***Curso de Banco de Dados***

***( SQL e NoSQL )***

***Sessão 2 - Introdução ao Banco de Dados.***

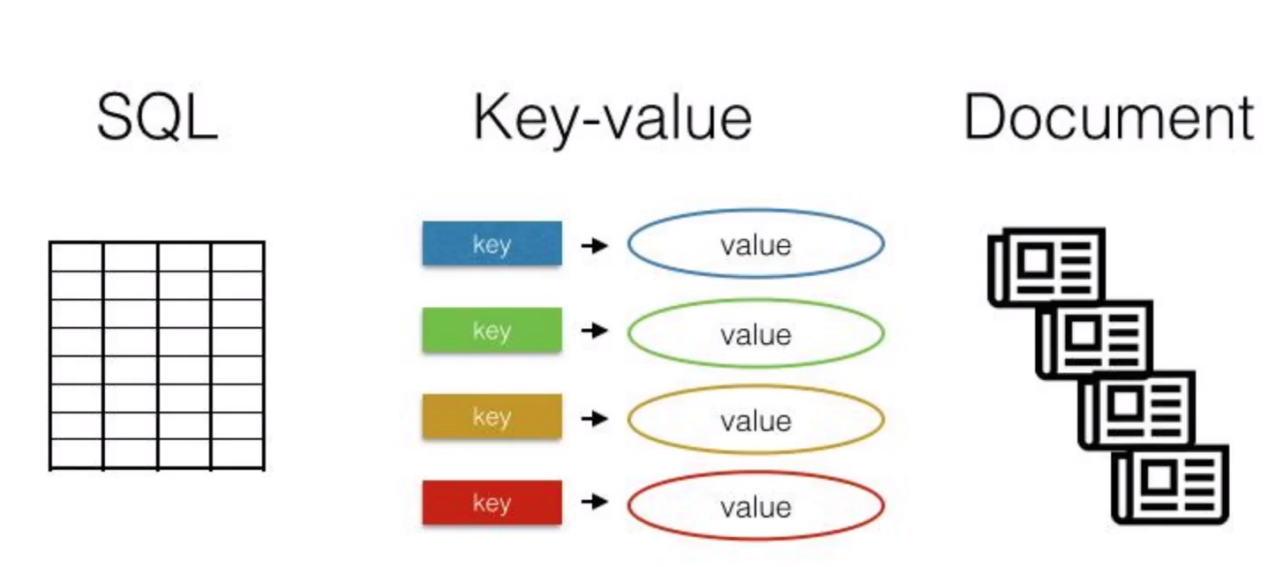
***Banco de Dados:***

**Dados →** É um elemento bruto geralmente proveniente de uma entrada do usuário ou de outros sistemas.

**Informação →** Elemento processado ( lapidado ) constuído de Dados.

******

***Banco de Dados Relacionais x Não Relacionais.***



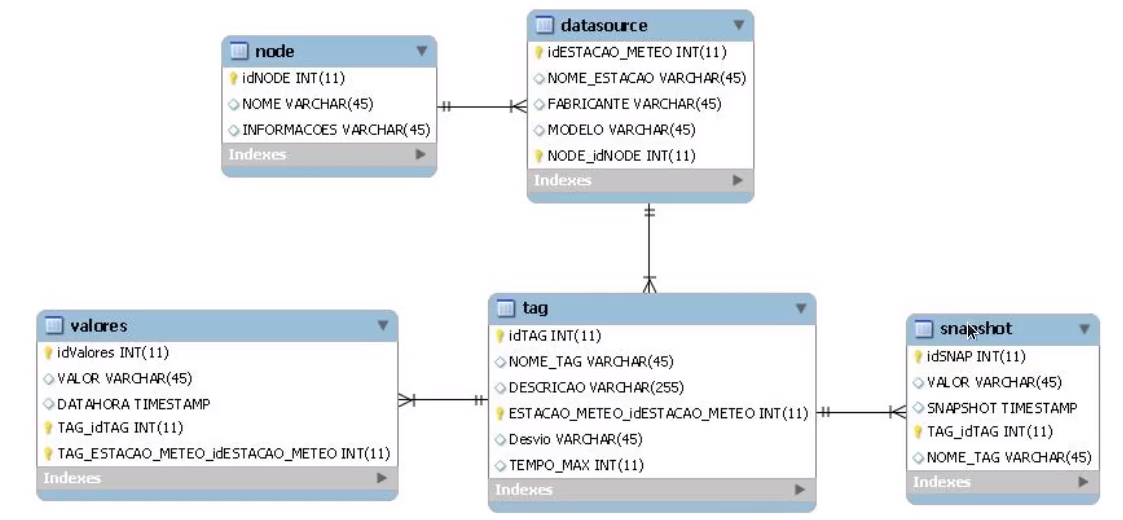
Os Banco de Dados podem trabalhar em forma de tabelas, em chaves / valor e em documentos.

***Banco de Dados Relacionais →*** *Foi criado por Edgar Frank Codd em 1970 e é mais conhecido como Linguagem SQL ( Structured Query Language = Linguagem de Consulta Estruturada ).*

***Pontos fortes →*** Os SGBDRs ( Sistemas Gerenciadores de Banco de Dados ), oferecem ao usuário um serviço completo e eficaz em seu uso.

Exemplo: Segurança; Recuperação de Falhas;; Controle de Transações, Otimização de Consultas; Processos de Validação, Verificação; Garantias de Integridade dos Dados.

**Pontos fracos →** É difícil trabalhar com demanda de aumento de dados muito alto, e em sistemas distribuídos para muitas pessoas com particionamento de dados.



*Exemplos de Banco de Dados Não Relacionais:*

**

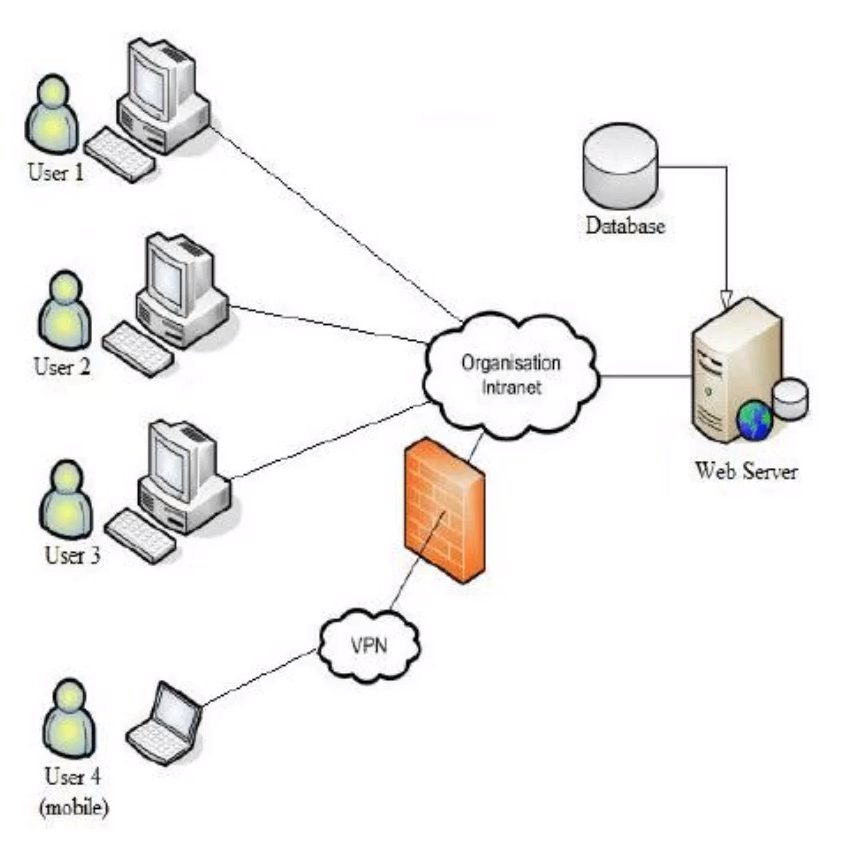
***Banco de Dados Não Relacionais →*** Surgiu em 1988, no começo foi baseado na Arquitetura Relacional, porém sem oferecer uma interface da Linguagem SQL, assim posteriormente tomando um rumo próprio e se chamando NoSQL ( Not Only SQL = Não Apenas SQL ).

É um modelo que veio para agilizara a forma de trabalhar com os dados, sendo possível trabalhar com várias pessoas particionando dados, e em aumento de grande quantidade de dados.

*Exemplos de Banco de Dados Relacionais:*

**

***Arquitetura Cliente - Servidor.***



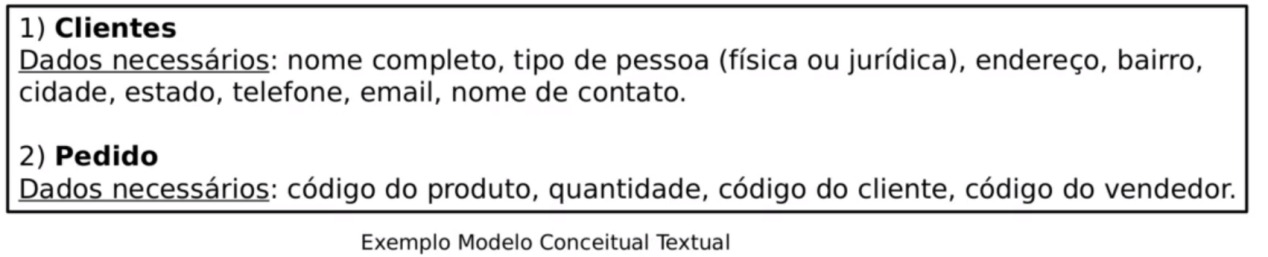
***Sessão 3 - Modelagem dos Dados.***

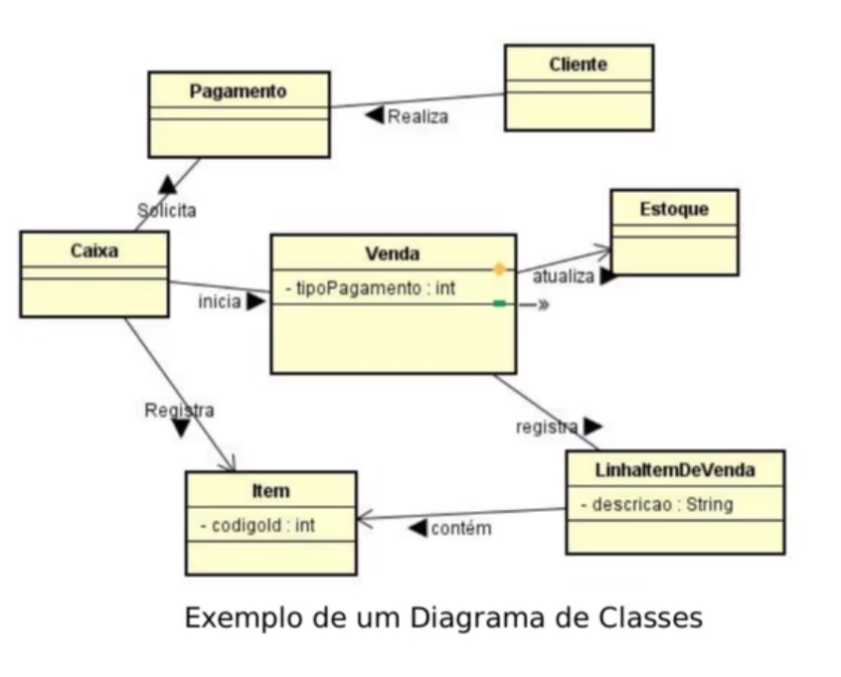
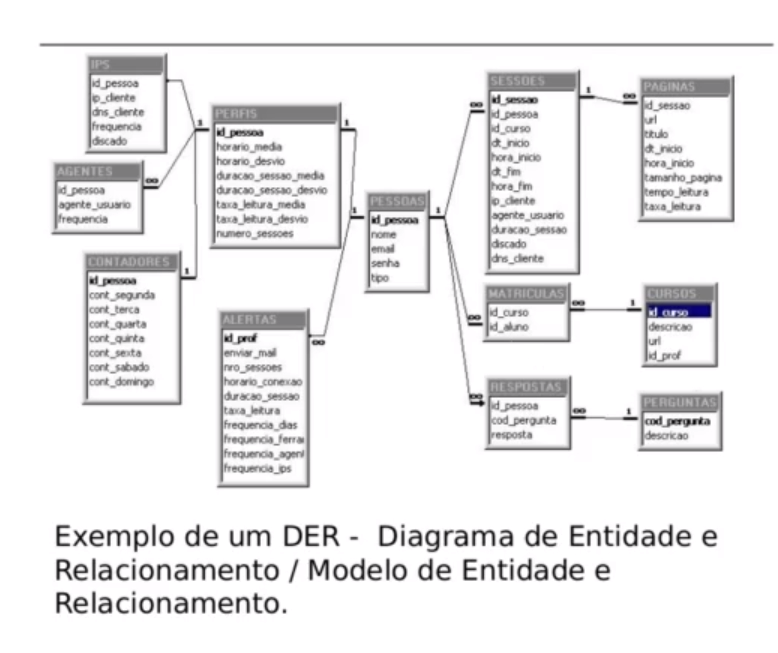
***Modelagem Conceitual, Lógica e Física.***

São modelos que descrevem a estrutura de um Banco de Dados. Servem como parte importante para a documentação dos sistemas que serão feito, e também servem como documentação para o cliente que contratar o serviço da desenvolvedora.

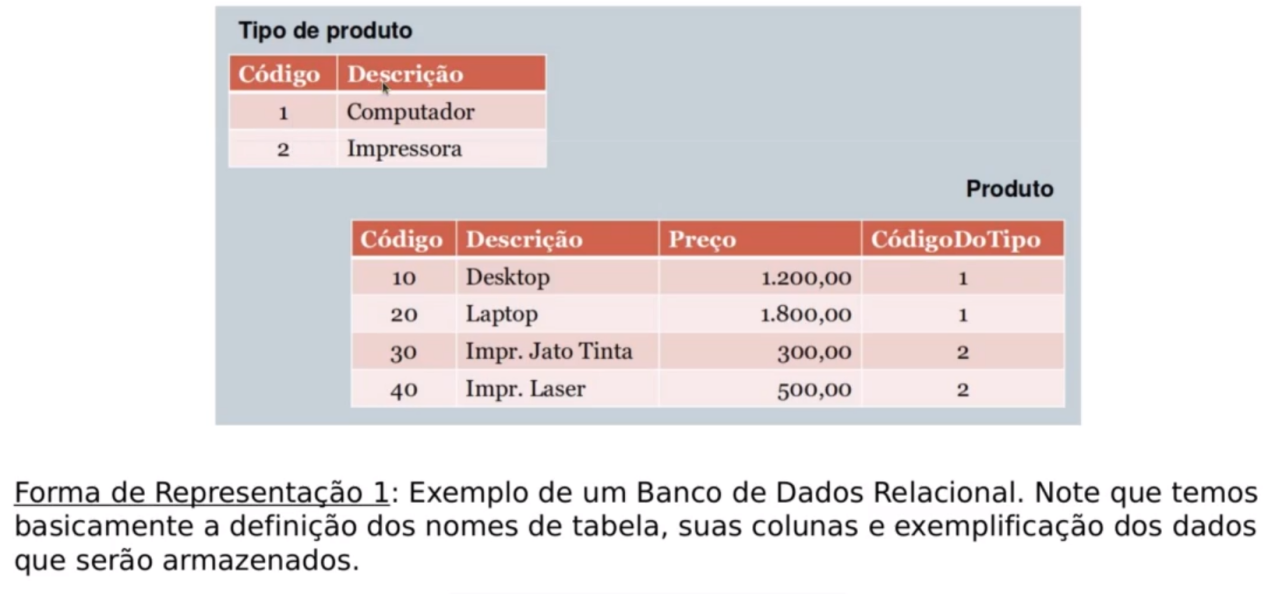
***Modelo Conceitual →*** *É de Alto Nível pois está perto do usuário final. É de responsabilidade do Analista de Sistema adquiri-lo em contato com o usuário, fazendo uma abstração dos dados mais importantes.*

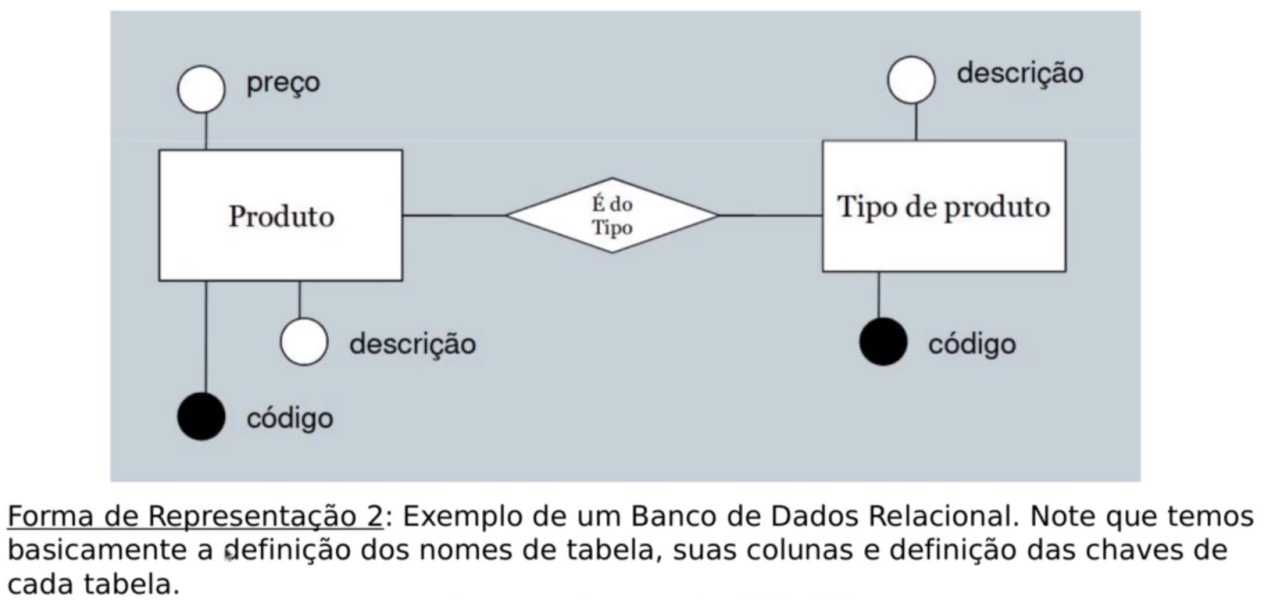
*O documento pode ser elaborado de forma textual, e em dois tipos de diagramas, sendo o Diagrama de Entidade e Relacionamento e o Diagrama de Classes.*





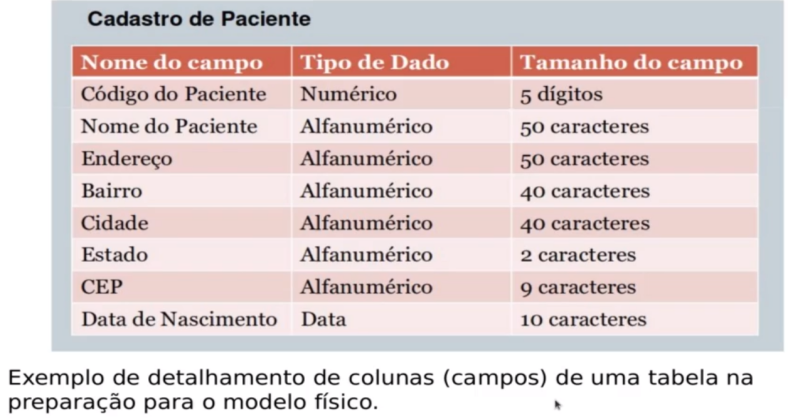
**Modelo Lógico →** Descreve como os dados serão armazenados e qual tecnologia será usada, sendo a relacional ou a não relacional.

****

****

**Modelo Físico →** É o modelo de implementação que descreve por meio de uma linguagem que pode ser a SQL, como será feita a armazenagem dos dados, e qual o SGBD será utilizado.

Pode ser feito em modo de tabela ou até mesmo com a Linguagem SQL.



***Outros Conceitos Essenciais.***

**Entidade ou Produto →** Pode ser um objeto ou um evento.

Exemplo: Aluno, carro, produto, vendedor, venda e etc.

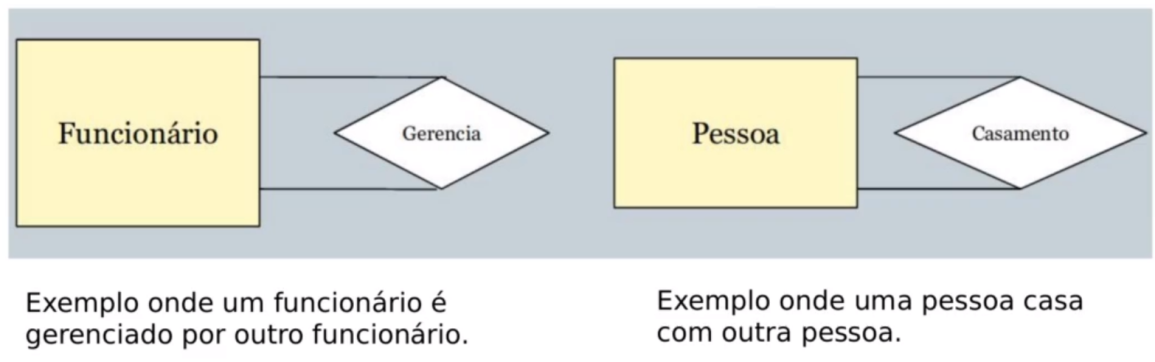
**Atributo →** Itens da entidade ou produto.

**Atributo Chave →** Conhecido como chave primária ou primary key ( pk ), é um campo que possui um valor único.

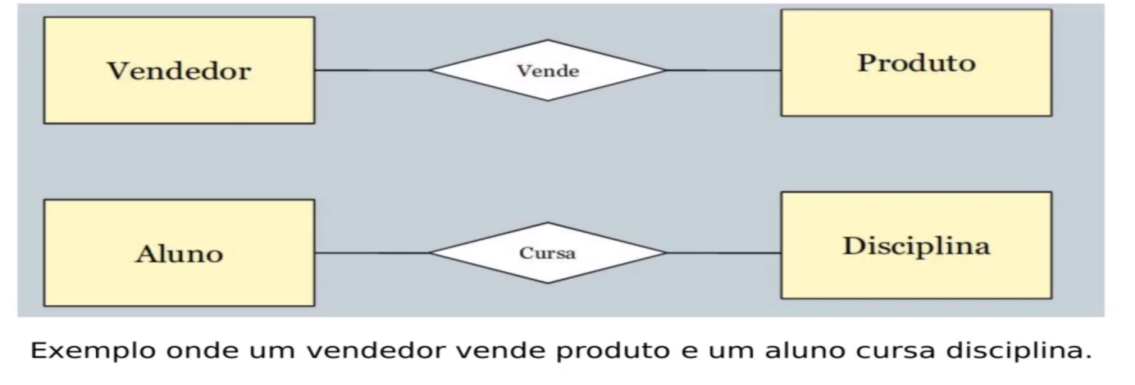
**Relacionamentos →** Fazem a ligação de uma ou mais entidades ou produtos.

**Graus de Relacionamento** → Indica quantas entidades estão ligadas em um relacionamento.

1. **Relacionamento Unário ( Grau 1 ) →** Uma entidade se relaciona com ela mesma.



1. **Relacionamento Binário ( Grau 2 ) →** Uma entidade se relaciona com outra entidade.



1. Relacionamento Ternário ( Grau 3 ) - Três entidades se inter-relacionam.



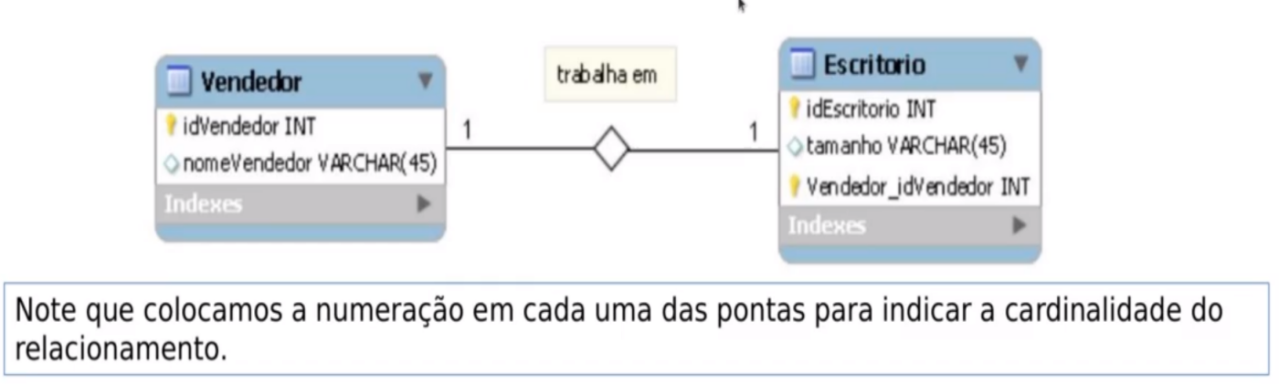
**Chave Estrangeira →** Foreign Key ( Fk ); São valores em uma entidade que representam o valor de outra.

**Cardinalidade ( Máxima ) →** Define a quantidade máxima de ações que uma entidade pode ter com outra entidade.

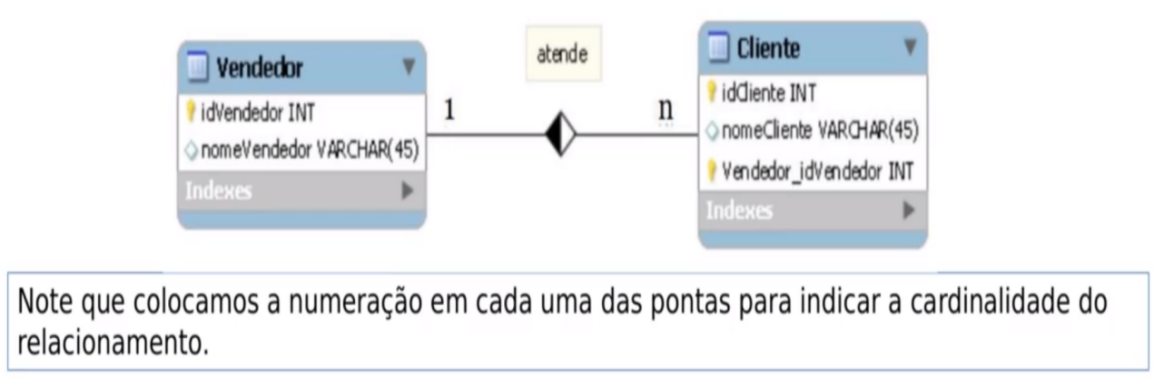
Exemplo: Um vendedor pode vender quantos produtos?

Um produto pode ser vendido por quantos vendedores?

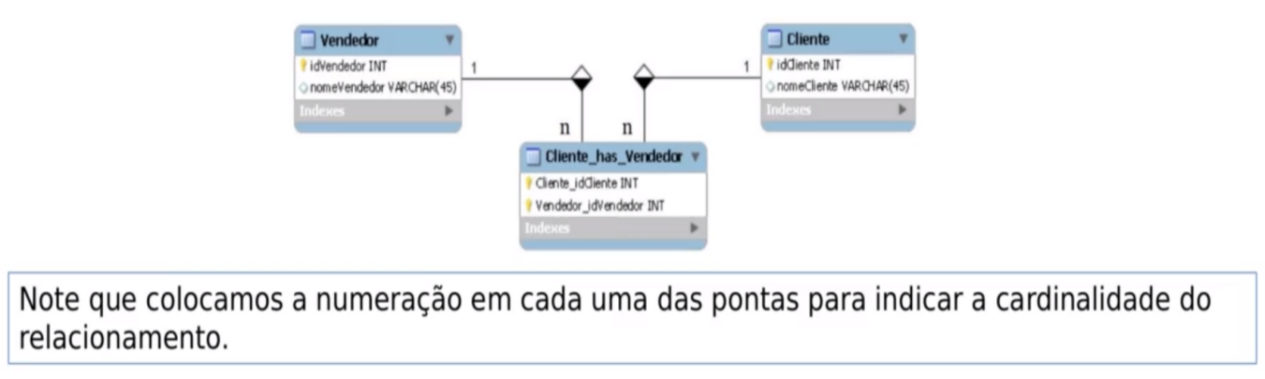
1. **Relacionamento Binário One-to-One ( Um-para-um ) ( 1:1 ) →** Indica que a entidade **A** pode se ter apenas uma ação com a entidade **B** e vice versa.



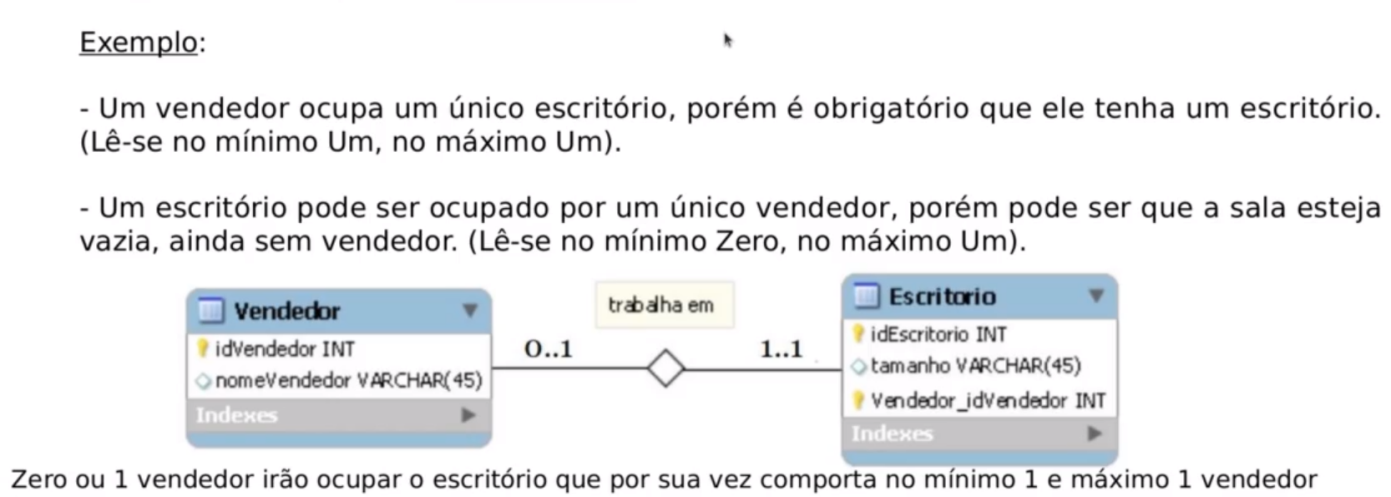
1. **Relacionamento Binário One-to-Many ( Um-para-Muitos ) ( 1:n ) →** Indica que a entidade **A** pode gerar várias ações com a entidade **B** porém o inverso não pode ocorrer.



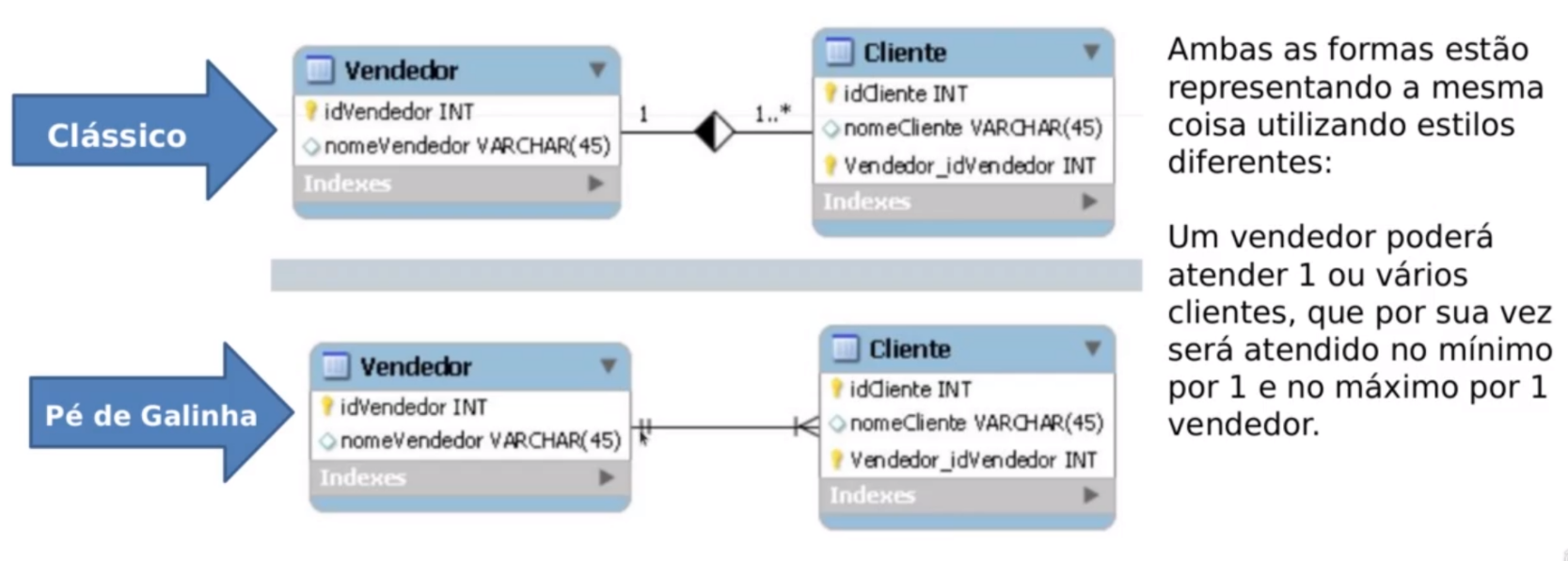
1. **Relacionamento Binário Many-to-Many ( Muitos-para-Muitos ) ( n:m ) →** Indica que a entidade **A** pode ter várias ações com a entidade **B** e vice versa.

****

**Cardinalidade ( Mínima ) →** Define a quantidade mínima de ações que uma entidade pode ter com outra entidade. São representadas por 0 e 1.



**Formas de Representação →** Os mais comuns são Clássico e o Pé de Galinha.



***Normalização de Dados.***

É o processo onde são descritas as entidades, relacionamentos, atributos, chaves primárias e chaves estrangeiras que serão gerados no banco de dados a partir da descrição dada pelo Analista de Sistemas.

Esse processo tem o objetivo de evitar irregularidades ao fazer acesso dos registros no banco de dados, ajudando também a evitar desperdícios no espaço de armazenamento, e outros benefícios.

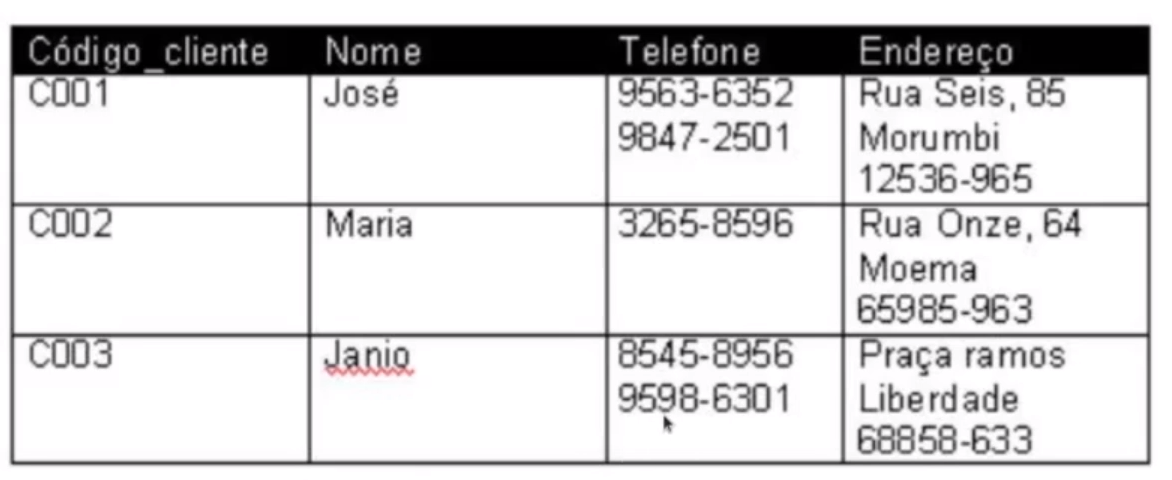
***Formas Normais.***

Para realizar esse procedimento precisamos lembrar cada etapa precisa ter a etapa anterior bem estabelecida a partir da Segunda Forma Normal.

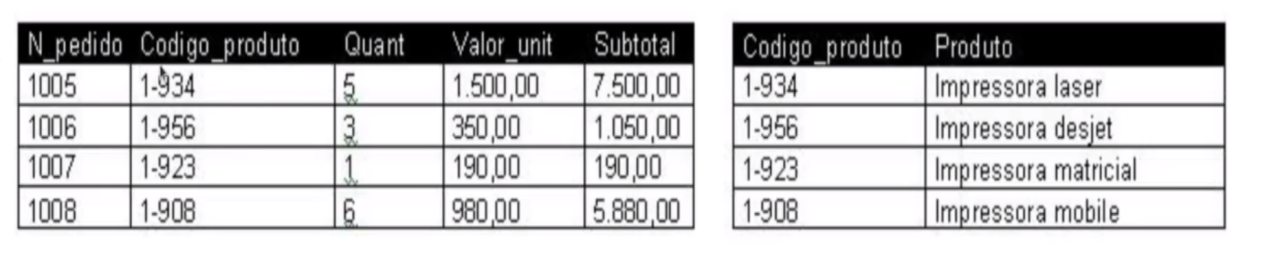
Existem 5 regras formais, porém as mais utilizadas são apenas as 3 primeiras.

**Primeira Forma Normal →** Nesse primeiro passo vamos tirar todos os valores multi valorados da tabela, evitando redundâncias de dados ( duplicidade de dados desnecessários ) e desorganização.

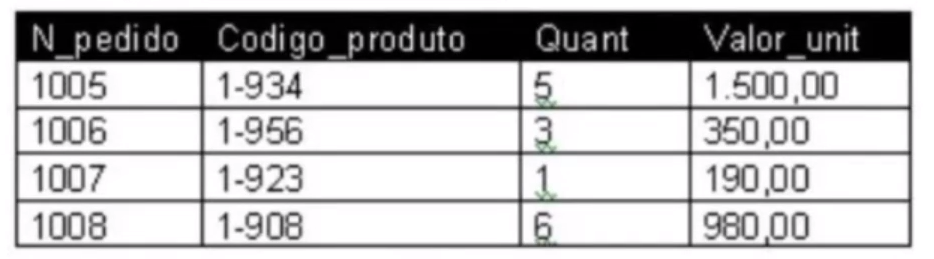
É tão comum fazer isso que os Analistas de Sistemas ao adquirir as informações já aplicam a primeira forma normal.

****

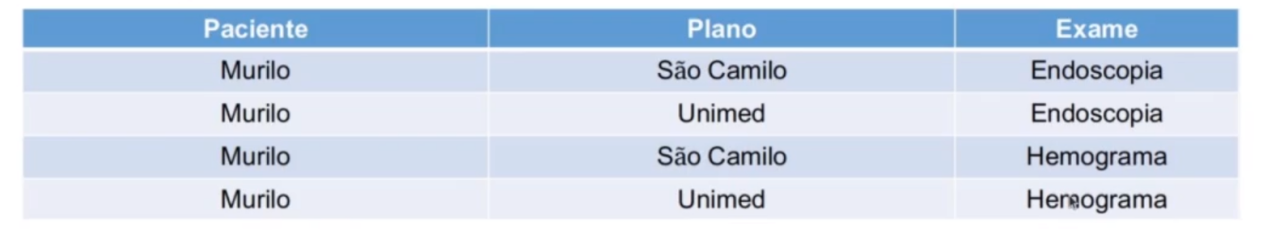
**Segunda Forma Normal →** É feita uma nova tabela e os atributos da primeira são totalmente dependentes dos atributos do primeiro campo ou seja a primeira chave primária da segunda tabela, podendo ser homologado uma chave estrangeira.

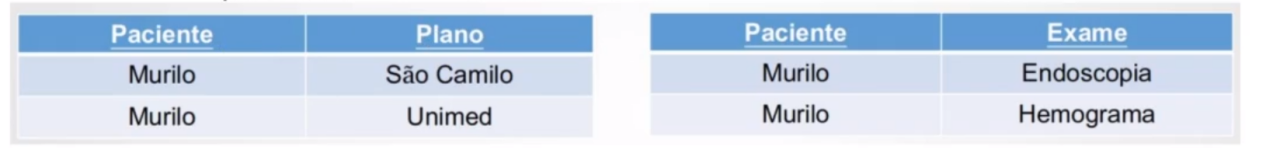
****

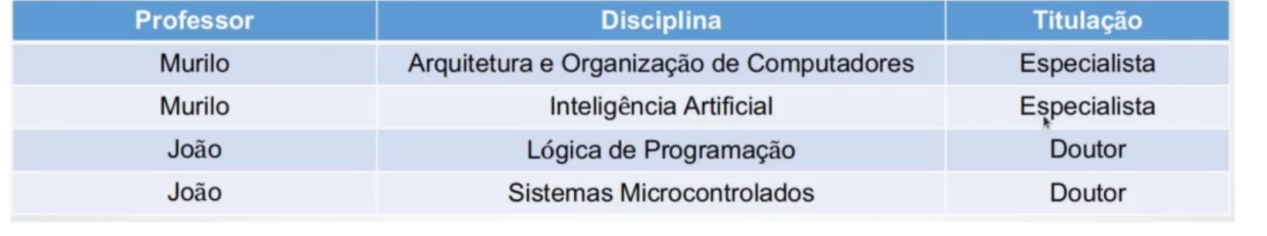
**Terceira Forma Normal →** Aqui faremos a eliminação de todos os campos que podem ter seu valor obtido através da equação de outros campos.

****

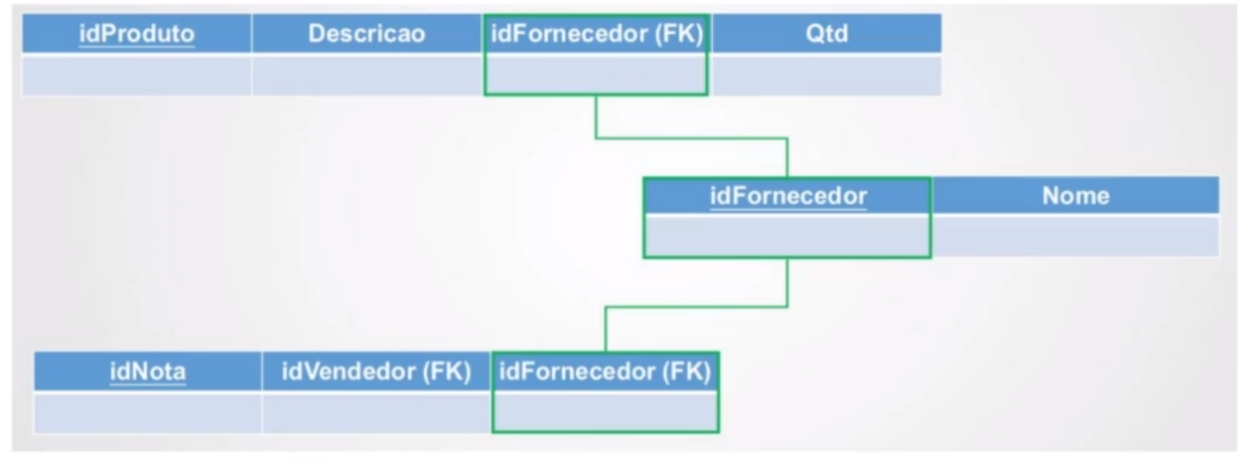
**Quarta Forma Normal →** Nesta etapa não obrigatória vamos evitar duplicidades.

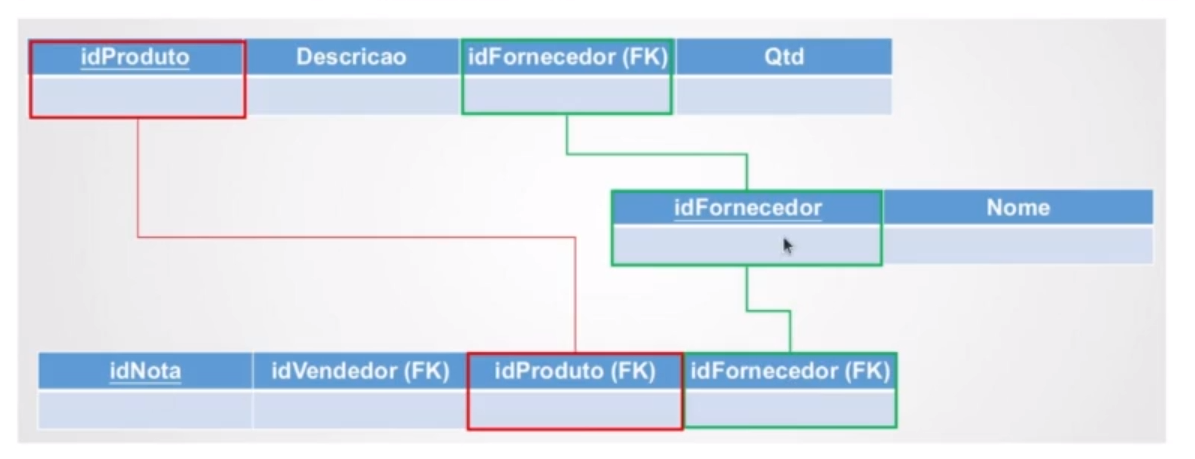
****

****

****

****

**Quarta Forma Normal →** Nesta etapa não obrigatória vamos remover atributos que são desnecessários. Ou seja... atributos que podem ser acessados por outro campo.



***Sessão 4 - Linguagem SQL ( Parte 1 ).***

**DQL - Data Query Language** - É uma Linguagem Estruturada que busca trazer de antemão o funcionamento de consultas e Bancos de Dados não Relacionais.

**Comandos:**

**SELECT** colunas\_da\_tabela **FROM** nome\_da\_tabela**;** →

Comando utilizado para fazer uma consulta na tabela, especificando os dados de colunas específicas.

**SELECT p**.nome\_da\_coluna **AS** apelido\_para\_a\_coluna, **p**.nome\_da\_coluna **AS** apelido\_para\_a\_coluna **FROM** nome\_da\_tabela **AS p;** →

Comando utilizado para dar apelidos mais simples para as colunas das tabelas no momento de consulta.

**DML - Data Manipulation Language** - É uma Linguagem que permiti inserir dados, alterar dados e deletar dados em uma tabela.

**Comandos:**

**USE** nome\_do\_banco\_de\_dados →

Comando utilizado para ter acesso ao Banco de Dados.

**INSERT -** Usado para inserir um registro a uma tabela existente.

**INSERT INTO** nome\_da\_tabela

(coluna\_da\_tabela , outra\_coluna\_da\_tabela)

**VALUES**

(dado\_da\_primeira\_coluna , dado\_da\_segunda\_coluna)**;**

→

Comando utilizado para inserir dados nos específicos campos das colunas da tabela.

**INSERT INTO** nome\_da\_tabela **VALUES (**coluna\_da\_tabela , coluna\_da\_tabela , coluna\_da\_tabela **);** →

Comando utilizado para inserir todos os campos de dados de uma tabela. Todos as colunas da tabela em dados precisam ser preenchidas.

**UPDATE -** Usado para alterar valores de dados em um ou mais registros de uma tabela.

**UPDATE** nome\_da\_tabel

**SET** coluna\_que\_será\_alterada **=** ‘novo\_da\_dado’

**WHERE** código\_id **=** numero\_do\_código\_id**;** →

Comando utilizado para alterar um dado específico da tabela.

**DELETE -** Usado para remover registros de uma tabela.

**DELETE FROM** nome\_da\_tabela **WHERE** código\_id **=** número\_do\_código\_id**;** →

Comando utilizado para apagar uma linha da específica da tabela.

**DDL - Data Definition Language** - É uma Linguagem que permiti criar, alterar, e excluir, banco de dados, tabelas, e elementos associados como índice e visão.

**CREATE -** Usado para criar um banco de dados, tabela e outros objetos em um banco de dados;

**CREATE DATABASE** nome\_do\_banco\_de\_dados**;** →

Comando utilizado para criar um novo Banco de Dados.

**CREATE TABLE** nome\_da\_tabela **(**

coluna\_da\_tabela especificações\_da\_coluna

coluna\_da\_tabela especificações\_da\_coluna

coluna\_da\_tabela especificações\_da\_coluna

coluna\_da\_tabela especificações\_da\_coluna

**);** →

Comando utilizado para criar uma nova tabela no Banco de Dados.

**ALTER -** Usado para alterar a estrutura de tabelas ou outro objeto em um banco de dados.

**ALTER TABLE** nome\_da\_tabela

**ADD** nova\_coluna\_da\_tabela especificações\_da\_nova\_coluna**;** →

Comando utilizado para adicionar uma nova coluna em uma tabela já existente no Banco de Dados.

**DROP -** Usado para apagar bancos de dados, tabelas e outros objetos.

**DROP TABLE** nome\_da\_tabela →

Comando utilizado para a pagar a tabela do Banco de Dados.

**DROP DATABASE** nome\_do\_banco\_de\_dados →

Comando utilizado para apagar um Banco de Dados.

**DCL - Data Control Language** - É uma Linguagem de controle de dados.

**GRANT -** Usado para autorizar um usuário a executar ou setar operações no banco de dados.

**GRANT SELECT ON** nome\_da\_tabela **TO** nome\_do\_usuário**;** →

Comando utilizado para dar permissão a usuários fazerem consultas no Banco de Dados.

**REVOKE -** Usado para remover ou restringir a capacidade de um usuário de executar operações.

**REVOKE CREATE TABLE FROM** nome\_do\_usuário**;** →

Comando utilizado para tirar a permissão de um usuário específico para criar tabelas no Banco de Dados.

**DTL - Data Transaction Language -** É uma Linguagem usada para permitir que transações sejam feitas de forma correta no Banco de Dados.

**BEGIN / COMMIT -** Usado para marcar e finalizar uma transação que pode ser completada ou não, gravando os dados dentro da tabela no Banco de Dados.

**CREATE TABLE** nome\_da\_tabela **(**

coluna\_da\_tabela especificações\_da\_coluna

coluna\_da\_tabela especificações\_da\_coluna

coluna\_da\_tabela especificações\_da\_coluna

coluna\_da\_tabela especificações\_da\_coluna

**);**

**BEGIN TRANSACTION;** -- começamos a transação

**INSERT INTO** nome\_da\_tabela **VALUES (**dado\_da\_tabela**);**

**INSERT INTO** nome\_da\_tabela **VALUES (**dado\_da\_tabela**);**

**COMMIT;** -- termina a transação e grava os dados.

→

Comando utilizado para iniciar uma transação que é finalizada gravando dados dentro da tabela no Banco de Dados.

**ROLLBACK -** Faz com que as mudanças nos dados existentes deste o último COMMIT sejam descartadas.

**CREATE TABLE** nome\_da\_tabela **(**

coluna\_da\_tabela especificações\_da\_coluna

coluna\_da\_tabela especificações\_da\_coluna

coluna\_da\_tabela especificações\_da\_coluna

coluna\_da\_tabela especificações\_da\_coluna

**);**

**BEGIN TRANSACTION;** -- começamos a transação

**INSERT INTO** nome\_da\_tabela **VALUES (**dado\_da\_tabela**);**

**INSERT INTO** nome\_da\_tabela **VALUES (**dado\_da\_tabela**);**

**ROOLBACK;** -- as inserções das linhas acima foram desfeitas. →

Comando utilizado para impedir que os dados que foram iniciados em uma transação sejam descartados.

***Sessão 5 - Linguagem SQL ( Parte 2 ).***

**Filtrando Consultas com Where:**

**DQL Query Language:**

**Entendendo o funcionamento da cláusula WHERE:**

**SELECT \* FROM** nome\_da\_tabela **WHERE** coluna\_especifica\_da\_tabela **=** dado\_especifico\_da\_coluna →

Comando utilizado para consultar uma linha específica da tabela.

**Consultas em Múltiplas Tabelas:**

Para juntar dados de múltiplas tabelas é importante destacar que elas precisam ter um relacionamento por meio de um Foreign Key.

**SELECT** apelido\_da\_tabela\_secundaria**.**coluna\_da\_tabela\_secundaria **AS** apelido\_da\_coluna\_da\_tabela\_secundaria **,** apelido\_da\_tabela secundaria**.**coluna\_da\_tabela\_secundaria **AS** apelido\_da\_coluna\_da\_tabela\_secundaria **,** apelido\_da\_tabela\_principal**.**coluna\_da\_tabela\_principal **AS** apelido\_da\_coluna\_da\_tabela\_principal **,** apelido\_da\_tabela\_principal**.**coluna\_da\_tabela\_secundaria **AS** apelido\_da\_coluna\_da\_tabela\_secundaria

**FROM** tabela\_secundaria **AS** apelido\_da\_tabela\_secundaria **,** tabela\_principal **AS** apelido\_da\_tabela\_principal

**WHERE** apelido\_coluna\_secundaria**.**fk( Foreign Key )\_da\_coluna\_secundaria\_que\_tem\_relecionamento\_com\_a\_tabela\_principal **=** apelido\_coluna\_principal**.**pk( Primary Key )\_da\_coluna\_principal →

Comando utilizado para fazer consultas em uma tabela por colunas específicas utilizando o método de Foreign Key e apelidos para as tabelas e apelidos para suas respectivas colunas.

**Junção de Tabelas:**

Podemos ter uma ou mais tabelas relacionadas em um Banco de Dados. Para fazer esse relacionamento precisamos primeiramente desenvolver suas junções.

**Junção de Produto Cartesiano -** É uma junção que origina uma terceira tabela a partir dos da junção dos dados da primeira tabela com a segunda tabela.

**SELECT** tabela\_secundaria**.**pk(Primary Key)\_da\_tabela\_secundaria **,** tabela\_secundaria**.**coluna\_da\_tabela\_secundaria **,** tabela\_secundaria**.**coluna\_da\_tabela\_secundaria **,**

tabela\_secundaria**.**coluna\_da\_tabela\_secundaria **,**

tabela\_principal**.**coluna\_da\_tabela\_principal

**FROM** tabela\_secundaria **,** tabela\_principal

**WHERE** tabela\_secundaria**.**fk(Foreign Key) **=** tabela\_principal**.**pk(Primary Key)**;** →

Comando utilizado para consultar duas tabelas de forma correta, como visto anteriormente.

**Junção Interna (Inner Join) -** É uma junção que vai atender apenas as condições de junção, ou seja; apenas as linhas que se relacionam entre as tabelas. Nesta usamos **ON** ao invés de **WHERE**.

**SELECT** tabela\_secundaria**.**pk(Primary Key)\_da\_tabela\_secundaria **,** tabela\_secundaria**.**coluna\_da\_tabela\_secundaria **,** tabela\_secundaria**.**coluna\_da\_tabela\_secundaria **,**

tabela\_secundaria**.**coluna\_da\_tabela\_secundaria **,**

tabela\_principal**.**coluna\_da\_tabela\_principal

**FROM** tabela\_secundaria **INNER JOIN** tabela\_principal

**ON** tabela\_secundaria**.**fk(Foreign Key) **=** tabela\_principal**.**pk(Primary Key)**;** →

Outra maneira de consultar os dados de duas tabelas de forma correta.

**Junção Externa (Outer Join) ... Left Outer Join -** Esse metódo de junção armazena todos os registros da primeira tabela com todos os valores da segunda tabela correspondentes.

**SELECT \* FROM** primeira\_tabela

**LEFT OUTER JOIN** segunda\_tabela

**ON** primeira\_tabela**.**fk(Foreign Key) **=** segunda\_tabela**.**pk(Primary Key) →

Comando utilizado para analisar todos os dados de uma tabela específica e os dados de outra tabela que tenham relacionamento com esta.

**Junção Externa (Outer Join) ... Right Outer Join -** Nesse método de junção, a consulta atuará de forma contrária a consulta **Left Outer Join**, pois a tabela de referência será a segunda tabela. Então os dados impressos na tela serão os dados da segunda tabela, e os dados da primeira tabela que que correspondem a primeira tabela.

**SELECT \* FROM** primeira\_tabela

**RIGHT OUTER JOIN** segunda\_tabela

**ON** primeira\_tabela**.**fk(Foreign Key) **=** segunda\_tabela**.**pk(Primary Key)**;** →

Comando utilizado para mostrar os dados de uma tabela com os dados de outra tabela onde existe um relacionamento, que correspondem aos valores desta.

**Junção Externa (Outer Join) ... Full Outer Join -** Diferente da consulta **Left Outer Join** e **Right Outer Join**, está consulta vai resultar o valor de ambas mostrando todos os dados na tela.

**Obs.:** Está consulta não funciona no MySQL, porém pode ser usado o **LEFT JOIN** e **RIGHT JOIN** para simular a operação.

**SELECT \* FROM** primeira\_tabela

**FULL OUTER JOIN** segunda\_tabela

**ON** primeira\_tabela**.**fk(Foreign Key) **=** segunda\_tabela**.**pk(Primary Key)**;** →

Comando utilizado para apresentar na tela os valores de duas tabelas que tem um relacionamento.

**SELECT \* FROM** primeira\_tabela

**LEFT OUTER JOIN** segunda\_tabela

**ON** primeira\_tabela**.**fk(Foreign Key) **=** segunda\_tabela**.**pk(Primary Key)**;**

**UNION**

**SELECT \* FROM** primeira\_tabela

**RIGHT OUTER JOIN** segunda\_tabela

**ON** primeira\_tabela**.**fk(Foreign Key) **=** segunda\_tabela**.**pk(Primary Key)**; →**

**MYSQL -** Comando utilizado para apresentar na tela os valores de duas tabelas que tem um relacionamento.

**Funções de Agregação:**

São funções muito importantes nela faremos menções sobre dados de uma coluna completa do Banco de Dados, impondo somas, médias, números de registros, valores mínimos e máximos; ou seja... estatísticas, resumos e totalizações de dados em um registro ( coluna da tabela ).

**Max -** É uma função utilizada para retornar o maior valor em uma coluna da tabela.

**SELECT MAX**(coluna\_da\_tabela) **FROM** nome\_da\_tabela

→

Comando utilizado para retornar o maior valor em uma coluna da tabela.

**SELECT** pk(Primary Key)\_da\_tabela **, MAX**(coluna\_da\_tabela) **FROM** nome\_da\_tabela **GROUP BY** pk(Primary Key)\_da\_tabela →

Comando utilizado para fazer a busca do maior elemento de uma tabela agrupando-o a chave primária respectiva do registro.

**SELECT** pk(Primary Key)\_da\_tabela **, MAX**(coluna\_da\_tabela) **FROM** nome\_da\_tabela **GROUP BY** pk(Primary Key)\_da\_tabela

**HAVING MAX**(coluna\_da\_tabela) condição\_de\_operação valor\_comparador

→

Comando utilizado para fazer a busca do maior elemento de uma tabela agrupando-o a chave primária respectiva do registro em uma condição específica.

**Min** - Método utilizado para retornar o menor valor em uma tabela.

**SELECT MIN**(coluna\_da\_tabela) **FROM** nome\_da\_tabela

→

Comando utilizado para retornar o menor valor em uma coluna da tabela.

**Sum -** É uma função utilizada para realizar a soma de todos os valores em uma determinada coluna.

**SELECT SUM**(coluna\_da\_tabela) **FROM** nome\_da\_tabela **WHERE** pk(Primary Key) **=** indentificador\_fk(Foreign Key)\_específico →

Comando utilizado para retornar a soma dos valores de uma coluna da tabela.

**AVG -** É uma função utilizada para realizar a média de uma dos valores de uma coluna da tabela.

**SELECT AVG**(coluna\_da\_tabela) **FROM** nome\_da\_tabela

→

Comando utilizado para a calcular a média dos valores de uma coluna específica da tabela.

**ROUND -** É uma função utilizada para arredondar valores, especificando quantas casas decimais queremos apresentar em uma determinada consulta.

**SELECT ROUND(AVG**(coluna\_da\_tabela)**,** quantidade\_de\_casas\_decimais) **FROM** nome\_da\_tabela →

Comando utilizado para calcular a média dos valores de uma coluna específica, apresentando o valor com uma quantidade de casas decimais específica também.

**COUNT -** É uma função utilizada retorna o número de registros de uma determinada coluna.

**OBS.:** Por padrão os valores **NULL** não são mencionados. Para que todos os valores sejam mencionados sem exceção é necessário utilizar o “ **\*** “ .

**Funções de Agrupamento e Ordenação:**

São funções que permitem agrupar e/ou ordenar os dados de uma tabela para uma melhor apresentação em uma determinada consulta.

**GROUP BY -** É uma função utilizado para agrupar dados do mesmo tipo.

**SELECT** coluna\_da\_primeira\_tabela **, SUM**(coluna\_da\_segunda\_tabela) **WHERE** pk(Primary Key) **=** indentificador\_fk(Foreign Key)\_específico **GROUP BY** coluna\_específica\_da\_primeira\_tabela →

Comando utilizado para ordenar os dados a partir de uma coluna específica da tabela agrupando os da tabela estrangeira a partir de uma uma coluna da tabela principal.

**SELECT** coluna\_da\_primeira\_tabela **,** coluna\_da\_segunda\_tabela **SUM**(coluna\_da\_terceira\_tabela) **WHERE** pk\_(Primary Key)\_da\_primeira\_tabela **AND** fk(Foreign Key)\_da\_segunda\_tabela **=** fk(Foreign Key)\_da\_terceira\_tabela

**GROUP BY** coluna\_específica\_da\_primeira\_tabela **,** coluna\_específica\_da\_segunda\_tabela **GROUP BY** coluna\_específica\_da\_primeira\_tabela **,** coluna\_específica\_da\_segunda\_tabela →

Comando utilizado para agrupar os dados de duas tabelas a partir da soma dos dados de uma terceira tabela, ordenando primeiramente por uma coluna específica da primeira tabela e depois por uma coluna específica da segunda tabela.

**SELECT** coluna\_da\_primeira\_tabela **,** coluna\_da\_segunda\_tabela **SUM**(coluna\_da\_terceira\_tabela) **WHERE** pk\_(Primary Key)\_da\_primeira\_tabela **AND** fk(Foreign Key)\_da\_segunda\_tabela **=** fk(Foreign Key)\_da\_terceira\_tabela

**GROUP BY** coluna\_específica\_da\_primeira\_tabela **,** coluna\_específica\_da\_segunda\_tabela **GROUP BY** coluna\_específica\_da\_primeira\_tabela **,** coluna\_específica\_da\_segunda\_tabela

**HAVING SUM(**coluna\_especifica\_da\_terceira\_tabela**) >** quantidade\_especifica →

Comando utilizado para agrupar os dados de uma terceira tabela a partir dos dados outras duas tabelas, ordenando primeiramente por uma coluna específica da primeira tabela, e depois por uma coluna específica da segunda tabela, e por ultimo limitando os dados da terceira tabela a partir de uma coluna específica.

**Obs.:** O GROUP BY vai evitar a repetição de dados da primeira e segunda tabela.

**GROUP BY -** É uma função que ordena os dados de acordo com uma ou mais colunas específicas.

**SELECT \* FROM** nome\_da\_tabela

**ORDER BY** coluna\_específica\_da\_tabela →

Comando utilizado para ordenar a tabela a partir de uma coluna específica.

**SELECT \* FROM** nome\_da\_tabela

**ORDER BY** coluna\_específica\_da\_tabela **DESC** →

Comando utilizado para ordenar a tabela a partir de uma coluna específica de forma decrescente.

**Funções de Data e Hora:**

São funções específicas para fazer o trabalho com Datas e Horas.

**CURDATE( ) -** Essa função retorna a Data em formato americano sendo.

**a** = ano.

**m** = mês. **aaaa-mm-dd**

**d =** dia.

**SELECT CURDATE( ) AS** ‘titulo’**; →**

Comando utilizado para retornar a data atual com um título amigável.

**CURTIME ( ) -** Retorna a hora atual no formato.

**h** = hora.

**m =** minuto.

**s** **=** segundo.

**SELECT CURTIME ( ) AS ‘**titulo**’;** **→**

Comando utilizado para retornar a hora atual com um título amigável.

**CURRENT\_TIME( ) -** Faz o mesmo que CURTIME( ) usado anteriormente.

**SELECT CURRENT\_TIME( ) AS** ‘Hora Atual’**; →**

Comando utilizado para retornar a hora atual com um título amigável.

**DATE\_ADD(data, intervalo) -** Comando utilizado para adicionar intervalos em uma data.

**SELECT DATE\_ADD(CURDATE( ), INTERVAL** numero\_de\_dias **DAY) AS ‘**titulo**’; →**

Comando utilizado para adicionar um número específico de dias a uma data específica. Pode ser usado em datas de vencimentos por exemplo.

**DATE\_SUB(data, intervalo) -** Comando utilizado para subtrair intervalos em uma data.

**SELECT DATE\_SUB(CURDATE( ), INTERVAL** numero\_de\_dias**) AS ‘**titulo**’;** **→**

Comando utilizado para subtrair um número de específicos de dias de uma data específica. Pode ser usado como por exemplo em datas de matrícula.

**DATEDIFF(expressão1, expressão2) -** Comando utilizado para verificar a diferença entre um intervalo de datas.

**SELECT DATEDIFF(CURDATE( ), DATE\_SUB(CURDATE( ), INTERVAL** número\_de\_dias **DAY)) AS ‘**titulo**’; →**

Comando utilizado para verificar a diferença de dias específicos de uma data específica para a data atual.

**DATE\_FORMAT(data, formato) -** Comando utilizado para retornar uma data em formato específico.

**SELECT DATE\_FORMAT(CURDATE( ),** ‘formato\_específico’ **AS ‘**titulo**’; →**

Comando utilizado para retornar uma data específica em um formato específico.

**Exemplo:** %d/%m/%Y **.**

**DAYNAME(data) -** Comando utilizado para retornar o dia da semana.

**SELECT DAYNAME(CURDA TE( )) AS ‘**titulo**’; →**

Comando utilizado para verificar o dia da semana indicando um título específico.

**SET Ic\_times\_names = ‘**pt\_BR**’**

**SELECT DAYNAME(CURDATE( )) AS ‘**titulo**’; →**

Comando utilizado para verificar o dia da semana em formato brasileiro, indicando um título específico.

**DAYOFMONTH(data) -** Comando utilizado para retornar o dia do mês.

**SELECT DAYOFMONTH(CURDATE( )) AS ‘**titulo**’; →**

Comando utilizado para verificar o dia do mês indicando um título específico.

**DAYOFYEAR(data) -** Comando utilizado para retornar o dia do ano.

**SELECT DAYOFYEAR(CURDATE( )) AS ‘**titulo**’; →**

Comando utilizado para verificar o dia do ano indicando um título específico.

**FROM\_DAYS(n) -** Comando utilizado para informar a data a partir de um número determinado de dias.

**SELECT FROM\_DAYS(**número\_específico\_de\_dias**) AS ‘**titulo**’;**

**→**

Comando utilizado para verificar uma data a partir de um número específico de dias, indicando um título específico.

**OBS.:** Este comando funciona apenas com respostas baseadas no Calendário Gregoriano.

**NOW() -** Comando utilizado para retornar a data atual.

**SELECT NOW( ) AS ‘**titulo**’; →**

Comando utilizado para verificar a data e horário atual, indicando um título específico.

**SELECT DATE\_FORMAT(NOW( ), ‘**%d/%m/%Y %h:%m:%s**’ AS ‘**titulo**’; →**

Comando utilizado para verificar a data e horário atual em formato brasileiro, indicando um título específico.

**SEC\_TO\_TIME(segundos) -** Comando utilizado para verificar um horário em formato brasileiro a partir de um número específico de segundos.

**SELECT SEC\_TO\_TIME(**quantidade\_de\_segundos**) AS ‘**titulo**’; →**

Comando utilizado para verificar um horário em formato brasileiro, a a partir de um número específico de segundos, indicando um título específico.

**TIME\_TO\_SEC(hora) -** Comando utilizado para atuar de forma contrária ao comando acima. Nesse comando estamos verificando quantos segundos tem em um formato de horário brasileiro.

**SELECT TIME\_TO\_SEC(**‘horas:minutos:segundos’**) AS ‘**titulo**’; →**

Comando utilizado para verificar uma quantidade em segundos a partir de um horário em formato brasileiro, indicando um título específico.

**HOUR(hora), MINUTE(hora), SECOND(hora) -** Comando utilizado para apresentar uma tabela separando colunas entre, **Hora**, **Minutos**, **Segundos** a partir de um horário em formato brasileiro.

**SELECT HOUR(**’horas:minutos:segundos’**) AS** ‘titulo’, **MINUTE(‘**horas:minutos:segundos’**) AS** ‘titulo’**, SECOND(**‘horas:minutos:segundos’**) AS** ‘titulo’**; →**

Comando utilizado para retornar uma tabela separando os dados em colunas de Horas, Minutos e Segundos a partir de um formato específico de horário em formato brasileiro, indicando um título específico.

**PERIOD\_DIFF(periodo1, periodo2) -** Comando utilizado para calcular a diferença de meses entre datas de mês e ano.

**SELECT PERIOD\_DIFF(**‘AnoMês’ **,** ‘AnoMês’**) AS** ‘titulo’**; →**

Comando utilizado para verificar quantos meses existe de diferença a partir de duas datas de Ano e Mês específicas, indicando um título específico.

**TIME\_DIFF(periodo1, periodo2) -** Comando utilizado para calcular a diferença de horários entre dois horários específicos em formato brasileiro.

**SELECT TIME\_DIFF(**‘horas:minutos:segundos’ **,** ‘horas:minutos:segundos’**) AS** ‘titulo’**; →**

Comando utilizado para verificar a diferença entre dois horários específicos em formato brasileiro, imprimindo a resposta em horário brasileiro, indicando um título específico.

**QUARTER(DATA) -** Comando utilizado para retornar o trimestre anual a partir de uma data.

**SELECT QUARTER(**‘data\_específica’**) AS ‘**titulo**’; →**

Comando utilizado para retornar o trimestre anual a partir de uma data específica, indicando um título específico.

**WEEK(data) -** Comando utilizado para verificar qual a semana do ano uma data se encaixa.

**SELECT WEEK(**‘data\_específica’**) AS** ‘titulo’**; →**

Comando utilizado para verificar em qual semana do ano uma data específica em formato americano se encaixa.

**WEEKDAY(data) -** Comando utilizado para verificar qual a semana do ano uma data se encaixa.

**SELECT WEEKDAY(**‘data\_específica’**) AS** ‘titulo’**; →**

Comando utilizado para verificar em qual dia da semana uma data específica em formato americano se encaixa.

**YEAR(data) -** Comando utilizado para verificar qual ano uma data se encaixa.

**SELECT YEAR(**‘data\_específica’**) AS** ‘titulo’**; →**

Comando utilizado para verificar o ano que uma data específica em formato americano se encaixa.

**MONTH(data) -** Comando utilizado para verificar qual mês uma data se encaixa.

**SELECT MONTH(**‘data\_específica’**) AS** ‘titulo’**; →**

Comando utilizado para verificar o mês que uma data específica em formato americano se encaixa.

**DAY(data) -** Comando utilizado para verificar qual dia uma data se encaixa.

**SELECT DAY(**‘data\_específica’**) AS** ‘titulo’**; →**

Comando utilizado para verificar o dia que uma data específica em formato americano se encaixa.