



Manipulando dados com a linguagem SQL

Adriel Sales

O QUE JÁ SABEMOS...



- O que é um Banco de Dados.
- Projeto de Banco de Dados
- Modelo de Dados Relacional
- Diagrama Entidade Relacionamento
- SGBDs
- Normalização
- Etc.



- **Structured Query Language, ou Linguagem de Consulta Estruturada ou SQL;**
- Trata-se de uma linguagem específica para a manipulação de tabelas de dados;
- A linguagem padrão universal para manipular bancos de dados relacionais através dos SGBDs.

GRUPOS DE COMANDOS SQL



- Os comandos do SQL **são classificados em três grupos**, de acordo com suas principais funções:
- **DML – Data Manipulation Language**
- **DDL – Data Definition Language**
- **DCL – Data Control Language**



- (Linguagem de Manipulação de Dados);
- É o subconjunto mais utilizado da linguagem **SQL**, pois é através da DML que operamos sobre os dados dos bancos de dados com instruções de inserção, atualização, exclusão e consulta de informações. Comandos como INSERIR, DELETAR, ATUALIZAR, SELECIONAR E ETC.



- **(Linguagem de Definição de Dados)** é o subconjunto da **SQL** utilizado para gerenciar a estrutura do banco de dados. Com a DDL podemos criar, alterar e remover objetos (tabelas) no banco de dados.



- **(Linguagem de Controle de Dados)** é o subconjunto da **SQL** utilizado para controlar o acesso aos dados, basicamente com dois comandos que permite ou bloqueia o acesso de usuários a dados;

SQL vs MYSQL



- Só para constar, o MySQL não é uma extensão do SQL.
- O MySQL é um Sistema de Gerenciamento de Banco de Dados.
- O SQL é a linguagem para manipulação dos dados no SGBD.

SQL vs MYSQL



- Para utilizar as características e o funcionamento do SQL é preciso se servir de um Sistema de Gerenciamento de Bancos de Dados (SGBD), isto é, de um ambiente no qual possamos utilizar os comandos desta linguagem para manipular dados.

SQL – REGRAS



- Todas as palavras-chave das instruções **SQL serão escritas em maiúsculo;**
- Sempre no final de cada instrução, deve **ser terminado com um ponto-e- virgula (;)**

INSTALAR MYSQL



- Existem alternativas para conseguir o MySQL em seu computador.
 - Baixar o MySQL no seu site e instalá-lo;
 - (ou) Instalar pacotes que venham com o MySQL incluso, caso do **XAMPP** e **WAMP**;
 - MySQL Workbench;

PRIMEIRO ENCONTRO



- Abrir o Prompt Comando do Windows.
- Atalho: Win + R
- Executar: **cmd**
- **Go! Go! Go!**

PRIMEIROS PASSOS



- Acessando o Banco de Dados



PRIMEIRO ENCONTRO



- Acessar o diretório c:/xampp/mysql/bin
- Ou c:/wamp/mysql/bin
pelo prompt

- Usar o comando:

cd xampp/mysql/bin

cd wamp/mysql/bin

CONEXÃO COM MYSQL



- Precisamos utilizar um comando para acessar o prompt do MySQL.
- Ao instalarmos o MySQL é obrigatório criar um usuário e senha para o acesso dos Banco de Dados. **Por padrão**, o usuário é **root** e a senha é vazia.
- Estas informações (usuário e senha) são necessários para este passo.

CONEXÃO COM MYSQL



- O comando para acessarmos o MySQL é:
`mysql -u usuario -p senha`
- **Em nosso caso ficando:**
`mysql -u root -p`

Conexão realizada!



Cmder

```
C:\
λ cd C:\xampp\mysql\bin

C:\xampp\mysql\bin
λ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.16-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> |
```

```
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 5.1.41 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

CRIAR UM BANCO DE DADOS



- Para criar de um banco de dados o comando é simples.

```
mysql> CREATE DATABASE meu-banco;
```

CREATE DATABASE seguido do nome desejado de banco de dados.

DELETAR UM BANCO DE DADOS



- Para excluir um banco de dados, usa-se o comando `DROP DATABASE`, seguido do nome do banco de dados que deseja deletar.

```
mysql> DROP DATABASE meu-banco;
```

MOSTRAR BANCO DE DADOS



- Podemos verificar rapidamente a existência do BD recém-criado, bem como a de todos os outros criados anteriormente, utilizando a instrução `SHOW DATABASES` (mostrar bancos de dados);

```
mysql> SHOW DATABASES;
```

CRIAR BANCO DE DADOS



- SE NÃO EXISTIR...
- Para verificar se existe um determinado banco de dados antes da criação de um novo. O comando utilizado é:

```
mysql> CREATE DATABASE IF NOT EXISTS banco;
```

CUIDADO AO DELETAR



- É preciso ressaltar que, **ao apagar um banco de dados, todas as suas tabelas e os dados nelas contidos também serão apagados e, portanto, perdidos de maneira irreversível.**
- **ENTÃO, CUIDADO!** 😊

USAR UM BANCO DE DADOS



- Como vimos, podemos criar vários bancos de dados, porém, podemos manipular apenas um por vez. Assim, antes de começar, é preciso selecionar qual será o banco de dados que queremos alterar.
- Isso é feito utilizando o comando **USE** (“usar” em inglês), seguido pelo nome do banco de dados em questão.

```
mysql> USE meu-banco;
```

IMPORTAR/EXPORTAR BD



■ Comandos do mysql

```
C:\xampp\mysql\bin
λ ls
aria_chk.exe*          myisamchk.exe*        mysql_secure_installation.pl*  mysqldumpslow.pl*
aria_dump_log.exe*     myisamlog.exe*        mysql_tzinfo_to_sql.exe*      mysqlhotcopy.pl*
aria_ftdump.exe*      myisampack.exe*       mysql_upgrade.exe*            mysqlimport.exe*
aria_pack.exe*        mysql.exe*             mysql_upgrade_service.exe*    mysqlshow.exe*
aria_read_log.exe*    mysql_client_test.exe* mysql_upgrade_wizard.exe*     mysqlslap.exe*
debug/                mysql_client_test_embedded.exe*  mysqladmin.exe*              mysqltest.exe*
echo.exe*              mysql_config.pl*        mysqlbinlog.exe*              mysqltest_embedded.exe*
innochecksum.exe*     mysql_convert_table_format.pl*  mysqlcheck.exe*               perror.exe*
my.ini                 mysql_embedded.exe*      mysqld.exe*                   replace.exe*
my_print_defaults.exe* mysql_install_db.exe*     mysqld_multi.pl*
myisam_ftdump.exe*    mysql_plugin.exe*        mysqldump.exe*
```


EXPORTANDO BD MYSQL



- Usando a CLI (linha de comando):
 - `mysqldump -u usuario -p banco > arquivo.sql`
- Onde:
 - **usuario:** nome do usuário no MySQL que tem acesso ao banco.
 - **banco:** nome do banco de dados que será exportado.
 - **arquivo.sql:** nome do arquivo para o qual os dados serão gravados.
 - Em seguida será solicitada a senha do usuário Mysql

```
C:\xampp\mysql\bin
λ mysqldump -u root -p cursos > C:\xampp\htdocs\senac\programadorWeb2017\mysql\bkp-cursos-set-2017.sql
Enter password:
```

IMPORTANDO BD MYSQL



- Usando a CLI (linha de comando):

- `mysql -u usuario -p banco_de_dados < arquivo.sql`

- Onde:

- **usuario:** nome do usuário no MySQL que tem acesso ao banco de dados.
 - **banco_de_dados:** nome do banco de dados que receberá os registros.
 - **arquivo.sql:** nome do arquivo que contém os dados que serão importados.
 - Em seguida será solicitada a senha do usuário Mysql

IMPORTANDO BD MYSQL



- É preciso criar o banco antes de importar!

```
C:\Users\adriel
λ cd \xampp\mysql\bin\

C:\xampp\mysql\bin
λ mysql -u root -p cursos < C:\xampp\htdocs\senac\programadorWeb2017\mysql\bkp-cursos.sql
Enter password:

C:\xampp\mysql\bin
λ mysql -u root -p cursos < C:\xampp\htdocs\senac\programadorWeb2017\mysql\bkp-cursos.sql
Enter password:
ERROR 1049 (42000): Unknown database 'cursos'
```

IMPORTANDO BD MYSQL



1º - Criando o BD cursos

```
MariaDB [(none)]> create database cursos;  
Query OK, 1 row affected (0.07 sec)
```

Logado ao Mysql

```
MariaDB [(none)]> quit;  
Bye
```

Desloga do Mysql

2º - Executando o comando de importação no diretório **/bin** do mysql

```
C:\xampp\mysql\bin
```

```
λ mysql -u root -p cursos < C:\xampp\htdocs\senac\programadorWeb2017\mysql\bkp-cursos.sql  
Enter password:
```

O Mysql Executa as instruções e importa os dados do backup para a base de dados cursos.

TABELAS



TABELAS



1 - Modelo Conceitual



3 - Modelo Físico

id	nome	nota

2 - Modelo Lógico

```
CREATE TABLE alunos(  
  id int NOT NULL auto_increment,  
  nome varchar(55) NOT NULL,  
  nota float default 0,  
  primary key(id)  
);
```

TABELAS



```
INSERT INTO alunos (nome, nota)  
VALUES ("José", 8.5);
```

```
INSERT INTO alunos (nome, nota)  
VALUES ("Maria", 9.8);
```

```
INSERT INTO alunos (nome, nota)  
VALUES ("João", 5.5);
```

```
INSERT INTO alunos (nome, nota)  
VALUES ("Marlene", 7.0);
```

id	nome	nota
1	José	8.5
2	Maria	9.8
3	João	5.5
4	Marlene	7

CRIAR UMA TABELA



- A regra base do comando para criar uma tabela no banco de dados é o comando criar tabela, seguido do nome da tabela.
- Também é necessário informar os campos da tabela, seu tipo e seu tamanho.

CRIAR UMA TABELA

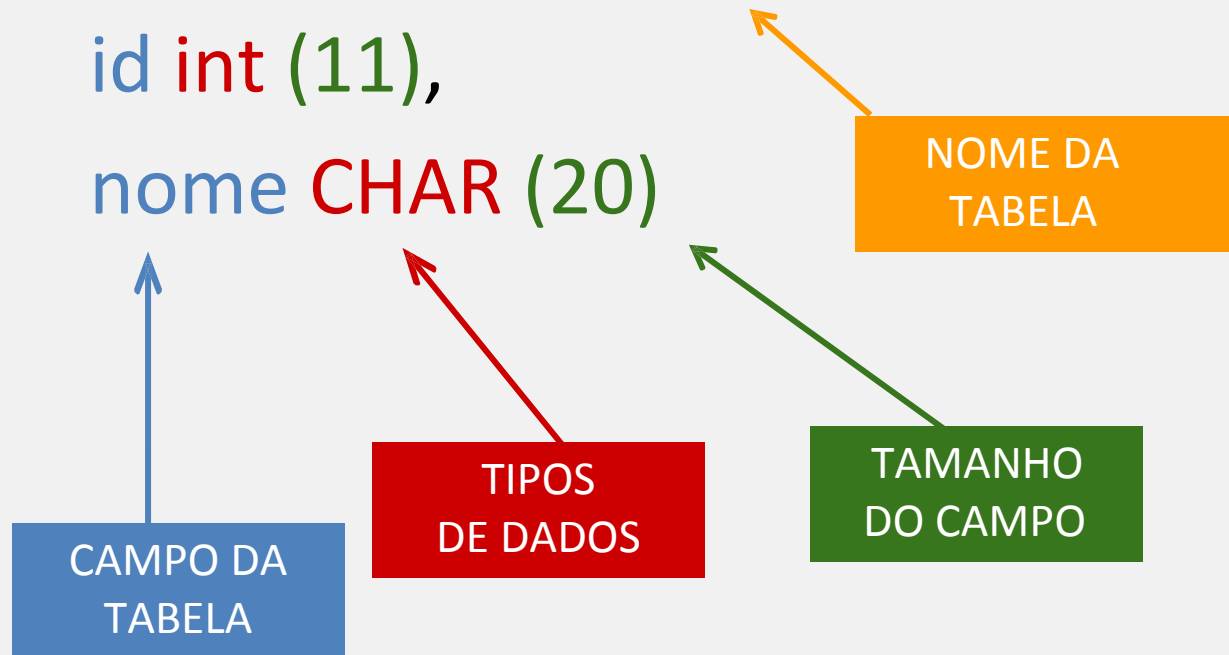


```
create table disciplina (  
    id int not null auto_increment,  
    nome varchar(55) not null,  
    descricao text,  
    primary key(id)  
);
```

CRIAR UMA TABELA



```
mysql> CREATE TABLE cadastro (  
    id int (11),  
    nome CHAR (20)  
);
```



TIPOS DE CAMPOS



- Existem vários tipos possíveis de dados no SQL, embora os mais comuns sejam:
- **INT** ou **INTEGER**: Para inteiros de tamanho normais.
- **TIMESTAMP**: Para data e hora, que pode ser atribuído automaticamente;
- **CHAR** e **VARCHAR**: Para textos de até 255 caracteres;
- **TEXT** ou **LONGTEXT**: Para textos longos;

MOSTRAR TABELA



- Para exibir a lista de tabelas do banco de dados que está usando atualmente, basta utilizar o comando:

```
mysql> SHOW TABLES;
```

MOSTRAR ESTRUTURA DA TABELA



- Podemos também analisar a estrutura de uma tabela de maneira aprofundada usando o comando DESCRIBE (“descrever”, em inglês), seguido pelo nome da tabela.

```
mysql> DESCRIBE minha-tabela;
```

Ou

```
mysql> DESC minha-tabela;
```

ALTERAR TABELA



- Para alterar uma tabela, basta utilizar **ALTER TABLE**, o nome da tabela o qual quer alterar e qual operação de alteração quer fazer.
- **Operações:** Adicionar novo campo, renomear nome da tabela e etc.

As operações estão em cores destacadas.

RENOMEAR TABELA, ADICIONAR E MODIFICAR CAMPOS



```
mysql> ALTER TABLE pessoas RENAME TO cadastros;
```

```
mysql> ALTER TABLE pessoas ADD idade INT(3);
```

```
mysql> ALTER TABLE pessoas MODIFY idade INT(5);
```

DELETAR E ORDENAR A ADIÇÃO DE NOVOS CAMPOS



//deleta o campo nome da tabela pessoas
mysql> **ALTER TABLE** pessoas **DROP** nome;

//adiciona após o campo indicado
mysql> **ALTER TABLE** pessoas **ADD** idade **INT**(3)
AFTER campo;

//adiciona o campo no início da tabela
mysql> **ALTER TABLE** pessoas **ADD** idade **INT**(3)
FIRST;

OPÇÕES DOS CAMPOS



- Alguns campos podem ter particularidades. Por exemplo, ser chave primária, não pode ser vazia e etc. Veremos algumas opções.



NOT NULL



- O campo com a opção **NOT NULL**, significa que o campo não poderá ser nulo.
- Para utilizar isso, basta na criação do campo adicionar NOT NULL na frente dele.

```
mysql> CREATE TABLE pessoas (  
Nome VARCHAR(255) NOT NULL  
);
```

PRIMARY KEY



- Para definirmos que um campo é chave primária, utilizamos a opção **PRIMARY KEY**, após o nome do campo ou no final da tabela.

```
mysql> CREATE TABLE pessoas (  
    Id INT(5) PRIMARY KEY  
);
```

```
mysql> CREATE TABLE pessoas (  
    id INT(5) NOT NULL,  
    PRIMARY KEY(id)  
);
```

AUTO INCREMENT



- Auto incremento, significa que a cada registro de uma tabela, o valor será incrementado (aumentado) automaticamente.
- Geralmente, utilizamos para campos ID, CODIGO ou CHAVES PRIMARIAS;

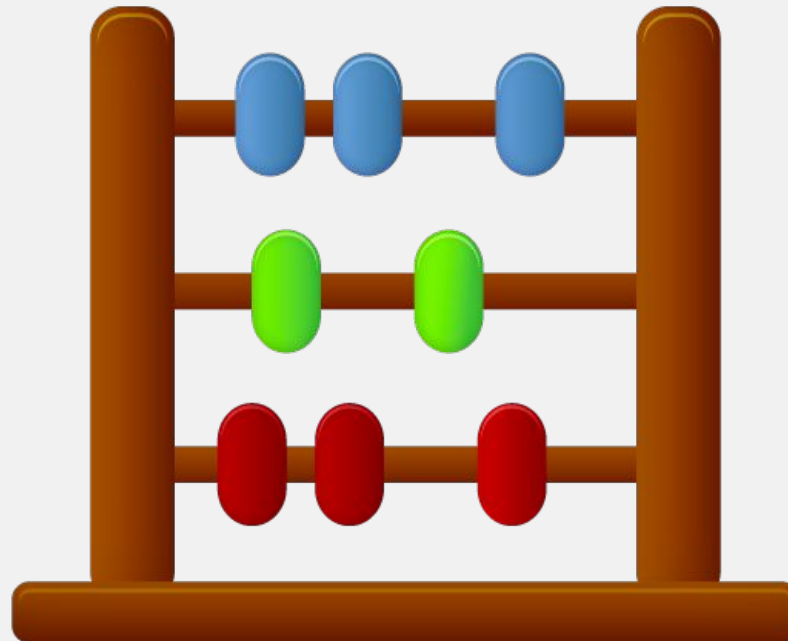
```
mysql> CREATE TABLE animals (  
id INT(5) NOT NULL PRIMARY KEY AUTO_INCREMENT,  
name VARCHAR(50) NOT NULL  
);
```

EXERCÍCIOS



- CRIAR BANCO DE DADOS cadastro;
 - CRIAR TABELA pessoas: id, nome, idade;
 - CRIAR TABELA times: id, time;
- OBS.: TODO ID É **PRIMARY KEY NOT NULL AUTO_INCREMENT**

MANIPULANDO DADOS NA TABELA



INSERIR VALORES NA TABELA



- O comando de INSERIR é um dos mais utilizados. Para inserir valores em uma determinada tabela, basta seguir a regra:

```
mysql> INSERT INTO nome_da_tabela  
VALUES ('valor1', 'valor2', ...);
```

INSERIR VALORES NA TABELA



```
mysql> INSERT tabela  
(campo1, campo2, campo3, ...)  
VALUES  
("valor1", "valor2", "valor3");
```

```
INSERT INTO alunos (nome, nota)  
VALUES ("José", 8.5);
```


ATUALIZAR VALORES NA TABELA



- A instrução **UPDATE** é usada para modificar os registros existentes em uma tabela.

```
UPDATE nome_da_tabela  
SET coluna1 = valor1, coluna2 = valor2, ...  
WHERE condition;
```

```
UPDATE nome_da_tabela  
SET coluna1 = valor1, coluna2 = valor2, ...  
WHERE condition;
```

SELECIONAR VALORES DA TABELA



- É possível selecionar valores da tabela, utilizando o comando **SELECT** do SQL. O comando **SELECT** é, basicamente, a ferramenta principal para consultar informações de um banco de dados, por isso, é comumente chamado de query.

```
mysql> SELECT dados_desejados  
      FROM nome_tabela;
```

SELECT COM ASTERISCO (*)



- **Significa tudo**, ou seja, todos os dados.
- Pode ser combinado com um ou mais caracteres para especificar conjuntos de dados com algo em comum.
- Por exemplo, em geral, se digitarmos o critério A* significa que queremos ver todos os registros cujo conteúdo começa com a letra A;

EXERCÍCIOS



- Crie um banco de dados chamado **cinema**.
- Cria a tabela **filmes**:
- Insira 5 registro;
- Mostre apenas os campos **titulo**,
duração e **ano** dos filmes cadastrados;

filmes
titulo: VARCHAR(255)
categoria: VARCHAR(50)
duracao: INT(5)
diretor: VARCHAR(100)
sinopse: TEXT
ano: INT(4)

CLAUSULA WHERE



- Usando a cláusula **WHERE**, podemos especificar um critério de seleção para selecionar os registros necessários de uma tabela.
- Funciona como uma condição em qualquer linguagem de programação.
- Esta cláusula é usada para comparar determinado valor com o valor do campo disponível na tabela MySQL.

CLAUSULA WHERE



SELECIONE nome_do_campo DA tabela_y
ONDE nome_do_campo seja igual ao valor

SELECT idade FROM aluno
WHERE idade = 18;

CLAUSULA WHERE



Operador	Descrição	Exemplo
=	Verifica se os valores dos dois operadores são iguais ou não, se sim, então condição torna-se verdade.	(A = B) não é verdade.
!=	Verifica se os valores de dois operandos são iguais ou não, se os valores não são iguais então a condição torna-se verdade.	(A! = B) é verdadeiro.
>	Verifica se o valor do operando esquerdo é maior que o valor do operando da direita, se sim, então a condição se torna verdadeira.	(A > B) não é verdade.
<	Verifica se o valor do operando esquerdo é menor que o valor do operando da direita, se sim, então condição torna-se verdade.	(A < B) é verdadeiro.
>=	Verifica se o valor do operando esquerdo é maior ou igual ao valor do operando da direita, se sim, então a condição se torna verdadeira.	(A > = B) não é verdade.
<=	Verifica se o valor do operando esquerdo é menor ou igual ao valor do operando da direita, se sim, então condição torna-se verdade.	(A <= B) é verdadeiro.

WHERE: AND, OR e NOT



- Os operadores AND e OR são usados para filtrar registros com base em mais de uma condição.
- **O operador AND** **exibe registros** se todas as condições separados por e é verdadeira.
- **O operador OR** **exibe registros**, se qualquer uma das condições separados por OR é TRUE.
- **O operador NOT** **exibe registros** se a condição não é verdadeira.

WHERE: AND, OR e NOT



```
SELECT * FROM usuarios  
WHERE pais='Brasil' AND cidade='João Pessoa';
```

```
SELECT * FROM Customers  
WHERE cidade='João Pessoa' OR cidade='Recife';
```

```
SELECT * FROM Customers  
WHERE NOT pais='Brasil';
```

Podemos usar a função **IN** para múltiplos valores:

```
SELECT * FROM usuarios WHERE pais IN ('Brasil', 'França');  
SELECT * FROM usuarios WHERE pais NOT IN ('Brasil', 'França');
```

LIMITAR



- Pode-se limitar a quantidades de registros.
- Se não queremos uma lista extensa, apenas precisamos das 5 primeiras linhas, coloca-se o LIMIT de 5.

```
mysql> SELECT * FROM pessoas LIMIT 5;
```

ORDER BY - Ordenar



- Quando for necessário ordenar a lista de registros em ordem crescente (ASC) ou decrescente (DESC).
- Para utilizar a ordenação, precisa escolher por qual campo será feita a ordenação.

ORDER BY - Ordenar



ORDEM DECRESCENTE

```
mysql> SELECT *  
FROM pessoas ORDER BY idade DESC.
```

ORDEM CRESCENTE

```
mysql> SELECT *  
FROM pessoas ORDER BY idade ASC.
```

LIKE



- O **LIKE** é usado para fazer buscas por partes de conteúdos. Por exemplos, precisamos capturar todas as pessoas com que tem Ana no nome, utilizamos do seguinte código:

```
mysql> SELECT * FROM pessoas  
WHERE nome LIKE '%ana%' LIMIT 2;
```

- O LIKE é utilizado da seguinte forma:
LIKE %conteudo%

LIKE - Outras formas de uso



Operador	Descrição
LIKE “_E%”	A letra E está na segunda posição.
LIKE “%O”	Termina com a letra O.
LIKE “A%E%O”	Começa com a letra A, termina com a letra O e possui a letra E no meio.
NOT LIKE “%@%”	Retorna aqueles que NÃO contêm o caractere @.

FUNÇÕES DE AGREGAÇÃO



MIN () e MAX()



- As funções MIN () e MAX () retornam o menor ou o maior valor da coluna selecionada:

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```


COUNT(), AVG () e SUM ()



- A função **COUNT()** retorna o número de linhas que corresponde a um campo específico da tabela.
- A função **AVG()** retorna a média de uma coluna numérica.
- A função de **SUM()** retorna a soma total de uma coluna numérica.

COUNT(), AVG () e SUM ()



```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

GROUP BY - Agrupar



- A instrução **GROUP BY** é muitas vezes utilizado com **funções de agregação** (COUNT, MAX, MIN, SUM, AVG) ao grupo de resultados definido por uma ou mais colunas.

```
SELECT COUNT(id_usuario), pais  
FROM usuarios  
GROUP BY pais;
```

ALIASES



- São utilizados para dar a uma tabela ou a uma coluna da tabela, um nome temporário.
- São muitas vezes utilizados para tornar os nomes das colunas mais legíveis.
- Um alias só existe no momento da consulta.

ALIASES



```
SELECT column_name AS alias_name  
FROM table_name;
```

```
SELECT column_name(s)  
FROM table_name AS alias_name;
```

BETWEEN (ENTRE)



- O operador **BETWEEN** seleciona valores dentro de uma determinada faixa.
- Os valores podem ser números, texto ou datas.

```
SELECT column_name(s)  
FROM table_name  
WHERE column_name BETWEEN value1 AND value2;
```

JUNTANDO TABELAS



UNIR TABELA COM JUNÇÃO



- Pode-se unir duas ou mais tabelas.
- Para isso utiliza-se o junções ou JOINS.
- **Exemplo de Junção:**

```
select disciplina.nome, aluno.nome, aluno.nota  
from aluno, disciplina  
where aluno.disciplina_id = disciplina.id  
and disciplina.nome = "Matemática";
```


UNIR TABELA COM INNER JOIN



- **Exemplo com INNER JOIN.**

```
mysql> SELECT *  
      FROM pessoas  
      INNER JOIN times;
```

UNIR TABELA COM INNER JOIN



```
mysql> SELECT pessoas.nome, times.time  
FROM pessoas  
INNER JOIN times  
ON pessoas.time_id = times.id LIMIT 5;
```

UNIR TABELA COM INNER JOIN



```
SELECT tabela1.campo1, tabela1.campo2, tabela2.campo1  
FROM tabela1  
INNER JOIN tabela2  
ON tabela1.campo 1 = tabela2.campo1;
```

Referências



Mysql.com

<https://www.mysql.com/>

LACERDA, Ivan Max Freire de, OLIVEIRA, Ana Liz Souto.

Programador Web: um guia para programação e manipulação de banco de dados.

Rio de Janeiro: Senac Nacional, 2013.

Slide do Professor Rangel:

<https://pt.slideshare.net/Ranginaldo/bd-parte-0148slides>

W3schools:

<https://www.w3schools.com/sql/default.asp>

SlideShare: Mer - Modelo Entidade Relacionamento:

<https://pt.slideshare.net/professor-rade/mer-23596358>

Apontamentos da Aula:

<https://github.com/adrielsales/senac/wiki/Aulas-Senac>

