécie de “filtro”.
- Sintaxe: **SELECT** colunas **FROM** tabela **WHERE** campo **LIKE** ‘valor’
- Nessa Instrução, ‘valor’ pode ser informado de várias formas:
 - **valor**: serão retornados todos os registros que contenham **EXATAMENTE** o texto ‘valor’;
 - **%valor%**: serão retornados os registros que contenham o ‘valor’ informado, podendo aparecer em qualquer parte do campo buscado;
 - **%valor**: serão retornados os registros que terminem com o ‘valor’ pesquisado, independente de qual texto esteja começando;
 - **valor%**: serão retornados os registros que comecem com o ‘valor’ pesquisado, independente de qual texto esteja terminando;

Clausula – WHERE com BETWEEN

- Comando **BETWEEN** é utilizado quando precisamos recuperar registros na tabela cujo valor de um campo encontra-se em um intervalo especificado. Muito usado para filtrar dados por intervalo de datas.
- Sintaxe: **SELECT** colunas **FROM** tabela **WHERE** campo **BETWEEN** inicio_intervalo **AND** fim_intervalo
- Exemplos:
 - **SELECT** * **FROM** Pessoa **WHERE** Nascimento **BETWEEN** '01-01-1981' **AND** '31-12-1990'
 - **SELECT** * **FROM** Produto **WHERE** preco **BETWEEN** 20 **AND** 40

Comando - UPDATE

- **Atualizando um Registro – UPDATE**
- A atualização de dados em linhas existentes na tabela permite que:
- Especifique-se uma determinada coluna e altere-se seu valor.
- Seja indicada uma linha específica ou uma condição de identificação de linhas para que sejam alterados valores de determinadas colunas.
- Sintaxe:
- **UPDATE** <nome da tabela>
 - **SET** <nome da(s) coluna(s)> = valor
 - **WHERE** <condição>;

Comando - UPDATE

- **Atualizando um Registro – UPDATE**

Ex1.: Alterar o valor unitário do produto 'parafuso' de R\$ 1.25 para R\$ 1.62.

- **UPDATE** produto
 - **SET** val_unit = 1.62
 - **WHERE** descricao = 'Parafuso';
- **SET** – informa o novo valor ao sistema
- **A clausula WHERE** informa a linha que será alterada.

Comando - DELETE

- **Apagando Registros da Tabela**
- Sintaxe:
- **DELETE FROM** <nome da tabela>
 - **WHERE** <condicao>;
 - Exe.: apagar todos os clientes que moram em 'São Paulo'.
 - **DELETE FROM** cliente
 - **WHERE** cidade = 'São Paulo';
- **Obs.: se a cláusula WHERE for omitida todos os registros da tabela serão apagados**