**Desenvolvimento de Banco de Dados**

## Tipos Numéricos:

|  |
| --- |
| • SMALLINT [(M)] [UNSIGNED] [ZEROFILL] - Inteiro no intervalo de -32768 a 32767. O intervalo sem sinal é de 0 a 65535.  • INT [(M)] [UNSIGNED] [ZEROFILL] - Inteiro no intervalo de -21473648 a 21473647. O intervalo sem sinal é de 0 a 42949695.  • BIGINT [(M)] [UNSIGNED] [ZEROFILL] - Intervalo de -922337203685808 a 922337205807.  • FLOAT [(M, D)] [UNSIGNED] [ZEROFILL] - Ponto flutuante, de precisão simples.  • DOUBLE [(M, D)] [UNSIGNED] [ZEROFILL] - Ponto flutuante de precisão dupla. |

## Tipo Data e hora:

|  |
| --- |
| • DATE - MySQL exibe no formato 'YYYY-MM-DD’ Permite números ou strings  • DATETIME - Combinação de data e hora. Formato 'AAAA-MM-DD HH: MM: SS [fracao]’  • TIMESTAMP, TIME , YEAR [4] - Os valores de TIMESTAMP são armazenados como o número de segundos desde a época ('1970-01-01 00:00:00' UTC).  • TIME [(fsp)] - O MySQL exibe valores “TIME” no formato 'HH: MM: SS [fração]';  • YEAR [(4)] - Um ano no formato de quatro dígitos. - Exibe valores YEAR (ano) no formato YYYY |

## Tipo Texto:

|  |
| --- |
| • CHAR [(M)]- [CHARACTER SET charset\_name] - Uma cadeia de comprimento fixo que é sempre preenchida à direita com espaços para o comprimento especificado quando armazenada.  • VARCHAR (M)- [CHARACTER SET charset\_name] - Cadeia de comprimento variável.  • ENUM- ('valor1', 'valor2', ...) [CHARACTER SET charset\_name] - Objeto de string que pode ter apenas um valor, escolhido na lista de valores 'valor1', 'valor2', [...], NULO, ou vazio.  Armazenado como inteiro pelo banco.  • TEXT- TEXTO/DESCRIÇÃO  • BINARY [(M)]- O tipo BINARY é semelhante ao tipo CHAR, mas armazena cadeias de bytes binários em vez de cadeias de caracteres não binários.  • VARBINARY (M)- O tipo VARBINARY é semelhante ao tipo VARCHAR, mas armazena cadeias de bytes binários em vez de cadeias de caracteres não binários. M representa o comprimento máximo da coluna em bytes.  • TINYBLOB- Uma coluna “BLOB” com um comprimento máximo de 255 (28 - 1) bytes. Cada valor TINYBLOB é armazenado usando um prefixo de 1 byte.  • BLOB [(M)]- Uma coluna “BLOB” com um comprimento máximo de 65.535 (216- 1) bytes. Cada valor BLOB é armazenado usando um prefixo de 2 bytes |

# Criar um Banco de Dados

|  |
| --- |
| create database cadastro  default character set utf8mb4  default collate utf8mb4\_general\_ci; |

## Criando uma tabela

|  |
| --- |
| create table if not exists pessoas( # criar tabela se não existir  id int not null auto\_increment, # (1, 2, 3...)  nome varchar(10) not null, # não pode ser nulo (obrigatório)  nascimento date,  sexo enum ('F', 'M'),  altura decimal (3,2), # (default)- se ninguém colocar nada, padrão o Brasil  nacionalidade varchar (30) default 'Brasil'  )default charset = utf8mb4; #designa um conjunto de símbolos |

## Inserindo dados na tabela

|  |
| --- |
| insert into pessoas  (nome, nascimento, sexo, peso, altura, nacionalidade)  value # valores  ('1', 'João', '1998-20-04', 'M', '1.70', 'Japão'), # sempre entre parentes simples  ('2', 'Maria', '1999-03-12', 'F', '1.60', 'Portugal'); #decimal com ponto, nao virgula |

## Mostrar todas a tabela

|  |
| --- |
| select \* from pessoas; |

## Para adicionar nova coluna

|  |
| --- |
| alter table pessoas  add column profissão varchar(10);  add column profissao varchar(10) after nome; # after - depois do nome  add column codigo int first; # first - primeira posição |

## Para alterar estrutura da definição

|  |
| --- |
| alter table pessoas #modificar o tipo primitivo do campo  modify column profissao varchar(20); e todos os constraints (int(4) -> int(5)) |

## Para renomear o nome da coluna

|  |
| --- |
| alter table pessoas  change profissao prof varchar(20); #mudar o nome antigo(ano), para o novo (anos) |

## Para renomear a tabela inteira

|  |
| --- |
| alter table pessoas  rename to gafanhotos; |

## Criando outro Tabela

|  |
| --- |
| create table if not exists cursos( # se ele não existir  idcurso int,  nome varchar (30) not null unique,  descricao text,  carga int unsigned, # (unsigned) não aceita números negativos  totaulas int unsigned  ano year default '20'  )default charset = utf8mb4; |

## Colocando o id como primary key

|  |
| --- |
| alter table cursos  add primary key (idcurso); |

## Inserir dados na tabela

|  |
| --- |
| insert into cursos  (idcurso, nome, descrição, carga, totaulas, ano)  values  ('1', 'Algoritmo', 'Logica de Programação', '20', '15', '2014'),  ('2', 'Photoshop', 'Dicas de Photoshop', '10', '8', '2014'),  ('3', 'PGP', 'curso de PHP', '40', '20', '2010'),  ('4', 'Jarva', 'Linguagem Java', '10', '29', '2000'),  ('5', 'MySQL', 'Bancos de Dados', '30', '15', '2016'),  ('6', 'Word', 'Curso completo de Word', '40', '30', '2016'),  ('7', 'Ingles', 'Curso completo de Ingles', '30', '20', '2018'),  ('8', 'HTML4','curso HTML5', '37', '10', '2019'),  ('9', 'CSS3', 'Linguagem Css3', '40', '20', '2020'),  ('10', 'JS', 'Linguagem JavaScript', '40', '28', '2020'); |

## Alterando campos

|  |
| --- |
| update cursos # update = atualiza  set nome = 'HTML5' # set = configura  where idcurso = 8; # where = onde  update cursos  set nome = 'PHP', ano ‘2015’ # ja adicionando os nomes corretos  where idcurso = 3; #mudando 2 colunas ao mesmo tempo  update cursos  set nome = ‘Java’, , carga = '40', ano = '2015'  where idcurso = 4  limit = 1; # limit = segurança, alterar somente o 1º |

## Deletar uma linha da tabela

|  |
| --- |
| delete from cursos  where idcurso = 2; # vai deletar o id 2  delete from cursos  where ano = '2016' # vai deletar as linhas q tem o ano 2016  limit 3; # com um limite de apagar apenas os 3 primeiros |

## Apagar dados da tabela, mas não a estrutura

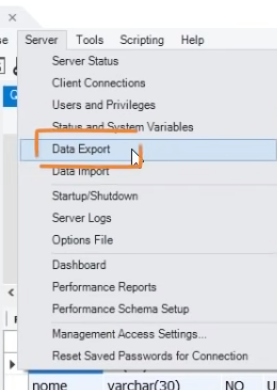
|  |
| --- |
| truncate table cursos; |

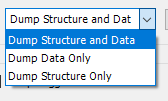
## Para excluir a tabela toda

|  |
| --- |
| drop table if not exists cursos; |

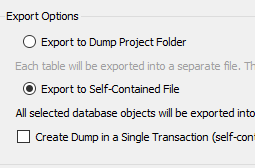
## Gerenciamento de copias de segurança (BACKUP)

Server > Date Export > selecione o BD > seleciona as tabelas



Gerar Dump a estrutura e os dados

Tem 2 opçaos de exportar, do Folder inteiro ou num arquivo único

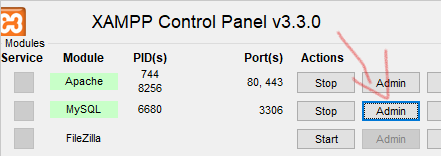
Selecionar:

 ou Adicionar declaração DROP DATABASE IF EXISTS no phpMyAdmin

Fica salvo numa pasta dumps

## Colocando em outro servidor

Server > Date Inport > import from self-contained file

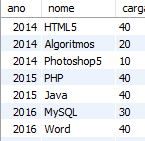
Seleciona o dump > Start Import

Para acessar o PHPMyAdmin > no XAMPP

## **Aula 11**

## **Order by** (as linhas serão atualizadas na ordem específica)

|  |
| --- |
| select \* from cursos  order by nome; #os nomes vão ficar na  order by nome asc; ordem crescente (A - Z)  ou  order by nome desc; # nomes vão ficar na decrescente (Z - A) |



## **Select** (filtrar colunas)

|  |
| --- |
| select ano, nome, carga from cursos; # vai mostrar somente estes 3 campos  order by nome;  ou  order by ano, nome; # ano vai ficar na ordem crescente em seguida o nome |

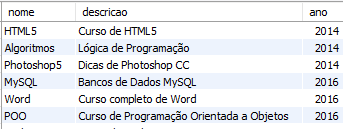
## **Select** (filtrar linhas)

|  |
| --- |
| select \* from cursos  where ano = ‘2016’ # vai mostrar somente os cursos de 2016  select nome, ano from cursos  order by nome # com nomes na ordem crescente  where ano <= ‘2015’ ; # onde o ano é menor ou igual a 2015 |



## **Between - and** (entre uma coisa e outra)

|  |
| --- |
| select nome, ano from cursos  where ano between '2014' and ‘2016’ # vai mostrar o ano entre 2014 e 2016  order by nome,  ou  order by ano desc, nome asc; |



## **In** (selecionar valores específicos)

|  |
| --- |
| select nome, descrição from cursos  where ano in ('2014', '2016'), #só vai mostrar os cursos desses anos  ou  select nome, carga, totaulas from cursos  where carga > 35 and totaulas < 30,  #mostrando carga maior que 35, e totaualas menor que 30 |

## **Like** (%)

|  |
| --- |
|  |

## **Distinct** (não repete os mesmos nomes, ano..)

|  |
| --- |
| select distinct nacionalidade from gafanhotos; 🡪    select distinct carga from cursos; 🡪 |

## Agregações **Count** (mostra a quantidade total)

|  |
| --- |
| select count(\*) from cursos; # vai mostrar a quantidade de cursos  ou  select count(\*) from cursos # vai contar quantos cursos  where carga > 40; # que a carga é maior que 40 |

## **Max** (o máximo dentro da tabela)

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

## **Min** (o mínimo dentro da tabela)

|  |
| --- |
| select min(cargas) from cursos;  ou  select nome, min(carga) from cursos; 🡪 |

## **Sum** (vai somar)

|  |
| --- |
| select sum(totaulas) from cursos;  ou  select sum(totaulas) from cursos where ano = '2016'; |

## **AVG** (tirar a média)

|  |
| --- |
| select avg(totaulas) from cursos; |

## **Group by** (agrupando os valores iguais, os resultados das seleções podem ser

## organizados em grupos, baseados no conteúdo existente em uma ou mais colunas)

|  |
| --- |
| select carga, count(\*) from cursos  group by carga #agrupar os valores iguais de carga  order by carga; #deixando a carga na ordem crescente |

## **Having** (tendo somente - dentro do group by)

|  |
| --- |
| select ano, count(\*) from cursos  group by ano  having count(nome) >= 5; #vai mostrar somente quem tem o contador maior que 5  ou  select ano, count(\*) from cursos  where totatulas > 30 #onde totaulas é maior que 30  group by ano #agrupando por ano  having ano > 2013 #mostrando somente o ano acima de 2013  order by count(\*) asc; #ordenando o total na ordem crescente |

# **Modelo Relacional**

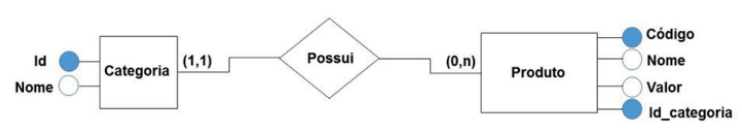


Diagrama E-R (entidades(retangulos) e relacionamento(losango)) 🡪 DER

## **Entidades**

Definem seu comportamento em relação a chaves primárias, quais eventos e como os processos de negócios acessam, modificam, excluem e incluem as tuplas.

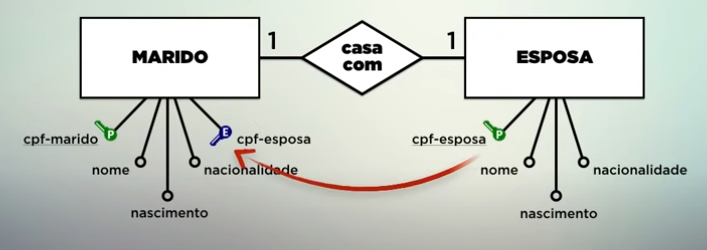
## **Atributo**

Atributo é tudo o que se pode relacionar como próprio da entidade (propriedade) que de alguma maneira a qualifique e a distinga de outras, estes podem ser classificados e identificados.

## **Relacionamento**

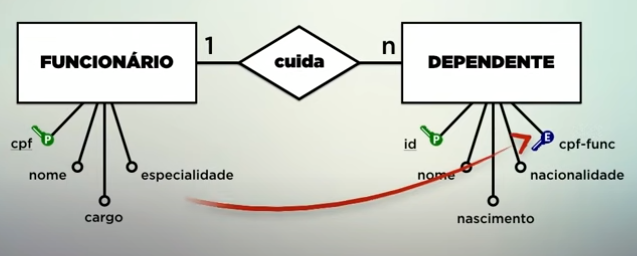
Relacionamento é a relação existente entre entidades, isto é a ligação lógica entre duas entidades que representa uma regra ou restrição de negócio.

* **Cardinalidade**: Indica a quantidade de ocorrências de determinado relacionamento.
  + N : várias vezes
  + 1 : apenas uma vez
  + 0: não acontece
* **Relacionamento 1-1**
  + Um para um, define que um item de uma entidade só poderá se relacionar com um item de outra entidade. Por exemplo, supondo que temos as entidades Cliente e Endereço que se relacionam de forma 1-1, um cliente só poderá possuir um endereço, que também só pode estar relacionado a um cliente.



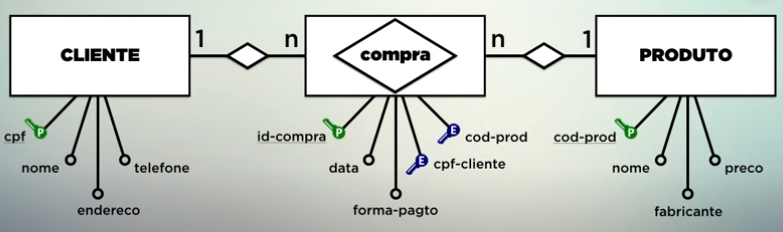
Lado esquerdo normalmente é o dominante. Chave primaria vira chave estrangeira na entidade marido

* **Relacionamentos 1:N**
  + Um para muitos, determina que um item de uma tabela pode se relacionar com vários itens de uma outra tabela. Continuando o exemplo acima, podemos determinar que um cliente pode realizar diversos pedidos em um restaurante, porém cada pedido só pode possuir um cliente atrelado a ele.



Pega a chave primaria do lado 1, e joga para muito como chave estrangeira.

* **Relacionamentos N:N**
  + Muitos-para-muitos, ocorre quando várias entidades de um lado de um relacionamento podem se associar a várias entidades do outro lado. Imagine um cenário em que alunos podem se inscrever em várias disciplinas e, ao mesmo tempo, uma disciplina é frequentada por vários alunos.



* O relacionamento vira uma entidade com os (n) junto a ele;
* Adicionando mais dois relacionamentos entre eles;
* Colocando pra 1 em cada entidade;
* Voltando para regra 1:n
  + Pega a chave primaria do lado 1, e joga para muito como chave estrangeira.

## Adicionar uma nova coluna

|  |
| --- |
| alter table gafanhotos  add column cod\_cursospreferido int; |

## Adicionando **chave estrangeira**

|  |
| --- |
| alter table gafanhotos  add foreign key (cod\_cursopreferido)  references cursos(idcurso); |

OBS: quando se cria chave estrangeira, ela tem que ser do mesmo tipo primitivo, da

chave primária pela qual vai se referir.

## Adicionando um curso preferido para cada gafanhoto

|  |
| --- |
| update gafanhotos set cod\_cursopreferido = '6'  where id = 1; #o aluno 1 vai preferir o curso 6 |

OBS: nao pode apagar(deletar) mais o curso 6, pois tem relação, vai dar erro.

Para facilitar colocando um curso pra cada, mexer na tabela, dando **apply.**

## **Cláusula as**

|  |
| --- |
| Renomear um campo, sendo atributo ou tabela:  nome-antigo as nome-novo  Ou para transformar em apelido:  cursos as c, gafanhotos as g |

## 

## **Inner Join** (junção - unir duas ou mais tabelas. Só vai mostrar a tabela que tem uma relação com a outra)

|  |
| --- |
| select g.nome, g.cod\_cursopreferido, c.nome, c.ano  from gafanhotos as g join cursos as c #juntando o gafanhotos e cursos  on c.idcurso = g.cod\_cursopreferido #fazendo a relação entre a chave primária com a estrangeira  order by g.nome; |

## **Left Join** (é um tipo de junção externa)

|  |
| --- |
| select g.nome, g.cod\_cursopreferido, c.nome, c.ano  from gafanhotos as g **left** join cursos as c #vai aparecer todos os  on c.idcurso = g.cod\_cursopreferido gafanhotos, ate os q n tem  order by g.nome; preferência de nenhum curso |

OBS: as linhas da tabela da esquerda são projetadas na seleção juntamente

com as linhas não combinadas da tabela da direita. Ou seja, como resultado

dessaseleção, algumas linhas em que não haja relacionamento entre as tabelas

da esquerda para a direita retornarão o valor nulo (NULL).

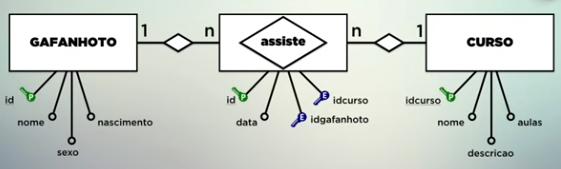
## **Right Join** (é um tipo de junção externa)

|  |
| --- |
| select g.nome, g.cod\_cursopreferido, c.nome, c.ano  from gafanhotos as g **right** join cursos as c #vai aparecer todos os  on c.idcurso = g.cod\_cursopreferido cursos inclusive os que  order by g.nome; nenhum gafanhoto prefere |

OBS: similar ao comando LEFT JOIN, com o comando RIGHT JOIN as linhas da

tabela da direita são projetadas na seleção juntamente com as linhas não

combinadas da tabela da esquerda.

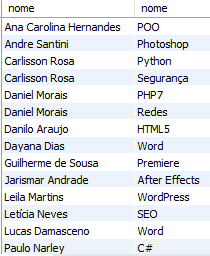


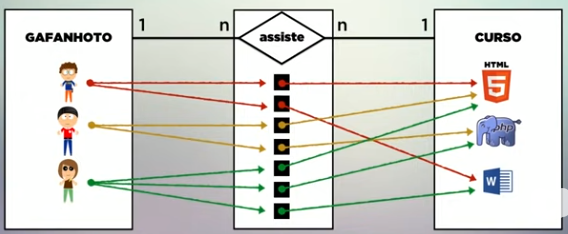
## Vamos criar uma nota tabela

|  |
| --- |
| create table gafannhoto\_assiste\_cursos(  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 = utf8mb4; |

## Inserindo dados

|  |
| --- |
| insert into gafanhoto\_assiste\_cursos values  ('default', '2014-03-04', '1', '29'); |

****Para facilitar colocando um curso para cada, mexer na tabela, dando **apply.**



## **Inner Join** (juntando as 3 tabelas)

|  |
| --- |
| select g.nome, c.nome from gafanhotos as g #mostrando apenas os nomes do gafanhoto e do curso  join gafanhoto\_assiste\_cursos as a #juntar o gafanhoto com gafanhoto\_assiste\_curso  on g.id = a.idgafanhoto #juntar a chave primária com estrangeira  join cursos c #juntar o gafanhoto com cursos  on c.idcurso = a.idcurso #juntar a chave primária com estrangeira  order by g.nome; |