DDL (CREATE, ALTER, DROP)

**Banco de dados relacional**

Os códigos a baixo são para criação de um banco de dados e tabela utilizando o mysql

--criando banco de dados

**create database wm;**

**-- criando tabela estado**

**create table estado (**

**id int unsigned not null auto\_increment,**

**nome varchar(45) not null**,

sigra varchar(2) not null,

regial ENUM ('norte', 'nordeste', 'centro-oeste','sudeste' , 'sul' ) not null,

populacao DECIMAL(5,2) not null,

primary key(id),

unique key(nome),

unique key(sigla)

);

Os comandos acima criam um banco de dados uma tabela e define as regras da coluna da tabela. Já o código a seguir faz a inserção dos valores na tabela estado

**insert into estado (nome, sigla, regial, populacao)**

**value ('bahia','ba', 'nordeste', 15.84)**

Após criar a tabela estado agora vamos desenvolver a tabela cidade e nessa tabela teremos que referenciar a tabela estado para criarmos uma relação entre as tabelas. A baixo esta um exemplo

**create table cidade(**

**id INT unsigned NOT NULL auto\_increment,**

**nome VARCHAR(255) NOT NULL,**

**estado\_id INT unsigned NOT NULL,**

**area decimal(10,2),**

**primary key (id),**

**foreign key (estado\_id) references estado (id)**

**)**

A coluna **estado\_id** será referencia para a coluna **id** de estado destacando que nessa relação especifica cidade pode ter um estado.

Agora vamos criar a tabela de prefeitos que vai possuir uma relação de um para um com cidade a baixo esta o exemplo

**create table** PREFEITOS (

ID **int unsigned not null auto\_increment**,

NOME **varchar(255) not null**,

CIDADE\_ID **int unsigned**,

**primary key(**ID**),**

**unique key** (CIDADE\_ID),

**foreign key**(CIDADE\_ID) **references** CIDADE (ID)

Podemos notar que a coliuna que faz referencia para cidade CIDADE\_ID foi definida como unique oi seja podemos ter apenas uma cidade para prefeito essa configuração determina a relação de uma cidade por prefeito

**Relação muitos para muitos**

Quando temos um relacionamento many to many temos que desenvolver uma nova tabela que vai receber o id das duas tabelas que possuem uma relação muitos para muitos

**create table** empresas(

id **int** **unsigned** **not null auto\_increment**,

nome **varchar(255) not null**,

cnpj **int** **unsigned unique key**,

primary **key**(id)

);

**create table** empresas\_unidade(

empresa\_**id int unsigned not null**,

cidade\_id **int unsigned not null**,

sede **tinyint(1) not null,**

**primary key**(empresa\_id, cidade\_id)

)

Na comando acima criamos a tabela empresas que possui uma relação muitos para muitos com cidade, essa relação esta sendo construída pela tebela empresas unidade que como podemos ver recebeu o id de ambas as tabelas para criar essa relação

**Alter**

Com o comando alter é possível realizar alterações na estrutura da tabela já desenvolvida

**alter** **table** empresas **modify** cnpj **varchar(14)**

O comando apresentado modifica a coluna cpf para varchar essa coluna já existia como int. Note como podemos facilmente alterar o tipo de dado da coluna

CRUD (DML)

**SELECT**

Agora iremos saber como funciona o selects. A baixo deixarei algumas consultas

**select \* from** estado;

**select** nome , sigla **from** estado;

**select** nome **as** 'nome do estado', sigla, populacao **from** estado **where** populacao >= 10 **order by** populacao **desc**;

A primeira consulta busca todos os dados no banco de dados, o \* no select determina que pode ser mostrada todas as colunas da tabela . A seguir temos um select que também traz todos os dados do banco porem ele determina qual coluna deve ser exibida indicando seu nome após o select. E por fim temos um select com restrições utilizando o comando where que determina qual os dados devem ser recebidos e com ordenação utilizando o order by (coluna ) desc temos o comando as também que determina o nome da coluna que será exibido na consulta

É possível em um select a partir da tabela indicada isso ajuda em consultas muito grandes, veja a baixo um exemplo

**Select** est.nome **from** estado est **where** sigla = ‘ma’

Podemos notar que utilizamos o apelido na frente da coluna indicada para reforçar que aquela coluna pertence ao estado

**SELECT COM FUNÇÃO AGREGADORA**

Nesse select utilizamos alguns comando pre determinados do sql para realizar algumas funções

**SUM**: realiza a soma indicada

SELECT regiao AS ‘região’

SUM(populacao) as Total

FROM estado

GRUP BY região

ORDER BY total DESC

**AVD**: realiza a media indicada

SELECT AVG (população) AS total FROM estado

**SELECT DE TABELAS UTILIZANDO O COMANDO INNER JOIN**

Podemos fazer um select buscando varias tabelas utilizando o comando **where** ou o **innerjoin** segue exemplo no código a baixo

**select**

e.nome **as** Estado,

c.nome **as** cidade,

regial **as** regiao

**from** estado e, cidade c

**where** e.id = c.estado\_id;

.............................................................................................................................

**select**

e.nome **as** Estado,

c.nome **as** cidade,

regial **as** regiao

**from** estado e **inner join** cidade c

**on** e.id = c.estado\_id;

Os comandos acima retornam resultados parecidos com métodos diferentes. No primeiro select podemos notar a utilização da clausula **where** na para buscar os resultados já no segundo select vemos o uso do **inner join** . A seguir teremos mais informações de como esses comandos funcionam e os tidos de join existentes e como cada um deles funciona

**Tipos de InnerJoin**

Contamos Com 4 tipos de join são eles o **inner Join, left Join, right Join e o full Join** (o fullJoin não e suportado pelo mysql mas pode se simulado )

* **Inner Join**: Representa os dados que possuem ligação entre as duas tabelas
* **Left Join**: Traz os dados que possuem ligação entre as tabelas e os dados sem ligação da tabela a esquerda
* **Right Join**: Esse funciona de forma oposta ao leftJoin traz os dados ligados entre as tabelas e os da tabela a direita da relação
* **Full Join**: Essa função não e suportada pelo mysql mais pode ser simulada, ela e responsável por trazer todos os dados tanto os com ligação e os sem ligação de todas as tabelas

A baixo temos os exemplos dos 4 tipos de joins lembrando que o full join esta sendo simulado

**select** \* **from** cidade c **inner** **join** prefeitos p **on** c.id = p.id;

**select** \* **from** cidade c **left** **join** prefeitos p **on** c.id = p.id;

**select** \* **from** cidade c **right** **join** prefeitos p **on** c.id = p.id;

## **select** \* **from** cidade c **full** **join** prefeitos p **on** c.id = p.id; ( esse comando não funciona no mysql )

**select** \* **from** cidade c **left** **join** prefeitos p **on** c.id = p.id

**union**

**select** \* **from** cidade c **right** **join** prefeitos p **on** c.id = p.id

**Selecionando Muitos para muitos**

**select** c.nome **as** 'cidade', e.nome **as** 'empresa'

**from** cidade c , empresas e , empresas\_unidade eu

**where** c.id = eu.cidade\_id

**and** e.id = eu.empresa\_id

Pra realizar uma busca many to many utilizamos a terceira tabela que foi desenvolvida para realizar a junção das tabelas, como vemos acima na clausula from chamamos as 3 tabelas e no where indicamos que os ids de cidade e empresa deve corresponder com seus respectivos ids da tabela empresa\_unidade responsável pela relação entre elas

**UPDATE**

Agora vamos aprender a trabalha com update, veja a baixo um exemplo

update estado

set nome ='bahia test' where sigla = 'ba';

**update**: comando indica a tabela que será atualizada

**set**: indica a coluna ou colunas que deseja realizar a alteração

**where**: onde deve ser o valor alterado

Utilizar o updade sem a clausula where atualiza todos os dados da tabela podendo ser muito arriscado dependendo do tipo de alteração

**DELETE**

O domando DELETE nos permite deletar uma ou mais colunas de acordo com a condição determinada, a baixo esta um exemplo de como é feito

DELETE FROM estado WHERE id >= 1000

No comando acima estamos deletando todas as colunas com id maior que 1000 e assim como o comando UPDATE o comando DELETE utilizado sem o WHERE apaga todas as colunas do banco de dados.

**INSERT**

Agora vamos aprender a inserir valores em nosso banco de dado.

**insert into** estado **(**nome, sigla, regial, populacao**)**

**value (**'minas','mg', 'nordeste', 30.84**);**

**insert into** estado **(**nome, sigla, regial, populacao**)**

**value (**'pernanbuco','pe', 'nordeste', 13.84**);**

Com os comando acima fazemos a inserção de valores na tabela estado agora vamos fazer uma inserção de valores na tabela cidade que possui uma foreng key

**insert into** cidade **(**nome, area, estado\_id**)**

**value (**'são paulo',133.00, 3**);**

**insert into** cidade **(**nome, area, estado\_id**)**

**value (**'copacabana',52.33, **(select** id **from** estado **where** sigla **=** 'rj'**));**

Podemos notar que neste segundo **insert** de cidade nos utilizamos um **select** para buscar o id do estado