Estudos MySQL

Aprendizagem sobre MySQL

V1.0.0 (OCT/2023)

Sumário

1 INTRODUZINDO

TIPOS DE COMANDOS

DDL - data definition language >São comandos de definição de estrutura. >Create data base >Create table >Alter table >Drop table

DML - data manipulation language >São comando para manipulação do banco de dados. >Insert into. >Update >Delete >Truncate

DQL - Data query language >Select

Criar um banco de dados.

Create database <nome do banco de dados>;

Apagar banco de dados.

Drop database <nome do banco de dados>;

Habilitar caracteres acentuados

create database cadastro default character set utf8 default collate utf8mb3_general_ci;

Criar tabelas. Create table <nome da tabela>(aqui são colocados os campos... id int not null auto_increment, nome varchar(30) not null, nascimento date, sexo enum('M', 'F'), peso decimal(5,2), altura decimal(3,2), nacionalidade varchar(20)default 'Brasil')default charset = utf8mb3;

> **not null** - Obriga o usuário a digitar algo.
>
> **date** - Tipo primitivo para datas.
>
> **enum('M', 'F')** - Faz com que só aceite um dos caracteres desejados.
>
> **decimal(5,2)** - Indica que o número terá 5 digitos e 2 deles serão pós vírgula.
>
> **auto_increment** - adiciona 1 unidade no id.
>
> **Chaves primárias** - adicionado nome id(identificador), not null, auto_increment

2 Inserir dados

2.1 Código

```
insert into pessoas
(id, nome, nascismento, sexo, peso, altura, nacionalidade)
values
('default', 'Godofredo', '1984-1-02', 'M', '78,5', '1.83', 'Brasil');
```

2.2 Explicando o código

INSERT INTO pessoas >INSERT INTO é o comando, “pessoas” é o nome da tabela.

(nome, nascimento, sexo, peso, altura, nacionalidade) >Constantes sem parâmetros pois isso é após adicionar a tabela. > >Se necessário mostrar “id”, então use “DEFAULT” no valor, para seguir o padrão.

VALUES >Comando para inserir valores as constantes

(‘Godofredo’, ‘1984-1-02’, ‘M’, ‘78,5’, ‘1.83’, ‘Brasil’);

> **Valores que serão adicionado as contantes, tem que estar na mesma ordem do parêntese de “INSERT INTO”**
>
> **Outras formas para escrever esse código. Ex:** insert into pessoas values (‘Godofredo’, ‘1984--1-2’, ‘M’, ‘78,5’, ‘1.83’, ‘Brasil’), (‘Luana’, ‘1990-12-22’, ‘F’, ‘60’, ‘1,67’, ‘França’), (‘Ana’, ‘1978-10-01’, ‘F’, ‘1,83’, ‘2’, ‘EUA’);
>
> **Assim podemos adicionar quantas pessoas quisermos em apenas uma linha de código.**
>
> **Caso os valores não estejam na mesma ordem das constantes, o código não vai falhar, para corrigir, basta adicionar o nome de cada constantes logo após a linha do “insert into”. Ex:**

```
insert into pessoas
(id, nome, nascimento, sexo, peso, altura, nacionalidade)
values
(default, 'Godofredo', '1984-1-2', 'M', '78,5', '1.83', 'Brasil'),
(default, 'Luana', '1990-12-22', 'F', '60', '1,67', 'França'),
(default, 'Ana', '1978-10-01', 'F', '1,83', '2', 'EUA');
```

3 Manipulando tabelas e colunas

Criar colunas em tabelas

alter table `nome da tabela` add column `nome da coluna` `tipo primitivo`

> A coluna irá para última posição da tabela

Remoção de colunas

alter table `nome da tabela` drop column `nome da coluna`;

Adicionar uma coluna em posição especifica

alter table `nome da tabela` add column `nome da coluna` `tipo primitivo` after `nome da futura coluna antecessora`;

Adicionar a tabela na primeira posição

alter table `nome da tabela` add column `nome da coluna` `tipo primitivo` first;

Modificar definições de coluna alter table `nome da tabela` add column `nome da coluna` `novo tipo primitivo` `nome da constante para adicionar, alterar ou mudar`;

> Possibilita alterações nas constantes da coluna.

Alterar nome das colunas

alter table `nome da tabela` change column `nome atual da coluna` `novo nome da coluna` `tipo primitivo` `constantes`

> Se não informar as contantes existêntes, a coluna irá remover as contantes da coluna.

Alterar nome da tabela

alter table `atual nome da tabela` rename to `novo nome da tabela`

3.1 Criando e manipulando tabelas

CÓDIGO

```
1 Create table if not exists cursos(
2 nome varchar(30) not null unique,
3 descricao text,
4 carga int unsigned
5 totaulas int,
6 ano year default '2016'
7 ) default charset = utf8mb3;
```

> **if not exists** - “se não existir”, no caso é apenas se não exitir a tabela “cursos”, se tentar criar uma tabela com nome já existênte, irá apagar a tabela atual e criar uma nova, porém, vazia.
>
> **Para adicionar chave primaria é simples. Ex:**
>
> alter table `cursos` add column id int first; alter table `cursos` add primary key(id);

Apagar tabelas

drop table if exists `nome da tabela`;

4 Tipos primitivos

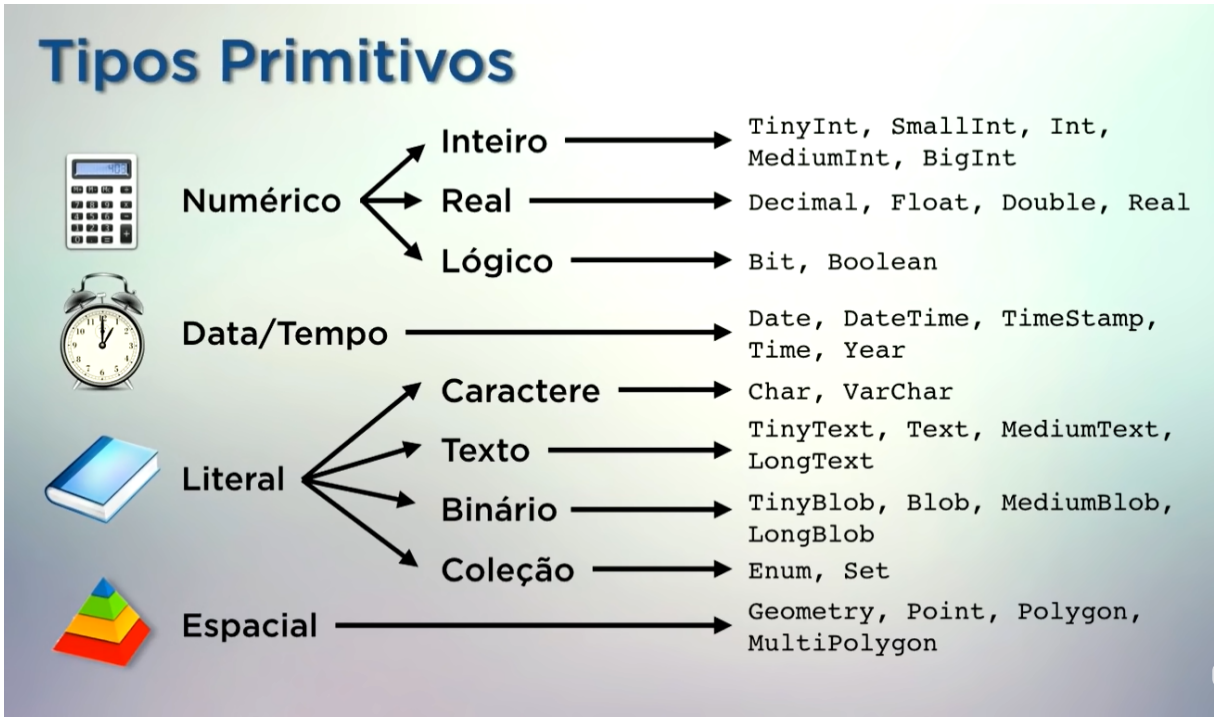

Figura 1: imagem

5 Manipulando linhas

> **Comando Update**
>
> **Serve para atualizar a linha da tabela.**
>
> **Código modificando linhas da coluna:**
>
> update `nome da tabela` set `nome da coluna` = '`novo nome da linha`' where `nome` **`do`** `id` = '`número da linha`';
>
> **Explicando o código**
>
> Update - significa "Atualizar". Set nome = - especifica a coluna que irá ser modificada. where - significa "onde", serve para dizer onde ocorrerá a modificação id - a modificação será direcionada baseada no primary key (id) e em seguida informaremos o número da linha com relação ao id.

Exemplo visual:

TABELA EXEMPLO

coluna id	coluna 1	coluna 2	coluna 3	coluna 4
1	NOME			
2				
3				
4				
5				

> **Atualizando dados:**
>
> update `exemplo` set `coluna 1` = '`NOVO NOME`' where `coluna id` = '`1`';

TABELA EXEMPLO (ATUALIZADA)

coluna id	coluna 1	coluna 2	coluna 3	coluna 4
1	NOVO NOME			
2				
3				

coluna id	coluna 1	coluna 2	coluna 3	coluna 4
4				
5				

5.1 Para modificar duas informações na mesma linha

TABELA EXEMPLO

coluna id	NOME	coluna 2	ANO	coluna 4
1	Curso A		2000	
2	Curso B		2001	
3	C		1639	
4	Curso D		2003	
5	Curso E		2004	

> **Atualizando dados:**
>
> update `exemplo` set `NOME` = '`Curso C`', `ANO` = '`2002`' where `coluna id` = '`3`' limit 1;
>
> **Note que adicionamos um parâmetro LIMIT isso faz serve para previnir que o comando update atualize todas as linhas.**

TABELA EXEMPLO (ATUALIZADA)

coluna id	NOME	coluna 2	ANO	coluna 4
1	Curso A		2000	
2	Curso B		2001	
3	Curso C		2002	
4	Curso D		2002	
5	Curso E		2002	

> **Para atualizar todas as linhas de uma ou mais colunas**
>
> update `exemplo` set `NOME` = '`Curso X`', `ANO` = '`2023`' where `ano` = '`2002`'
>
> Modificará todos os espaços de nome e ano desde que o ano da antiga coluna seja 2002

coluna id	NOME	coluna 2	ANO	coluna 4
1	Curso A		2000	
2	Curso B		2001	
3	Curso X		2023	
4	Curso X		2023	
5	Curso X		2023	

> **Alerta!**
>
> O uso errado desse comando faz com que o altere uma grande quantidade de conteúdo, o MySQL deixa pré-configurado um campo para evitar esse tipo de ação, isso pode ser alterado nas configurações, mas deve ser usado com bastate atenção.

Removendo linhas

> Remove linhas indesejadas de uma tabela.

Delete from `nome da tabela` where id = `número da linha`

Apagando todos os dados da tabela

> truncate table `nome da tabela`;
>
> pode ser simplificado com:
>
> truncate `nome da tabela`

6 Select Parte 1

Select * From `Nome da tabela`; > Select - é o comando que significa “selecionar” >* - significa selecione todas as colunas(registros). >From - É de onde, especificar o lugar. > >Podemos filtrar com adicionando algumas especificações.

Select * from `Nome da tabela`. Order by `nome`;

> Comando irá mostrar a coluna de acordo com a ordem alfabética.

> Podemos ordernar também por: Nome: Ordem alfabeta. ID: Ordem númerica. Ano: Anual Data de nascimento: Ordem de nascimento

> Podemos também adicionar ordens como:

Order by `nome` asc; >Para ascendente.

Order by `nome` desc; >Para decrescente.

Select `nome`, `carga`, `ano` from `nome da tabela` Order by `nome`;

> isso faz com que o comando selecione apenas as colunas desejadas. Comando order vai ordenar pela ordem especificada.

> Podemos fazer ordens multiplas e perdir especificar o sentido (Ascendente ou decrescente).

Select `nome`, `carga`, `ano` from `nome da tabela` Order by `ano` asc, `nome` desc;

> Fazendo isso só irá mostrar as colunas nome carga e ano. Nos registros ficará ordernado primeiro por ano de forma ascendente e depois será ordenado por nome em decrescente.

> Podemos deixar ainda mais especifico.

> **Select `nome`, `carga`, `ano` from `nome da tabela` Where ano = ‘2016’ Order by `nome`;**

> Adicionando where faz com que o comando selecione as colunas `nome`, `carga`, `ano` da tabela especifica onde o ano seja exclusivamente igual a 2016 e ordenado por nome. Podemos usar operdores relacionas juntamente com o where para ter uma tabela especifica.

Operadores relacionais

Diferente: <> `ou` != Menor que: < Maior que: > Menor ou igual a: <= Maior ou igual a: >=

`Between` - Podemos adicionar uma faixa, exemplo:

> **Where ano between 2016 and 2020**
>
> Irá mostrar aquivos de 2016, 2017, 2018, 2019 e 2020.
>
> `In` - Mostra apenas anos especificos, exemplo:
>
> **Where ano in (2016, 2020)**
>
> Mostrará arquivos apenas do ano de 2016 e 2020.

7 Select Parte 2

Operador relacional LIKE

Select * from `nome da tabela` Where nome like 'P%';

> Nesse caso, comando se lê assim: Selecione (select) Todas as colunas (*) Da tabela (`nome da tabela`) Onde (where) Tenha algo que se pareça (like) 'P%' (P seguido de qualquer coisa, inclusive de nada).
>
> % - significa nenhum ou vários carateres
>
> Se o % for colocado antes de uma letra (%A), então o comando pegará registros que termine com a letra "A"
>
> Para puxar todos os registros que contém "A" em qualquer posição, basta usar `'%A%'`
>
> O comando pegará todos os dados terminados com "A", independente se é maiúscula ou minúscula.
>
> Senão quisermos puxar dados com uma letra especifica, basta adicionarmos o not like, e com parâmetro `'L%'`, `'%L'ou '%L%'`, exemplo:

Select * from `nome da tabela` Where nome not like '%L%';

> Temos outro caratere coringa o _, ele é parecido com o %, porém, exige que tenha algum caractere.

Select distinct `nome da coluna` from `nome da tabela`;

> Serve para distinguir. Considera uma ocorrência de cada valor dentro do registro. Mostrar todos os arquivos diferentes, sem repetir.

Select count(`* ou nome da coluna`) from `nome da tabela`

> Tem a função de contar registros

Select max(`* ou nome da coluna`) from `nome da tabela`

> Server para mostrar o máximo, como por exemplo o máximo de horas que teve em um agrupamento.

Select min(`* ou nome da coluna`) from `nome da tabela`

> O mesmo para o máximo, porém pegar número mínimo de um agrupamento. **Select som(`* ou nome da coluna`) from 'nome da tabela**

> Tem a função de somar, podemos deixar mais especificado usando o where, para somar campos especificos.

Select avg(`* ou nome da coluna`) from `nome da tabela`

> Serve para médias, caucular média de um campo especifico.

8 Select Parte 3

Comandos de agrupamentos

Select `nome da coluna` from `nome da tabela` group by `nome da coluna`

> **Group by** - agrupado por.

> Podemos usar os parâmetros de distinção e agrupamento juntos, exemplo:

Select `nome da coluna`, count(nome) from `nome da tabela` group by `nome da coluna`

> O count vai criar um outra coluna que agrupa e conta quantos registros tem nesse agrupamento.

Having

> O Having é uma extensão do group by Serve para especificar algo dentro da coluna Podemos também usar o having especificando juntamente com outro select.

9 Modelo relacional

> **Para iniciarmos modelo relacional, precisamos entender uma coisa.**
>
> **O banco de dados é visto como diagrama.**
>
> **Tabelas são chamadas de entidades.**
>
> **Colunas são chamadas atributos.**
>
> **levando isso em consideração temos o DER, que significa: diagrama entidade-relacionamento. Relacionamento é uma relação de atributos entre entidades.**
>
> > **Cardinalidade** - é o de tipo de relacionamento entre entidades, temos três tipos entidade se relaciona com os atributos.
> >
> > **tipos de cardinalidades.** * 1 para 1 * 1 para muitos * muitos para muitos
>
> **Chave estrangeira** - chave primaria de uma entidade que é colocada em uma outra entidade, então receberá o nome de chave estrangeira.
>
> **Criar chave estrangeira**
>
> alter table ‘nome da tabela/entidade add column nome da chave estrangeira int(mantendo o tipo da chave de origem);
>
> alter table ‘nome da tabela atual`` add foreign key(nome da chave estrangeira) referencesnome da tabela da chave origem(nome original da chave“‘);
>
> > Após isso a chave estrangeira será adicionada em forma de coluna vazia na tabela. para adicionar algo no campo da chave estrageira basta usar o comando update.
>
> update nome da tabela onde foi adicionado a chave set nome da chave estrangeira = 'id do conteúdo contido na tabela estrangeira' where id(chave primária)1
>
> > **Atenção! Não é possivel usar o comando delete em uma entidade se houver um relacionamento com outra entidade. Isso afetará uma das regras de ACID, a regra de inconsistência.**

9.1 Regras relacionais

9.1.1 1 para 1

> **Definir entidade dominante Chave trasferida se torna estrangeira Mantém o mesmo tipo(int, varchar...)**

9.1.2 1 para muitos

> **Transporta a chave primária de 1 e coloque em muitos Chave se torna estrangeira Mantém o mesmo tipo**

9.1.3 Muitos para muitos

> **Relacionamento se tranforma em entidade, com atributos relacionais As chaves das entidades anteriores, transferem as chaves para entidade-relacionamento**
>
> **Regra de transação ACID**
>
> **Todas as ações que o banco de dados possa executar.**
>
> **Atomicidade - tudo é feito ou nada é considerado. Consistência - tudo que estava consistênte, tem que se permanecer consistênte após a transação. Isolamento - as transações devem ser executada como se tivesse sendo feita de forma isolada. Durabilidade - os dados devem ser durável o suficiente.**

10 Junções

> **Comando Join** Basta selecionar as colunas desejadas que deseja unir From `nome da tabela` inner join `segunda tabela` on `Chave primária` = `Chave estrageira`;
>
> > Básicamente, devemos selecionar as colunas que desejamos unir, após isso, devemos usar o comando `inner join` e obrigatóriamente usar a claúsula `on`, para dar sentido, o `on` apenas vincula as chaves
> >
> > O `inner join`, faz com que a junção não inclua os registros que não tem relação, para adicionar todos, devemos usar o `outer join` e especificar a tabela que irá dar preferência.
>
> left outer join ou right outer join
>
> > Comando mudará o sentido conforme o comando.
> >
> > Podemos usar o comando As, serve para criar um apelido a coluna/tabela, para facilitar o comando e deixar o comando mais limpo, deverá ser usado logo após o from.
>
> From `TabelaA` as `TA` inner join `TabelaB` as `TB`

11 Usando inner join com várias tabelas

> **Criando relações entre tabelas muitos para muitos** >Para isso devemos pensar que temos 2 entidades >Primeira tabela(alunos) com seus respetivos atributos (cpf(id)nome, idade, nacionalidade) >Segunda tabela(curso) atributos(idcurso,nome do curso, carga horária) > >Em seguida, para criar um relacionamento entre duas entidades devemos pensar que as entidades estão na extremidades de uma relação. >A relação deverá se tornar uma entidade com seus atributos e em seguida as entidades das extremidades deverão ser colocadas como 1 para muito com relação a entidade central

Alunos 1	N Relação N	1 Cursos
idaluno	id	idcurso
nome	Data	Nome do curso
idade		
nacionalidade		

> Create table alunos_relação_cursos (id int not null auto increment, data date, idaluno int, idcurso in, primary key(id), foreign key (idaluno) refenrences aluno(id), foreign key (idcurso) references curso(id)) default charset utf8mb3;
>
> > Isso é para criar a tabela de relação com as duas chaves estrangeiras das entidades das extremidades, necessário colocar o mesmo tipo primitivo das chaves
> >
> > Tendo três entidades faz com que a tabela de relação crie diferentes id para cada curso que os alunos irão fazer.
> >
> > Para fazer junções entre as três tabelas devemos fazer:
>
> Select * from `aluno` join `alunos_relação_curso` on `alunos.id` = `alunos_relação_curso.idalunos` join `cursos` on `alunos_relação_curso.idcurso` = `curso.idcurso`;
>
> > Fazendo isso, estamos puxando conteúdo das três tabelas.