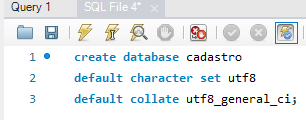
* Criar banco de dados:

Create database nomedobanco;

Os demais parâmetro servem para definir o tipo, exemplo aceitar acentos.

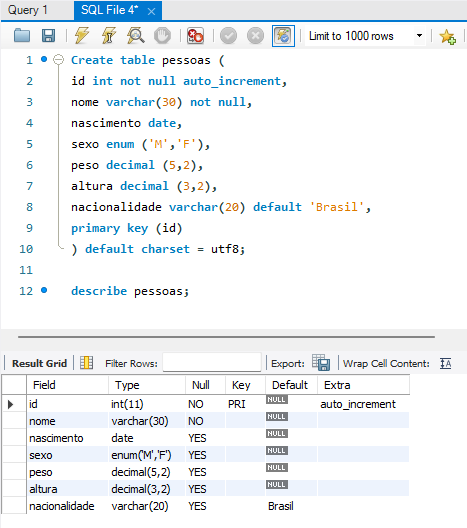


* Apagar banco de dados

Drop database nomedobanco;

* Criar tabelas

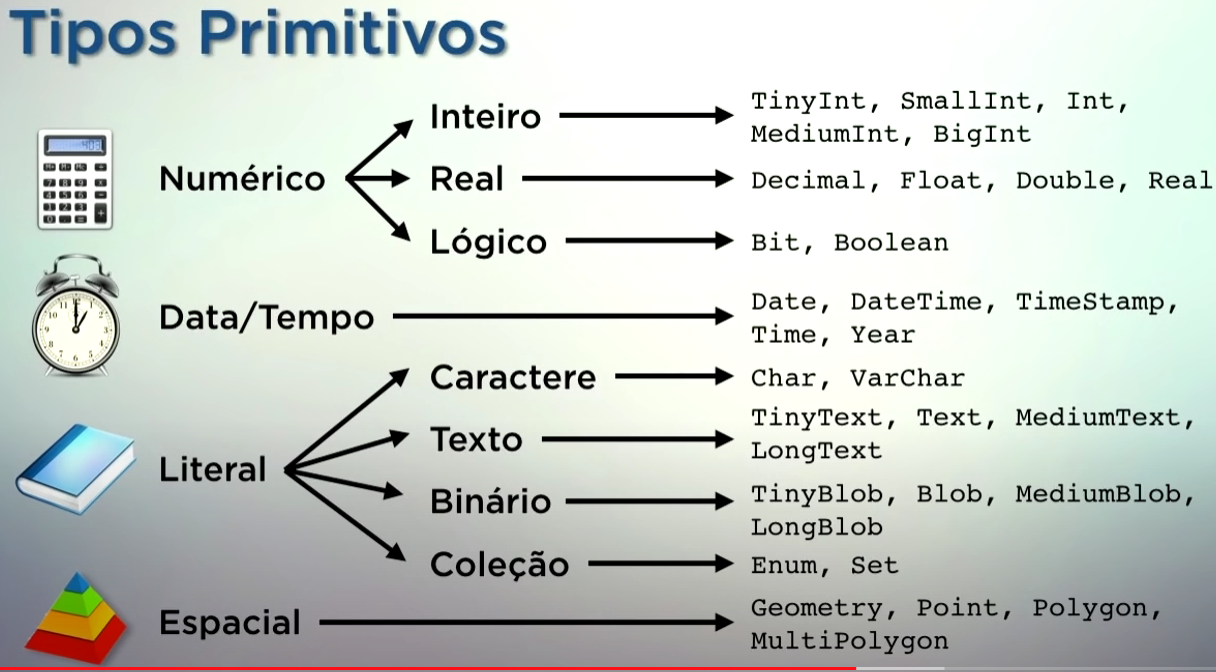
create table if not exists nomedatabela (campos conforme imagem abaixo)



* Apagar tabela

drop table if exists nomedatabela;

* Tipos

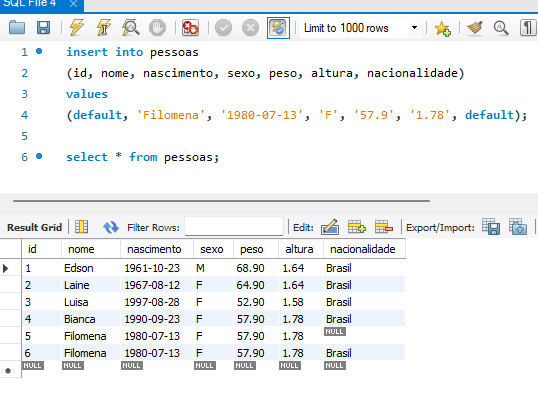


* Inserindo dados na tabela

Pode ser omitida a linha 2, desde que os dados sejam colocados na mesma ordem que estão na tabela.

Pode ser inserido dados de mais de uma pessoa, basta repetir a linha 4, separando por vírgula

Obs. Default faz a inserção pelo valor pre definido na criação da tabela.



* Alterando estrutura das tabelas

1. Adicionar uma nova coluna na tabela depois de uma outra já existente:

alter table nomedatabela add column novacoluna varchar(xx) after colunajaexistente;

1. Adicionar uma nova coluna na tabela no ínicio. (int é o tipo do campo)

alter table nomedatabela add novacoluna int first;

1. Adicionando uma **chave estrangeira** (chave primária de outra tabela)

Primeiro adiciona uma coluna com um nome qq, de preferencia um nome sugestivo.

Depois altera a coluna nova para que fique como uma chave MUL (múltipla), para isto faz a referência da coluna nova à coluna da outra tabela.

alter table gafanhotos add column cursopreferido int;

alter table gafanhotos add foreign key (cursopreferido) references cursos(idcurso);

1. Excluir uma coluna da tabela

alter table nomedatabela drop column nomedacoluna;

1. Mudar uma propriedade de uma coluna, no exemplo abaixo foi alterado o tipo do campo

alter table nomedatabela modify column nomedacoluna varchar(xx) not null default;

1. Mudar o nome da tabela

alter table nomedatabela rename to novonomedatabela;

1. Colocar chave primária em uma coluna

alter table nomedatabela add primary key (nomedacoluna);

1. Mudar o nome da coluna

alter table nomedatabela modify column nomedacoluna novonomedacoluna varchar(xx);

1. Mudar dados na tabela

update nomedatabela set nomedacoluna='novodado', nomedacoluna = novodado where chave = 'referenciadachave' limit 1; (limit vai limitar a quantidade de linhas alteradas)

1. Apagar uma linha (registro) da tabela

delete from nomedatabela where nomedacoluna = 'referencianacoluna' limit 1;

1. Apagar TODOS os registros de uma tabela

truncate table nomedatabela;

* SELECT

\* select \* from nomedatabela; //selecione todos os registros da tabela

\* select conula1, coluna2, ... from order by coluna1, coluna2...;

//seleciona so as colunas citadas e ordena tb pelas colunas citadas

\* select \* from cursos where ano='2016' order by nome;

//seleciona todos registros onde o ano seja 2016 e ordena pelo nome.

\* select nome, descricao, ano from cursos where ano <= 2015 order by ano desc, nome asc;

//seleciona registros cujo ano seja <=2015, lista as colunas citadas e ordena conforme descrito

\* select nome, ano from cursos where ano **between** 2014 and 2016;

//seleciona os registros cujo ano esteja entre 2014 e 2016.

\* select nome, ano, descricao from cursos where ano **in** (2014, 2016) order by ano;

//seleciona os registros cujos anos sejam 2014 ou 2016.

\* select \* from cursos where carga > 35 **and** totaulas < 30;

//seleciona os registros que tem carga > 35 e totaulas < 30.

\* select \* from cursos where carga > 35 **or** totaulas < 30;

//seleciona os registros que tem carga > 35 ou totaulas < 30.

\* select \* from cursos where nome **like** 'p%'; (obs: o **not** **like** faz o inverso)

// ‘p%’ todos registros (campo nome) que começam com p

//’%p%’ todos registros (campo nome) que tem p em algum lugar

//’%p’ todos registros (campo nome) que terminam com p

//’ph%p’ todos registros (campo nome) que começam com ph e terminam com p

//’ph%p%’ todos registros (campo nome) que começam com ph e tem p em qq parte e terminam com qq outra coisa.

//’ph%p\_’ todos registros (campo nome) que começam com ph e tem p na penúltima letra.

//’p\_p%’ todos registros (campo nome) que começam com p e tem uma letra qq depois e outro p em seguida e terminam com qq coisa. (**obs**: pode usar quantos sublinhados que forem necessários, cada sublinhado representa uma letra.)

\* select **distinct** nacionalidade from gafanhotos order by nacionalidade;

//seleciona apenas um registro de cada (campo nacionalidade), mesmo que na tabela existam vários registros repetidos.

\* select **count**(\*) from cursos where carga > 40;

//conta quantos registros atendem a consulta feita.

\* select **max**(carga) from cursos; (obs: o **min** faz o inverso)

select max(totaulas) from cursos where ano = '2016';

//seleciona o maior valor da coluna, se for texto pega o último da ordem alfabética. Na segunda linha foi feito com um filtro mais específico.

\* select **sum**(totaulas) from cursos where ano = '2016';

//soma o totaulas que tiveram em 2016

\* select **avg**(totaulas) from cursos where ano = '2016';

//faz a média do totaulas que tiveram em 2016

\* select totaulas, count(\*) from cursos **group by** totaulas;

// Agrupa pelo campo totaulas e mostra a quantidade de cada agrupamento

\* select ano, count(\*) from cursos group by ano **having** count(ano) >=5 order by count(\*);

// Mostra o ano, conta quantos tem em cada ano agrupado e filtra so os que tiverem >=5.

Dentro do having so se trabalha com o campo agrupado.

\* select carga, count(\*) from cursos where ano > 2015 group by carga having carga > (select avg(carga) from cursos);

//**um select dentro do outro.** Nesse caso é feito um agrupamento pelo campo carga e so mostra os registros que estejam acima da média.

* Relacionamento entre tabelas

\* select gafanhotos.nome, cursos.nome, cursos.ano from gafanhotos **inner** **join** cursos **on** cursos.idcurso = gafanhotos.cursopreferido;

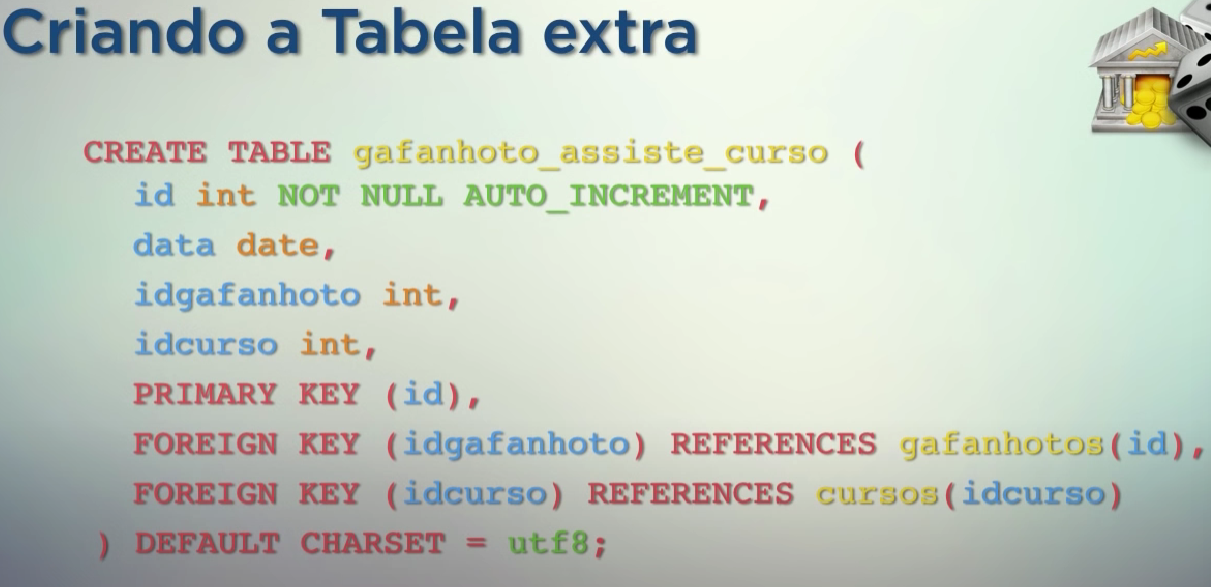
//Faz uma seleção de registros vinculando uma tabela com a outra (join) e faz o filtro (on) para mostrar apenas os registros que tenham relação entre as duas tabelas. O **inner** é facultativo.

\* select g.nome, c.nome, c.ano from gafanhotos **as** g join cursos **as** c on c.idcurso = g.cursopreferido;

//Faz o mesmo efeito que a linha anterior, neste caso foi usado o **AS** para dar **apelidos** às tabelas.

\* select g.nome, c.nome, c.ano from **gafanhotos** as g **right outer join** **cursos** as c on c.idcurso = g.cursopreferido;

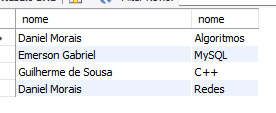
//Faz a seleção de todos os registros da tabela que está referenciada à **direita**, mesmo os que não tem relação entre as duas tabelas. Para fazer o mesmo da tabela que está à **esquerda** basta trocar o right por left. O **outer** é facultativo.



\* create table gafanhoto\_assiste\_curso (id int not null auto\_increment, data date, idgafanhoto int, idcurso int, primary key (id), foreign key (idgafanhoto) references gafanhotos(id), foreign key (idcurso) references cursos(idcurso)) default charset = utf8;

\* select g.nome, c.nome from gafanhotos as g join gafanhoto\_assiste\_curso as a on g.id = a.idgafanhoto join cursos as c on c.idcurso = a.idcurso;

//vincula as 3 tabelas e mostra os campos abaixo, resultado do cruzamento dos dados.



* ACID

\* Atomicidade = toda tarefa tem que ser concluída para que seja efetivada.

\* Consistência = O BD tem de continuar consistente após a transação (ex.: não posso apagar o registro de uma tabela que tem um a referência em outra, que esteja relacionada com uma chave estrangeira)

\* Isolamento = Duas transações acontecendo ao mesmo tempo devem ser executadas como se estivessem sendo feitas de forma isolada.

\* Durabilidade = uma transação tem que ser durável.