**MySQL**

**Banco de dados MySQL**

**SGBD** – **S**istema **G**erenciador de **B**anco de **D**ados.

- Quando é criada uma aplicação utilizando alguma linguagem de programação (PHP, Java, C#....), geralmente isso não é o suficiente para resolver problemas reais;

- Geralmente é necessário o armazenamento persistente de dados, para que esses dados possam ser utilizados ou compartilhados em outro momento.

- O **MySQL** é um **sistema gerenciador de banco de dados *RELACIONAL***, gratuito e que utiliza a linguagem SQL;

**SQL: S**tructured **Q**uery **L**anguage ou Linguagem de Consulta Estruturada;

- Para: **definir**, **manipular**, **controlar**, **transacionar** e **recuperar dados**.

- Funciona como uma **interface entre** a **aplicação e** os **dados**;

Principal objetivo: controlar o acesso, a manipulação e a organização dos dados persistidos em um servidor de dados.

O SGBD fornece uma API para as aplicações, para que assim elas possam ter acesso aos dados persistidos no servidor;

Sendo toda essa **comunicação** realizada entre a aplicação e o SGBD **feita através da linguagem SQL**.

**Banco de Dados Relacionais** – quando **há tabelas e dentro dessas tabelas temos registros** e esses registros estão relacionados entre essas tabelas.

Um exemplo seria, estamos falando de uma empresa de gestão acadêmica, temos duas tabelas (relação de cursos disponíveis e relação dos alunos):

|  |
| --- |
| TABELA\_CURSOS |
| id\_curso |
| nome\_curso |
| descricao\_curso |

|  |
| --- |
| TABELA\_ALUNOS |
| id\_aluno |
| nome\_aluno |
|  |

Esses registros podem ser relacionados, de modo a criar informações mais organizadas, com menor grau de redundância possível:

|  |
| --- |
| TABELA\_ALUNOS\_CURSOS |
| id\_aluno\_curso |
| id\_aluno |
| id\_curso |

**SQL** é a **linguagem padrão** nos principais banco de dados relacionais existentes hoje no mercado, tais como:

MySQL;

PostgreSQL;

SQLServer;

ORACLE.

Essa linguagem pode ser **dividida em 5 subcategorias** e intruções com objetivos específicos:

***DDL*** – Data Definition Language – Linguagem de definição de dados;

Nos possibilita implementar a mo delagem de dados (criação, alteração e remoção das estruturas de dados);

***DML*** – Data Manipulation Language - Linguagem de manipulação de dados;

Nos permite fazer a inclusão, alteração e remoção dos registros dentro das estruturas de dados;

**DCL** – Data Control Language - Linguagem de controle de dados;

Gerencia acesso por parte de usuários externos ao SGBD;

**DTL** – Data Transaction Language - Linguagem de transação de dados;

Nos permite efetivar ou cancelar as transações, junto ao SGBD;

***DQL*** – Data Query Language - Linguagem de consulta de dados;

Nos permite recuperar dados através do estabelecimento de cláusulas, operações lógicas, operações relacionais ou de funções de agregação.

**Manipulação do MySQL com PHPMyAdmin**

**PHPMyAdmin –** Aplicação Web escrita em PHP que serve para acessar e administrar o banco de dados MySQL;

Interface p/SGBD, que será acessada através do navegador;

E é no navegador que serão digitadas as instruções SQL.

Para **acessar o PHPMyAdmin** é fundamental:

- **Subir o serviço do apache** (pois o phpmyadmin é uma aplicação web e depende de um servidor HTTP para funcionar);

- **Subir o serviço do mysql** (sem isso não existe um SGBD a ser acessado).

Interface gráfica do usuário, Texto, Tabela

Descrição gerada automaticamente

No browser: ***localhost/phpmyadmin/***. Para ter acesso a detalhes sobre os serviços php e mysql.

**Criando e excluindo banco de dados**

**Banco de dados** – coleções organizadas de dados que se relacionam de algum modo;

A ideia consiste em agrupar registros de um domínio específico;

Não significando que existe um jeito certo e único de criar um banco de dados, tudo depende do nível de abstração do assunto.

Quando estamos desenvolvendo uma aplicação, geralmente criamos um banco de dados específico para a aplicação;

Mas, essa mesma aplicação, pode ter acesso ou até controle sobre diversos outros bancos de dados ao mesmo tempo;

É necessário tomar alguns cuidados em termo de teoria do assunto.

CREATE DATABASE db\_curso\_web;

New>nome\_do\_banco>criar

Criando o ***BANCO DE DADOS***.

[DROP](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/drop-database.html) [DATABASE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/drop-database.html) db\_curso\_web;

Operações>Apagar a Base de Dados (DROP)>Ok

excluindo o ***BANCO DE DADOS***.

**Tabelas e tipos de dados**

**Tabelas** são **semelhantes a planilhas** e podem ser entendidas como **unidade de armazenamento**.

São constituídas por um número finito de colunas;

Possui um **número indefinido de linhas**, pois **não sabemos a quantidade de registros** (linhas) que serão **armazenados dentro da tabela**.

**Cada coluna** de uma tabela **é responsável pelo armazenamento de um tipo de dado específico** e isso **deve ser definido no momento de criação** da tabela.

1ª coluna: valor numérico;

2ª coluna: texto de 100 caracteres (variável);

3ª coluna: texto de 50 caracteres (fixos);

4ª coluna: texto longo;

5ª coluna: data;

6ª coluna: booleano;

7ª coluna: valor numérico com fração;

Armazenar os dados de cadastro de cursos:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id\_curso | imagem\_curso | nome\_curso | resumo | data\_cadastro | ativo | investimento |
| 1 | web2.png | Web 2.0 | Curso ... | 2018/07/11 | 1 | 550.00 |

**id\_curso** – auxilia na identificação do registro dentro da tabela;

**imagem\_curso** – irá permitir a inclusão de uma URL que aponte para um recurso de imagem.

Quando se fala de imagens, sons (armazenamento de mídias) o que é armazenado não é a mídia em si, mas o nome do arquivo digital e sua respectiva extensão, para que a própria aplicação se encarregue de localizar esse recurso e entrega-lo de forma apropriada;

**nome\_curso** – texto curto cujo tamanho já está previamente alocada em disco, independentemente de ser ou não totalmente utilizado;

**resumo** – informações mais completas sobre o curso;

**data\_cadastro** – data em que o registro do curso foi cadastrado na tabela no banco de dados através da aplicação. Para fins de controle e log do registro;

**ativo** – armazena estado (true ou false), indica se o registro do curso está ou não ativo no sistema;

**investimento** – diferentemente do **id\_curso** suporta casas decimais, valores fracionados.

Campos de texto:

- text (tamanho variável que armazena uma grande quantidade de caracteres);

- varchar (tamanho variável que armazena de 0 até 255 caracteres);

- char (tamanho fixo que armazena de o até 255 caracteres).

Campos numéricos:

- int (valores numéricos inteiros, tanto positivos quanto negativos);

- float (valores numéricos fracionados, tanto positivos quanto negativos).

Campos de data e hora:

- Date (data no formato YYYY/mm/dd);

- Time (hora);

- Datetime (combinação de date e time em um mesmo campo).

Entra no banco de dados e procure “Criar tabela”;

Indique o nome da tabela;

Coloque a quantidade de colunas;

Clique em: Criar tabela.

Excluir tabela

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Clicar em remover tabela

**Criar tabela no SQL**

CREATE TABLE tb\_cursos ();

Parênteses delimitam as colunas criadas na tabela.

id\_curso int **not null,**

indica que não queremos atribuir um valor vazio a este campo (é necessário passar qualquer tipo de parâmetro no momento de inclusão do registro.

Utiliza-se a **vírgula** para separar declaração de colunas.

imagem\_curso varchar(100) not null,

variável com valor definido;

not null – precisa ser preenchido.

resumo text null,

por ser um texto longo eu não preciso indicar uma quantidade de caracteres;

texto é opcional então pode ser null;

null – indica que no processo de inserção de um registro, eu posso omitir a definição do valor dessa coluna.

Podendo ser um atributo do registro null.

ativo boolean default true,

através da palavra default, podemos atribuir uma informação que será

associada como valor dessa coluna no momento da inserção do registro como sendo a informação padrão caso ela não seja preenchida.

Null – pode ser omitido no momento da inserção;

Not null – obrigatórios no momento da inserção;

Default – definir valor padrão da coluna.

investimento float(8,2)

numérico com fração;

o primeiro número são os dígitos (**234431.15**);

e o último número é quantos números correspondem a fração (234431.**15**)

**removendo tabela com sql**

drop table tb\_cursos;

**Diferenças entre char e varchar**

**Char** – tamanho **fixo** em disco. -> pesquisas mais rápidas.

**Varchar** – tamanho **variável** em disco. -> por ser de tamanho variável ocupa apenas espaço necessário.

- É definido um **campo de 10 posições** com o tipo **char**;

- Todos os **espaços relativos** à quantidade de caracteres definidos ficam reservados em disco;

- Independente de inserir ou não texto com essa quantidade de caracteres, em função da string os espaços não utilizados ficarão **reservados com espaços em branco**.

- É definido um **campo de 10 posições** com o tipo **varchar**;

- Todos os **espaços relativos** à quantidade de caracteres definidos ficam reservados em disco;

- Diferentemente do char, tipos de dados varchar possuem a inteligência de **reservar** apenas a **quantidade de caracteres utilizadas para a string**;

- Ocupa menos espaço em banco de dados.

Conclui-se que a vantagem de um é a desvantagem do outro.

**Char – vantagem**: mais rápido para pesquisas.

**Char – desvantagem**: quando mau utilizado pode reservar espaço em disco de forma desnecessária.

**Varchar – vantagem**: por ser de tamanho variável ocupa apenas espaço necessário.

**Varchar – desvantagem**: por ser de tamanho variável possui um meta dado com uma instrução de finalização do texto, o que produz, em relação ao CHAR, maior lentidão de pesquisas.

**Editando nome de tabelas**

Pode ser preciso por:

Erro de digitação, mudança no banco de dados...

Renomeando

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente clica no arquivo da tabela



Logotipo

Descrição gerada automaticamente com confiança médiaoperações

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente



Renomea e executa

**No SQL**

RENAME TABLE <nome\_atual> TO <nome\_novo>

RENAME TABLE tb\_cursos\_teste TO tb\_cursos;

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Provavelmente é preciso que você atualize a página para aparecer a mudança.

**F5 ou** 



**Incluindo, editando e removendo colunas de tabelas**

**ALTER TABLE** – faz parte do subconjunto de instruções do SQL (DDL– Data Definition Language – Linguagem de definição de dados).

**ADD –** Permite a inclusão de uma nova coluna em uma tabela já existente;

**CHANGE –** Permite a alteração do nome de uma coluna e de suas propriedades, como por exemplo o tipo;

**DROP –** Permite a remoção de uma coluna.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente Criando uma nova colunaInterface gráfica do usuário

Descrição gerada automaticamentePara editar dados, basta expandir a coluna e clicar no item que você deseja editar, será aberta essa estrutura







Eliminando dados

Clica em eliminar e confirma (estrutura da tabela)

**Adicionando coluna - *SQL***

ALTER TABLE <nomedatabela> ADD COLUMN <nomedacoluna> <atributos (tipo, valor default, null)> <precisaounaoserpreenchido>

Ex.:

ALTER TABLE tb\_cursos ADD COLUMN carga\_horaria varchar(5) not null;

nomedatabela – cujo a coluna será adicionada;

nomedacoluna – coluna que será adicionada

**Editando coluna**

ALTER TABLE <nomedatabela> CHANGE <colunaqueseramodificada> <nomedacoluna> <tipo><oqueseramodificado>

Ex.:

ALTER TABLE tb\_cursos CHANGE carga\_horaria carga\_hora INT(5) null;

Nomedacoluna – se ela for modificada, incluir nome modificado;

Tipo – se não será modificado, coloca o original.

**Excluindo coluna**

ALTER TABLE <nometabela> DROP <coluna que será excluída>;

Ex.:

ALTER TABLE tb\_cursos DROP carga\_horaria;

**INSERT – Inserindo dados em tabela**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente



**Acessa banco dados**

**Acessa tabela**

***Clica em ‘Inserir’***

Interface gráfica do usuário

Descrição gerada automaticamente **Add informações a tabela**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**SQL**

INSERT INTO <nometabela>(<colunas>)values(<valoresatribuidosacoluna>);

Ex.:

INSERT INTO tb\_cursos(ativo, carga\_horario, data\_cadastro, id\_curso, imagem\_curso, investimento, nome\_curso, resumo)values(1, 35, '2023-08-21 15:52:20', 2, 'curso\_angular.jpg', 575.86, 'Web Completo com JS, TS e Angular', 'Aprenda a criar aplicações front-end incríveis com JavaScript, TypeScript e Angular');

Strings são passadas por aspas simples (‘’);

Valor deve ser passado na mesma ordem da coluna em que será adicionado o valor entre os parênteses;

Já a ordem original, não precisa ser seguida.

**Não há como fazer dois inserts de uma vez, por exemplo para duas tabelas (não é possível).**

**SELECT – Consultando dados**

Para fazer isso manualmente basta clicar na tabela:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**SQL**

SELECT <clolunas> FROM <tabela>

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM tb\_cursos; - caso eu queira selecionar todos os campos da tabela

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) ativo, carga\_horario, id\_cursos FROM tb\_cursos; – caso queira adicionar apenas alguns itens da tabela

Pode-se fazer isso com todas as colunas na ordem desejada;

Colunas são separadas por vírgula;

Não é recomendado usar o coringa (\* - asterisco), para evitar erros.

**Filtrando registros (WHERE)**

Como utilizar filtros para limitar as consultas, atualizações e remoções de registros;

**Filtros** – Nos permitem especificar o que queremos obter.

**WHERE** – Indica para o sistema gerenciador do banco de dados que para aquela consulta que será realizada, existem filtros que devem ser levados em consideração.

Exemplos:

SELECT id\_curso, nome\_curso FROM tb\_cursos WHERE investimento < 500.00

Só serão retornados os registros que o investimento é menor que 500.

SELECT id\_curso, nome\_curso FROM tb\_cursos WHERE investimento < 500.00 AND carga\_horaria > 30

Só serão retornados os registros que o investimento é menor que 500 e a carga horaria seja maior que 30.

**Operadores de comparação:**

**=** valor da esquerda **igual** ao valor da direita;

**<** valor da esquerda **menor** que o valor da direita;

**<=** valor da esquerda **menor ou igual** ao valor da direita;

**>** valor da esquerda **maior** que o valor da direita;

**>=** valor da esquerda **maior ou igual** ao valor da direita.

**Operadores lógicos:**

**AND** Todas as operações de comparação devem ser verdadeiras;

**OR** Pelo menos uma das operações de comparação deve ser verdadeira.

**Populando o banco de dados com registros para testes**

[**https://generatedata.com/**](https://generatedata.com/)- gera dados aleatórios (pode ser utilizado para fins de teste).

CREATE TABLE tb\_alunos (

id\_aluno int,

nome varchar(255) default NULL,

idade int default NULL,

interesse varchar(255) default NULL,

email varchar(255) default NULL,

estado varchar(255) default NULL

);

INSERT INTO `tb\_alunos`(.....................);

INSERT INTO `tb\_alunos`(.....................);

INSERT INTO `tb\_alunos`(.....................);

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Filtrando pessoas que tem interesse = Jogos.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE interesse = 'Jogos';

Tabela

Descrição gerada automaticamente com confiança baixa

Filtrando pessoas que tem a idade < 25.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE idade < 25;

Tabela

Descrição gerada automaticamente

Filtrando pessoas que tem a idade <= 25.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE idade <= 25;

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Filtrando pessoas que tem a idade > 30.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE idade > 30;

Tabela

Descrição gerada automaticamente

**SELECT – Filtros com Operadores Lógicos**

Pessoa deve ter interesse = Jogos e idade > 45; se atender apenas à uma das condições, será = false.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE interesse= 'Jogos' [AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_and) idade > 45;

Interface gráfica do usuário

Descrição gerada automaticamente

Além da pessoa ter interesse = Jogos e idade > 45, ela também precisa ser estado = ‘RN’.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE interesse = 'Jogos' [AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_and) idade > 45 [AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_and) estado = 'RN';

Interface gráfica do usuário

Descrição gerada automaticamente

**No operador *OR*, *apenas uma condição* precisa *ser verdadeira* para retornar *true*.**

Se apenas uma dessas condições for verdadeira, retorna true:

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE interesse = 'Jogos' [OR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_or) idade > 45;

Interface gráfica do usuário

Descrição gerada automaticamente

**SELECT – Filtros com o operador BETWEEN**

Filtra intervalos específicos, tanto numéricos, quanto intervalos de datas.

Retornar quando idade estiver entre ou for 18 e 25:

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE idade BETWEEN 18 [AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_and) 25;

Tabela

Descrição gerada automaticamente com confiança média

**SELECT – Filtros com o operador IN**

Retorna valores se eles atenderem aos requisitos passados.

Retorna se o interesse for = Jogos, Esporte ou Música.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE interesse = 'Jogos' [OR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_or) interesse = 'Esporte' [OR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_or) interesse = 'Música'; - **como seria feito sem o operador IN**

**Ex com operador IN.:**

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE interesse **IN**('Jogos','Esporte','Música');

Tabela

Descrição gerada automaticamente

Retorna se o interesse NÃO for = Jogos, Esporte ou Música.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE interesse **NOT IN**('Jogos','Esporte','Música');

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**SELECT – Filtros com o operador LIKE**

- Permite realizar filtros com base em uma pesquisa de caracteres de uma coluna textual.

**Caracteres coringas:**

**%** - Indica que pode haver a existência de qualquer conjunto de caracter no texto.

**\_** - Indica que pode haver a existência de um ou mais caracteres em uma posição específica do texto.

**A diferença é que com ‘%’ o texto pode ter qualquer quantidade de caractere, já com o ‘\_’, é a quantidade específica de ‘\_’ no comando;**

**Underline(\_)** pode ser utilizado quantas vezes for necessário;

Comando pode começar, terminar ou ter **%** ou **\_** no meio;

**Os coringas podem ser combinados dentro de uma mesma condição.**

Comando retorna todo nome que termina com **e**.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE nome [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html%23operator_like) '%e';

Tela de celular com publicação numa rede social

Descrição gerada automaticamente com confiança média

Retorna os nomes que terminam com **ne**.

Tabela

Descrição gerada automaticamente com confiança baixa

Retorna qualquer nome que tenha a letra **a** no meio;

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE nome [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html%23operator_like) '%a%';

**%a%** - termina e começa com qualquer quantidade de letra e qualquer letra.

Texto

Descrição gerada automaticamente com confiança média

Retorna qualquer nome que inicie com a letra **C**.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE nome [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html%23operator_like) 'c%';

Tela de celular com publicação numa rede social

Descrição gerada automaticamente com confiança média

Retorna todo nome que tem uma primeira letra qualquer e termina com ‘riel’.

Interface gráfica do usuário, Aplicativo, Site

Descrição gerada automaticamente

Retorna qualquer nome com 4 caracteres e que as 2 do meio são ’ru’.

Tela de celular com aplicativo aberto

Descrição gerada automaticamente com confiança média

Retorna nome que inicia com a letra A e tem mais 3 caracteres quaisquer.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Retorna nome que tiver **qualquer quantidade de caracteres**, **tt no meio** e **terminar com 1 caractere**.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `tb\_alunos` WHERE nome [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html%23operator_like) '%tt\_';

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**SELECT – Ordenando resultado**

- Palavra reservada: ORDER BY.

A construção de uma query (consulta), possui uma estrutura:

**SELECT**

<coluna(s)>

**FROM**

<tabela(s)>

WHERE **(opcional)**

<filtro(s)>

ORDER BY **(opcional)**

Idade **ASC**, nome **DESC**

**Ascending** (valor default) – acescente;

**Descending** – descendente;

Por mais que **Where** e **Order By** sejam opcionais, quando eles estão inclusos a ordem é importante.

**Ordena nome de forma ASC (a, b, c....):**

Interface gráfica do usuário, Tabela

Descrição gerada automaticamente

**Ordena nome de forma DESC (z, y, x...):**

Tabela

Descrição gerada automaticamente

***nome*** ordenada de forma **ASC**, ***idade*** ordenada de forma **DESC** baseada no nome e ***estado*** ordenada de forma **ASC** com base no nome e na idade**:**

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**SELECT – Limitando retorno**

Recurso muito utilizado na **limitação da quantidade de registros** **no momento de retorno** da consulta.

Utilizado quando a tabela ou o conjunto de **tabelas possuem muitos registros** e nós queremos **limitar** a **quantidade de registros** que serão **apresentados** **para** **usuário final**.

*Como em paginações:*

|  |  |
| --- | --- |
| ID | Descrição |
| 1 | Papel |
| 2 | Toalha |
| 3 | Mesa |
| 4 | Folha |

Pág. 1

|  |  |
| --- | --- |
| ID | Descrição |
| 5 | Apontador |
| 6 | Borracha |
| 7 | Cadeira |
| 8 | Tesoura |

Pág. 2

|  |  |
| --- | --- |
| ID | Descrição |
| 9 | Caneta |
| 10 | Grampeador |
| 11 | Cola |
| 12 | Quadro |

Pág. 3

Utiliza-se instrução **LIMIT**. Pode ser utilizado junto com o operador **OFFSET**;

A partir de qual registro devemos retornar uma determinada quantidade de registros;

**LIMIT** – opcional, mas quando usado, deve ser colocado ao final da query;

Podemos ter também **opcionalmente** um WHERE ou um ORDER BY, mantendo o LIMIT sempre no final;

**OFFSET** – só pode ser utilizado se LIMIT for declarado, e ele fica após. Porém, também é opcional.

s

Mostra (de 100), **25** registros.

Uma imagem contendo Tabela

Descrição gerada automaticamente

Mostra ***25 últimos registros*** (pois estão ordenados em ordem ***DESC***).

Tabela

Descrição gerada automaticamente

OFFSET pode ser exibido de 2 formas:

*Usando a palavra reservada:*

**LIMIT**

**5**

**OFFSET**

**2**

Ou

*Adicionando um 2° parâmetro ao LIMIT.*

**LIMIT**

**2.5**

***1°*** parâmetro = ***OFFSET***;

***2°*** parâmetro = ***LIMIT***.

Um exemplo:

Retornando 4 registros por página, na prática:

|  |  |
| --- | --- |
| ID | Descrição |
| 1 | Papel **0** |
| 2 | Toalha **1** |
| 3 | Mesa **2** |
| 4 | Folha **3** |

A partir do registro 0, traga 4 registros:

Tabela

Descrição gerada automaticamente

A partir do 4 registro, retorna 4 registros:

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

***Como é contado a partir do 0, o 4° registro é o número 5.***

A partir do 8°registro, retorna 4 registros:

Tabela

Descrição gerada automaticamente

**SELECT – Funções de agregação parte 1: MAX, MIN e AVG**

**Objetivo** das funções de agregação: **permitir** uma **operação aritmética** **sobre** os **valores** **de** **uma** determinada **coluna** **no registro de uma tabela**.

**SELECT**

**Funções de agregação**

**MIN(< coluna>)** – Retorna o **menor** valor de todos os registros com base em uma coluna;

**MAX(<coluna>)** – Retorna o **maior** valor de todos os registros com base em uma coluna;

**AVG(<coluna>)** – Retorna a **média** de todos os registros com base em uma coluna.

Para a utilização de tb\_cursos:

TRUNCATE tb\_cursos;

Limpa registros existentes na tabela.

Gráfico de dispersão

Descrição gerada automaticamente

Inserindo novos registros em tb\_cursos.

Retorna menor valor existente da coluna investimento

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) [MIN](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/aggregate-functions.html%23function_min)(investimento) FROM `tb\_cursos`;

Interface gráfica do usuário, Aplicativo, Site

Descrição gerada automaticamente

Retorna menor valor na coluna investimento, onde ativo é = true

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Maior valor na coluna investimento

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Exibe valor médio entre todos os valores da coluna ‘investimento’

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**SELECT – Funções de agregação parte 2: SUM e COUNT**

**SUM (<coluna>)** – Retorna a **soma** dos valores de todos os registros com base em uma coluna;

**COUNT (\*)** – Retorna **quantidade** de todos os registros de uma tabela.

**SELECT**

**funções de agregação**

**Investimento total (soma de todos os valores da coluna investimento):**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Soma de todos os registros ATIVOS:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamentemesmo valor pois curso que é inativo, equivale a 0,00 de investimento.

**Contando quantos registros ATIVOS existem na tabela:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Contando quantos registros existem na tabela:**

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

**SELECT – Agrupando seleção de registros (GROUP BY)**

**Pode ajudar nas seguintes situações de exibição:**

**-** Total de vendas por mês;

**-** Quantidade total de alunos por curso;

**-** Quantidade total de cliques por anúncio;

**-** Total de reclamações por categoria;

**-** Total de chamadas por número telefônico.

Utilizado na criação de relatórios que apresentem resultados com base nos registros agrupados;

Na sintaxe ele é posicionado após o **WHERE** ou **FROM** e antes do **ORDER** **BY** ou **LIMIT**;

**SELECT**

<coluna(s)>

**FROM**

<tabela(s)>

**WHERE**

<filtro(s)>

**GROUP BY**

**agrupamento**

**ORDER BY**

<ordenacao>

**LIMIT**

<offset>, <limit>

**agrupamento** – agrupa os registros com base em uma ou mais coluna cujos valores sejam iguais. Permite realizar funções de agregação em cada subconjunto agrupado de registros.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente



Count faz seu papel em cada item na tabela de interesse

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Retornando apenas coluna de interesse e valores

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Adicionando **AS** na coluna de de **Count** interesse

Interface gráfica do usuário

Descrição gerada automaticamente com confiança baixa

Retornando as 27 possibilidades de registro de estados

Interface gráfica do usuário, Texto

Descrição gerada automaticamente com confiança média

Retornando quantidade de registro por estado

**SELECT – Filtrando seleções agrupadas (HAVING)**

**Depende da instrução GROUP BY;**

**GROUP BY existe sem HAVING, mas HAVING não existe sem GROUP BY.**

**SELECT**

<coluna(s)>

**FROM**

<tabela(s)>

**WHERE**

<filtro(s)>

**GROUP BY**

<agrupamento>

**HAVING**

**filtro(s) sobre agrupamento**

**ORDER BY**

<ordenacao>

**LIMIT**

<offset>, <limit>

**filtro(s) sobre agrupamento –** Filtro realizado sobre o resultado dos agrupamentos (GROUP BY);

Retorna quantidade de registros por estados se for maior ou igual a 5:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Só retorna se a sigla dos estados for ‘MG’ ou ‘SP´.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Retorna se a sigla do estado for ‘CE’ ou ‘SC’ e o a quantidade de registro do estado for maior que 4.

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

Retorna registro que o **interesse é diferente de Esportes**, agrupados por estado, desde que o total de registro por estado seja maior que 3.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**UPDATE – Atualizando registros**

Update faz parte do subconjunto do SQL – DNL.

**UPDATE**

**<tabela>**

**SET**

**<coluna> = <valor>, <coluna> = <valor>, <coluna> = <valor>**

**WHERE - opcional**

**<filtro(s)>**

Podem ser acessadas ao mesmo tempo várias ou uma coluna.

[UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) tb\_alunos [SET](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/set.html) interesse = 'Saúde' WHERE idade >= 80;

Quando pessoa cadastrada no registro tiver a idade >= a 80, interesse = saúde.

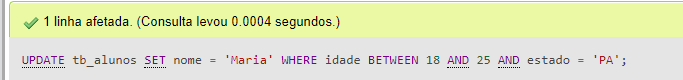
Uma imagem contendo Texto

Descrição gerada automaticamente

Quando id\_aluno for =18, os seguintes dados serão atribuídos ao respectivo registro:

Nome = ‘Ana’, idade = 17, email = ‘aninha@gmail.com’.

**Conectar operadores de comparação através de operadores lógicos**



Quando a idade for entre **18 e 25** **e** o estado for **‘PA’**, muda o nome para Maria.

**DELETE – Excluindo registros**

**Essa instrução faz parte do subconjunto de instruções do SQL – DML (Data Manipulation Language). Comando para manipulação de dados.**

**DELETE FROM**

<tabela>

**WHERE**

<filtro(s)>

Serão removidos os registros que idade for uma das citadas e interesse for igual a ‘Esporte’.

[DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) FROM tb\_alunos WHERE idade [IN](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/comparison-operators.html%23function_in)(10, 18, 22, 28, 34) [AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_and) interesse = 'Esporte';

**Introdução ao relacionamento entre tabelas, chave primária e estrangeira**

O nome **banco de dados relacionais** não é por acaso;

Apesar de tabelas serem implementadas com fins bem específicos, na prática é importante efetuar o **relacionamento entre** tais **tabelas**;

**Relacionamento entre tabelas** – consiste em tentar imitar a relação entre as coisas no mundo real. Levando essas relações para os nossos registros.

Existem **3 tipos de relacionamentos**:

**-** Um para UM;

**-** Um para Muitos;

**-** Muitos para Muitos.

**Chave primária** – constituída por 1 ou mais campos.

*Objetivo*: servir como um identificador único para cada registro dentro de uma tabela;

Esse identificador não pode se repetir em nenhum momento para nenhum outro registro;

Muito utilizada para a seleção de registros, atualização de registros, remoção de registros e criação de relacionamentos consistentes entre tabelas;

Quando a chave primária é constituída por 2 ou mais campos de um determinado registro dentro de uma tabela, é uma **chave primária composta**.

**Chave estrangeira** – Referência da chave primária de outra tabela, a partir de uma chave estrangeira.

**Projeto Loja Virtual – Relacionamento Um para Um**

Objetivo: criar estrutura de dados que suporte uma aplicação com a proposta de venda de produtos pela internet;

Serão criadas 2 tabelas:

**- tb\_produtos;**

*produto*

*valor*

**- tb\_descricoes\_tecnica**

*descricao\_tecnica*

**db\_loja\_virtual > tb\_produtos>id\_produto INT NOT NULL PRIMARY KEY,**

**produto VARCHAR (200) NOT NULL,**

**valor FLOAT (8,2) NOT NULL**

No momento de criação da tabela pode-se definir a primary key, colocando-a na coluna em que ela irá ficar;

O **Primary Key** nos dá a garantia de que o **valor** contido dentro dessa coluna. **não se repetirá para registros diferentes**.

Campos do tipo **AUTO\_INCREMENT** – gerenciados pelo próprio banco de dados, de modo que **ao inserir um novo registro** a informação que será **atribuída aquela coluna** será feita com base no **último valor inserido + 1**.

[CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) tb\_descricoes\_tecnicas( id\_descricao\_tecnica INT [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL PRIMARY KEY AUTO\_INCREMENT, id\_produto INT [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL, decricao\_tecnica TEXT [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL, FOREIGN KEY(id\_produto) REFERENCES tb\_produtos(id\_produto) );

**id\_produto - chave estrangeira.**

Cria tabela descrições técnicas;

*Id da descrição* – tipo: **numérica inteiro**, **não pode ser vazia**, **primary key** e **auto incrementável**;

*Id\_produto* - **relaciona tabela** com os registros da tabela tb\_produtos, **int** pois debe ser compatível com a chave que está sendo importada para a tabela, não pode ser vazia, pois toda descrição técnica precisa de um produto;

*Descrição técnica* – **text** (texto da descrição ficará aqui), **não pode ser vazia**;

***FOREIGN KEY () – espera um parâmetro que é o nome da coluna dentro da tabela que será utilizada como chave estrangeira.***

***REFERENCES <tabelaorigem> () – entre parênteses será colocado o campo primary key,***

*FOREIGN KEY (id\_produto) REFERENCES tb\_produtos (id\_produto)* – utilizando como **referência** a PrimaryKey (**id\_produto**) **da tabela tb\_produtos** para a tabela (tb\_descricoes\_tecnicas) como chave estrangeira (id\_produto).

**A chave estrangeir não precisa ter o mesmo nome da chave primária de referência.**

**Populando tabelas:**

tb\_produtos:

**INSERT INTO tb\_produtos(produto, valor) VALUES ('Notebook Dell Inspiron Ultrafino Intel Core i7, 16GB RAM e 240GB SSD', 3500.00);**

**INSERT INTO tb\_produtos(produto, valor) VALUES ('Smart TV LED 40" Samsung Full HD 2 HDMI 1 USB Wi-Fi Integrado', 1475.54);**

**INSERT INTO tb\_produtos(produto, valor) VALUES ('Smartphone LG K10 Dual Chip Android 7.0 4G Wi-Fi Câmera de 13MP', 629.99);**

tb\_descricoes\_tecnicas:

**INSERT INTO tb\_descricoes\_tecnicas(id\_produto, descricao\_tecnica) VALUES (1, 'O novo Inspiron Dell oferece um design elegante e tela infinita que amplia seus sentidos, mantendo a sofisticação e medidas compactas...');**

**INSERT INTO tb\_descricoes\_tecnicas(id\_produto, descricao\_tecnica) VALUES (2, 'A smart TV da Samsung possui tela de 40" e oferece resolução Full HD, imagens duas vezes melhores que TVs HDs padrão...');**

**INSERT INTO tb\_descricoes\_tecnicas(id\_produto, descricao\_tecnica) VALUES (3, 'Saia da mesmice. O smartphone LG está mais divertido, rápido, fácil, cheio de selfies e com tela HD de incríveis 5,3"...');**