**Italo Ferreira Lima**

Apostila sobre Banco de Dados e Entidade Relacionamento

São José dos Campos, SP

2022

O que é um Dado

Para se entender o que é um banco de dados, precisamos entender primeiramente o que é um dado: Dado é o menor nível de abstração da informação, isto é, a menor informação sobre alguma coisa, definindo até mesmo o tipo do dado que é armazenado, como por exemplo:

João Bezerra, 42, 55, queijo.

Esses dois dados, não fazem absolutamente nada, sozinhos não geram nenhuma informação, mesmo que eu tenha 4 dados (2 de texto e 2 de número), entretanto, quando eu passar um conjunto de dados, estou passando uma informação.

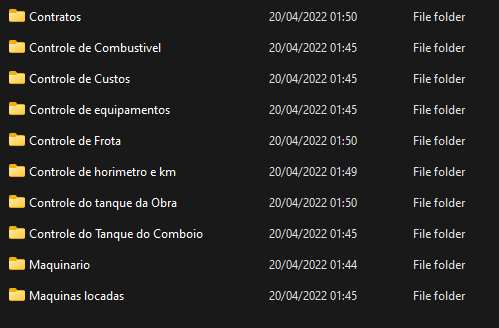
Nome: João Bezerra, idade: 55, Botas: 42, Comida\_Favorita: Queijo.

Por sua vez, esse conjunto de dados, que geram uma ou mais informações, me fornece conhecimento a respeito daquilo que eu busquei, pois agora eu sei quem é a pessoa, quantos anos ela tem, qual o numero que ela calça e qual a sua comida favorita.

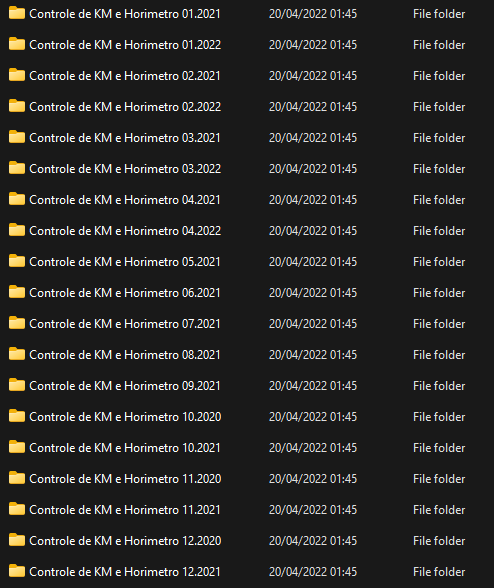
O que é um banco de dados

O banco de dados é uma coleção organizada de informações – ou dados – estruturadas, normalmente armazenadas eletronicamente em um sistema de computador. Um banco de dados é geralmente controlado por um sistema de gerenciamento de banco de dados (SGBD). Banco de dados que não contêm um SGBD, são muito mais lentos, tendo que ser acessado manualmente, uma pasta a uma, para depois ser manipulado o dado, enquanto um SGBD já faz o caminho automaticamente, ligada a todos os BD, todas as tabelas e fazendo as suas devidas modificações, conforme a necessidade do usuário.

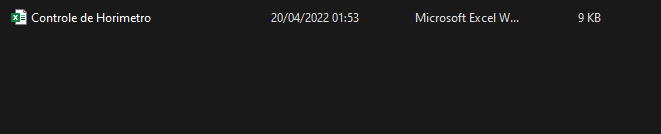
Exemplo de Banco de Dados sem um SGBD. Primeiro o Usuário teria que entrar na pasta onde esta sem BD e se depararia com diversas pastas, dessas diversas, ele teria que entrar na que ele deseja (controle de horimetro e km)



Depois ele teria que achar a pasta que ele quer, seja por meio de busca para ver o histórico ou para acrescentar novos dados (Controle de horimetro 04.22 – 04.22 é o mês referente que estou escrevendo essa apostila). Veja como é difícil enxergar a onde ela está, e como é demorado o processo para achá-la.



Após achar a pasta desejada, ele entraria nela e acessaria o conteúdo desejado, nesse caso é uma planilha com todas as informações a respeito da quilometragem dos caros e as horas rodadas do maquinário, para aí sim abrir o Banco de Dados, e inserir, alterar, pesquisar ou excluir os dados.



Todos esses processos de segundos, poderia ser reduzido a um milésimo de segundo, com um SGBD, ou o mesmo processo teria que ser feito manualmente, caso fosse registros físicos, além de gastar tinta e folha para as impressões.

Modelo Entidade Relacionamento

O Modelo Entidade Relacionamento (MER), também chamado de Modelo ER, é um modelo conceitual utilizado na Engenharia de Software para descrever os objetos (chamadas de entidades ou tabelas de um BD), e suas respectivas características (dados, chamado de atributo) e como elas se relacionam entre si (relacionamentos) em um banco de dados.

Tipos de Atributos

Os atributos descrevem uma característica da entidade, um dado, e possui um tipo, nome e valor específico. Os atributos podem ser Simples, Composto, Multivalorado, Determinante, Identificador, entre outros.

Atributo Simples/Atômico: Não possui características especiais, e são indivisíveis, por exemplo: CPF, CNPJ, RG.

Atributo Composto: É formado por itens menores, podendo ser subdividido em outros atributos, por exemplo: O Atributo Endereço, pode ser subdividido em 3 atributos, a Rua, CEP e Bairro.

Atributo Multivalorado: Pode conter mais de um valor para um mesmo registro (informação). Por exemplo: telefone 1, telefone 2, telefone 3.

Atributo Determinante: Define de forma única os registros de uma entidade, não podendo existir dois registros iguais com o mesmo valor, nesse atributo, não sendo necessariamente a primary key da entidade. (relembrando que o atributo recebe um tipo específico de dado, determinado pelo programador, por exemplo, texto, numérico). Por exemplo: CPF, RG, ID\_Produto

Chaves

Uma chave consiste em uma ou mais colunas de uma relação de entidades, cujo valores são usados para identificar uma linha ou um conjunto de linhas.

Chave Primaria / primary key: Identifica de forma exclusiva os registros em uma tabela, não podendo ter repetição de valores e nem ter valor nulo.

Chave Estrangeira: Chave primaria de uma tabela que se relaciona como chave estrangeira em outra tabela, e tem que obrigatoriamente, ter os mesmos tipos de dados que a coluna da que tem a chave primaria.

Chave Composta: Chave que é composta de dois ou mais atributos (colunas). Geralmente empregada quando não é possível utilizar uma única coluna de uma tabela para identificar de forma exclusiva seus registros (quando há duplicidade).

Chave Surrogada / Substituta: Valor numérico, único, adicionado a uma relação para servir como chave primária, quando não se consegue definir uma chave primaria nos próprios atributos da entidade.

Entidades

A entidade é um objeto, um item que vai ser representado no MER, que representa um tema, tópico ou conceito de negócio, que precisa ser representado em um banco de dados. Por exemplo: Empregados, Livros, Vendas, Produtos.

Algumas regras de nomeação de entidade:

* Devem começar com uma letra;
* Usar palavra no singular;
* Não podem ter espaços ou alguns caracteres especiais;
* Alguns caracteres como “$”, “#” e “\_” são permitidos em alguns bancos de dados.

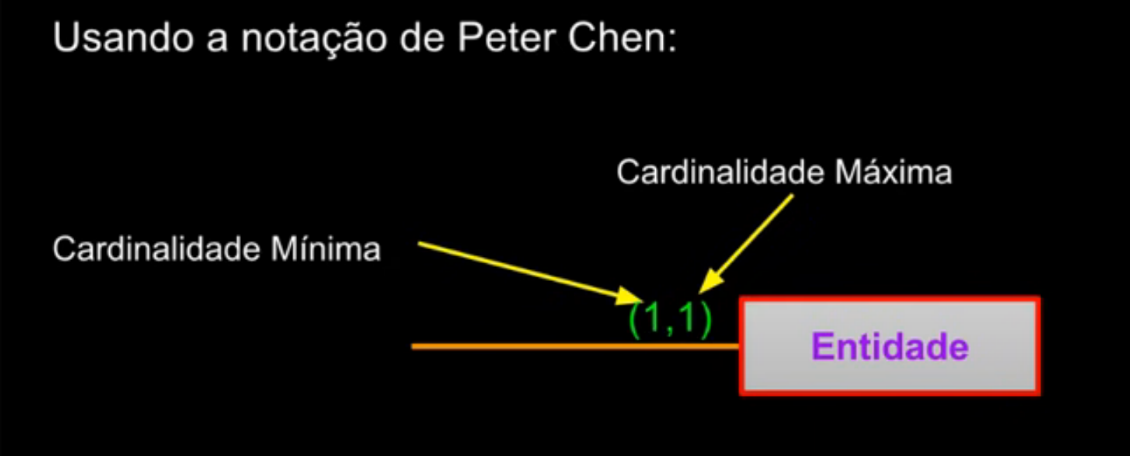
Os nomes de colunas devem ser únicos dentro de uma tabela de entidade, e o nome das tabelas/entidades devem ser únicos dentro do esquema.

Instância de Entidade: A instância de uma entidade é uma ocorrência específica de uma entidade, podendo também ser chamada de registro, pois o registro é uma instância de entidade, por ser algo específico dentro daquela entidade.

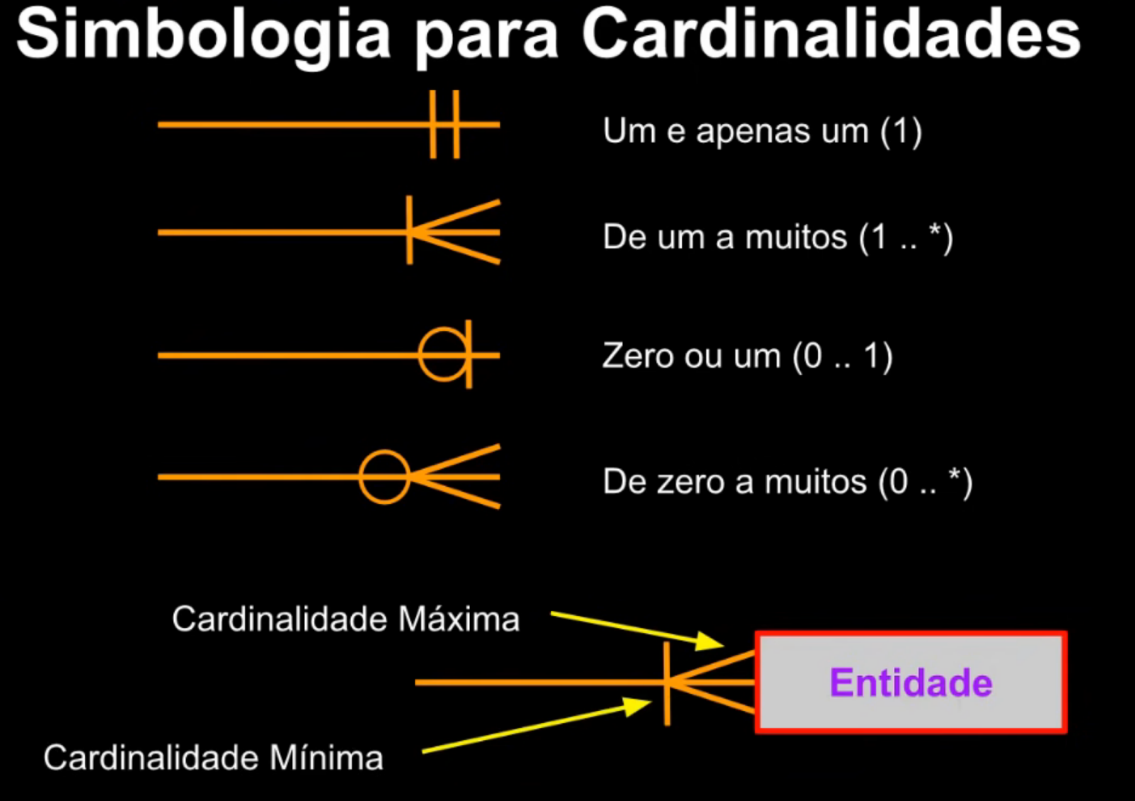
Cardinalidades: A Cardinalidade diz respeito ao número de itens que se relacionam nas entidades, tendo uma cardinalidade mínima e uma cardinalidade máxima.

Cardinalidade Máxima: É o numero máximo de instâncias de entidades (registros) que podem participar em um relacionamento, podendo ser 1 ou N ou M, caso o N já exista no relacionamento (letra utilizada para descrever muitos relacionamentos).

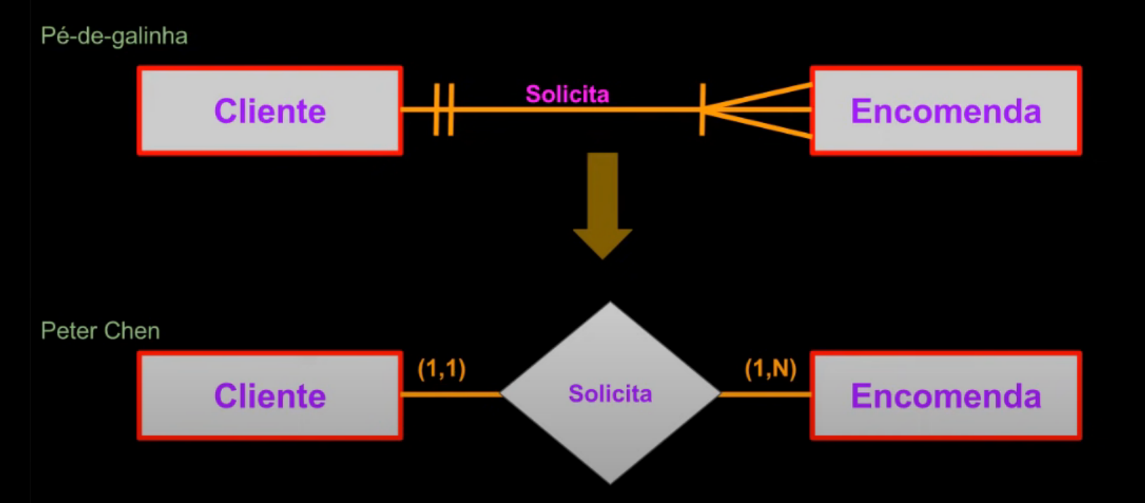
Cardinalidade Mínima: É o número mínimo de instâncias de entidades (registros) que devem participar em um relacionamento, podendo ser 0 (participação opcional), 1 é obrigatório ou N ou M(muitos).



Usando o sistema de Peter Chen no MER.



Usando a Simbologia Pé de Galinha no MER.



Exemplo no MER.

No exemplo acima, utilizando a lógica de mínimo na esquerda e máximo na direita.

Olhando a entidade Cliente, Cliente Solicita (mínimo de cliente, maximo de encomendas), pois o mínimo de cliente é 1 cliente e o máximo de encomendas são 1 ou muitas encomendas, visto que um cliente pode pedir mais de 1 encomenda.

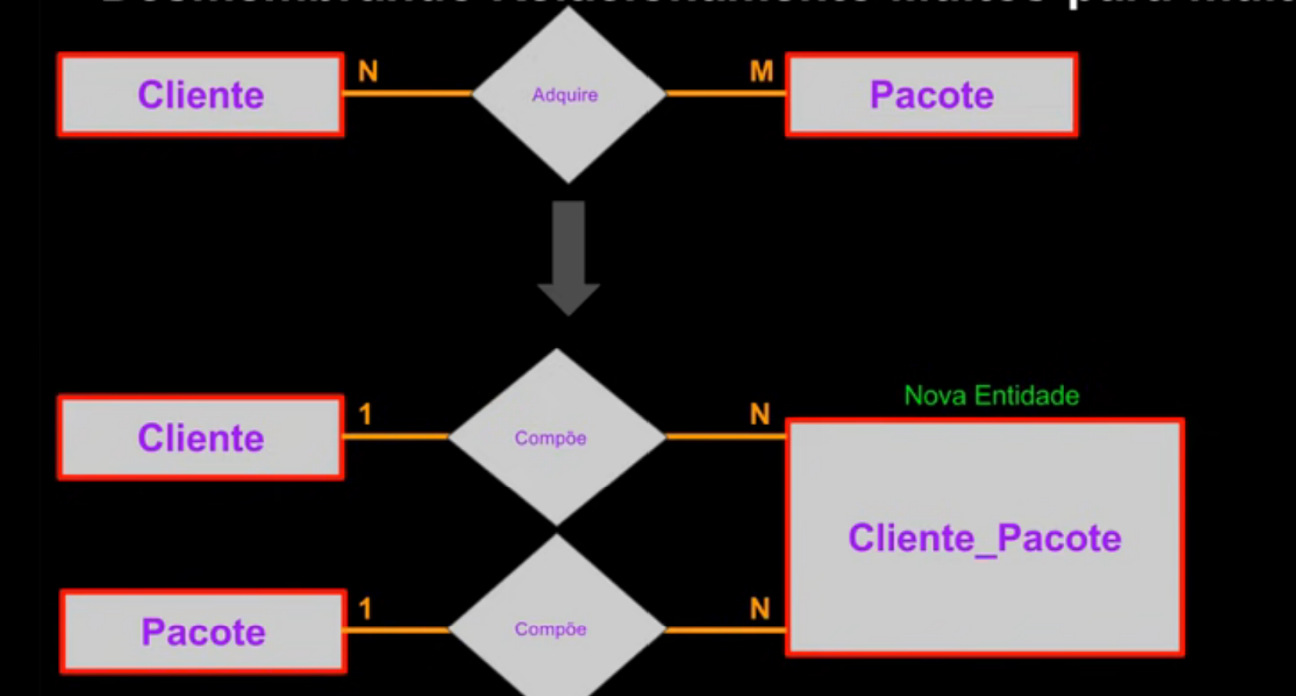
Olhando pelo lado da entidade encomenda, o mínimo é 1 cliente para 1 encomenda, pois uma encomenda não pode ter 2 clientes.

Dentro do MER, existem 3 tipos de entidades.

Entidades Fortes: São aquelas que não precisam de nenhuma outra entidade para existir, dentro de um MER, como por exemplo, uma entidade chamada “Produto”. A entidade “Produto”, existe por si só, e não precisa de um relacionamento para existir.

Entidades Fracas: São as contrarias do forte, elas precisam de outras entidades para existir, dentro de um MER, como por exemplo, uma entidade chamada “Venda”. A entidade “Venda”, precisa de alguma entidade, seja um produto ou serviço, pois sem elas, a entidade fraca perde seu significado, o seu valor dentro de um MER.

Entidades Associativas: Esse tipo de entidade surge quando há necessidade de associar uma entidade a um relacionamento existente, como por exemplo, o relacionamento Muitos para Muitos, para virar dois relacionamentos 1 para muitos, como por exemplo, o relacionamento entre uma entidade chamada “Cliente” e uma entidade chamada “pacote”. Mínimo 1 cliente, para no mínimo 1 pacote, mas 1 cliente pode ter vários pacotes, do mesmo jeito que um pacote pode ter vários clientes, Criando assim a entidade “Cliente\_Pacote”, 1 “Cliente” pode ter muitos pacotes (Cliente\_Pacote), enquanto 1 “Pacote” pode ter muitos clientes (Cliente\_Pacote).



Tipo de Dados do Banco de Dados

Quando vamos criar um banco de dados, precisamos saber quanto de espaço cada uma das colunas deverá armazenar, pois banco de dados armazenam muitos dados, gerando muito peso do arquivo, e reservar um grande espaço para algo que não vai ser ocupado com informações, somente gerará peso no seu banco de dados.

Ressaltando que quando uma tabela vai se relacionar com outra, a coluna da chave primaria da tabela que vai ser inserido como chave estrangeira na outra tabela, devem ser exatamente iguais.

Irei mostrar alguns tipos de dados mais comuns e como criar e modificar eles. Usarei o MariaDB, mas os mesmos comandos valem no MySQL.

Descreverei o nome do dado e entre parênteses () quantos caracteres que é recomendado se armazenar de cor na cor laranja, para que não se ultrapasse do valor máximo, e na cor roxa qual o verdadeiro limite do tipo. Caso não tenha a cor laranja, ou esteja ultrapassando a margem laranja, significa que você deve analisar o sistema que está querendo criar e ver qual dos mesmos se adapta melhor a ele, como por exemplo, de tinytext para text.

Tipos Numéricos:

* TINYINT(2, 128 e -128): Número inteiro muito pequeno.
* SMALLINT(4, 32.767 e -32.768): Número
* MEDIUMINT(6, 8.388.607 e -8.388.608): Número
* INT(9, 2.147.483.457 e -2.147.483.648): Número
* BIGINT(18, 9.223.372.036.854.775.807 e 9.223.372.036.854.775.808): Número
* DECIMAL(M, D / 65, 30): M é o numero de algarismos que vai ter (máximo 65), e o D e quantas casas depois da virgula, o número vai ter (máximo 30), utilizando o separador de decimais “.”.
* FLOAT(2, 128):
* DOUBLE(2, 128):
* BIT(2, 128):

Tipos de Strings (texto):

* CHAR(N , 255): Texto com N caracteres, sempre totalmente preenchido, seja pelo usuário ou pelo próprio BD como espaços.
* VARCHAR(N , 255): Texto que pode conter até N caracteres. O espaço não utilizado, não é preenchido com espaços.
* BINARY(N , 255): Qualquer informação em binário, até N algarismos, completando os algarismos não utilizados, como 0.
* VARBINARY(N , 255): Qualquer informação em binário, até N algarismos, sem completar o espaço não preenchido.
* BLOB(250, 255): Qualquer informação em Binário, geralmente utilizado para imagens, vídeos e executáveis. Isso vale para os outros tipos de blob.
* TINYBLOB(2, 128): Qualquer informação em Binário.
* MEDIUMBLOB(100.000.000, 116.777.216): Qualquer informação em Binário.
* LONGBLOB(4.000.000.000, 4.294.967.295): Qualquer informação em Binário.
* TINYTEXT(255):
* TEXT(65.535):
* MEDIUMTEXT(16.777.215): Texto até 16.777.215 Caracteres.
* LONGTEXT(4.294.967.295): Texto até 4.294.967.295 Caracteres.
* ENUM(“X”, “Y”, “Z”): Aceita somente uma das respostas definidas dentro do enum.
* SET(“X”, “Y”, “Z”): Aceita nenhuma ou várias respostas pré-definidas no Set, sendo inserido na tabela dentro das mesmas aspas que esta o campo “set”, tendo cada um deles separado por virgulas, como por exemplo: (“X, Y, Z”) ou (“”) ou (“X, Z”).

Tipos de Data e Hora:

* DATE: Data no formato YYYY-MM-DD.
* DATETIME: Data e hora no formato YYYY-MM-DD HH:MM:SS.
* TIME: Hora, apenas no formato HH:MM:SS
* YEAR: Ano no formato YYYY

**Atributos/Propriedades**

Os atributos são propriedades que você acrescenta a um campo, para que ele se torne mais específico e atenda melhor aos parâmetros (ex: entre 0 e 255) de dados que você vai inserir neles.

Ex: TINYINT(2, 128 e -128): pode ir até o numero 255, se você acrescentar o atributo unsigned ou zerofill (que também acrescenta o atributo unsigned).

* Zerofill: Não utiliza 0 antes do número na hora de mostrar a informação para o usuario (ex: 0002 e 00480 se transforma em 2 e 480)
* Signed (padrão): Permite o número ser negativo (ex: TINYINT(2, 128 até -128))
* Unsigned: Não permite valores negativos, aumentando a capacidade do tipo numérico (ex: TINYINT(2, 255)).
* Auto\_increment: Acrescenta automaticamente +1 ao ultimo registro, criando arquivos em sequência (ex: 1, 2, 3, 4, 5, 6), bastante utilizado para criar ID, garantindo que o valor sempre seja +1 e não se repita.
* Not Null: Campo tem que ser preenchido obrigatoriamente.
* Unique: Um dado não pode ser salvo se já houver outro dado exatamente igual (ex: fui cadastrar alguém e já constou no sistema que o CPF ou o nome já está cadastrado).
* Default: Define um valor padrão, caso o campo não seja preenchido na hora de inserir os novos dados no BD.

**Características de um BD**

**Transação:** Operação executada entre o inicio e o fim de uma transação, usando as propriedades **ACID** descritas abaixo.

**Transação Concorrente:** Dois usuários tentam acessar a mesma informação para realizar uma transação, e para não haver esse problema de 2 usuários tentando acessar a mesma informação, utilizamos um **bloqueio binário**.

**Atomicidade:** Em uma transação envolvendo duas ou mais partes de informações discretas, ou a transação será executada totalmente ou não será executada, garantindo assim que as transações sejam atômicas.

**Consistência:** A transação cria um novo estado válido dos dados ou em caso de falha retorna todos os dados ao seu estado antes que a transação foi iniciada.

**Isolamento:** Uma transação em andamento, mas ainda não validada deve permanecer isolada de qualquer outra operação, garantindo que não haverá interferência por nenhuma outra transação concorrente.

**Durabilidade:** Dados validados são registrados pelo sistema de tal forma que mesmo no caso de uma falha e/ou reinício do sistema, os dados estão disponíveis em seu estado correto.

**Linguagem SQL**

As linguagens SQL é uma só, porém ela é dividida em tipos, de acordo com a funcionalidade dos comandos, sendo:

**DDL - Data Definition Language:** Linguagem de Definição de Dados. São comandos que interagem com os objetos do BD.

Ex: CREATE, ALTER e DROP

**DML – Data Manipulation Language:** Linguagem de Manipulação de dados. São Comandos que interagem com os dados dentro das tabelas.

Ex: INSERT, DELETE e UPDATE

**DQL – Data Query Language:** Linguagem de Consulta de Dados. São comandos de consulta.

Ex: SELECT

**DTL – Data Transaction Language:** Linguagem de Transação de Dados. São os comandos para controle de transação.

Ex: BEGIN TRANSACTION, COMMIT e ROLLBACK

**DCL – Data Control Language:** Linguagem de Controle de Dados. São os comandos para controlar a parte de segurança do BD.

Ex: GRANT, REVOKE e DENY

**Comandos**

Tudo que estiver capitalizado (maiúsculo) é o comando, e o que estiver em minúsculo é variável que você pode definir. Será utilizado exemplo dos comandos puros, utilizados pelo CLI (Command Line Interface), e o mesmo pode ser aplicado manualmente a uma interface gráfica (como um workbench).

* CREATE DATABASE nome\_database; Cria um novo banco de dados.
* CREATE TABLE nome\_tabela; Cria uma nova tabela, tendo que se identificar o nome das colunas e o tipo de dados dela, e os atributos/propriedades, caso elas o tenham, além de identificar qual a chave primaria. Caso não faça isso de primeira instancia, você vai precisar definir isso com comandos para alterar as colunas, com o ALTER TABLE. Ex. de create table abaixo.

CREATE TABLE pessoas (

id int NOT NULL AUTO\_INCREMENT,

nome VARCHAR(30) NOT NULL,

nascimento DATE,

ano YEAR,

sexo ENUM(‘M’,’F’),

peso DECIMAL(5,2),

altura DECIMAL(5,2),

nacionalidade VARCHAR(20) DEFAULT ‘Brasil’,

PRIMARY KEY (id)

) DEFAULT CHARSET = utf8;

\\Relembrando que o DECIMAL(5,2),se refere a um numero decimal de 5 dígitos, sendo 2 deles após a virgula (ex: 182.22, 025.22) e o DEFAULT CHARSET = utf8; está mudando o dicionário para o nosso alfabeto, permitindo acentuações.

* DESC nome\_tabela; Descreve o nome, o tipo e as propriedades da coluna da tabela.
* INSERT INTO nome\_tabela; Inserir dados em uma tabela, tendo que se especificar quais campos você quer inserir, e depois os valores. Não precisa especificar a chave primaria, ou colocar como VALUES DEFAULT, caso ela seja um numero com o atributo Auto\_increment, pois ela será gerada automaticamente, de forma sequencial. Ex. abaixo.

INSERT INTO pessoas

(nome, nascimento, ano, sexo, peso, altura, nacionalidade)

VALUES

(‘Aline’,’2001-04-02’, ’2001’ , ’F’,’78.5’,’1.83’,’Japão’);

\\Caso a nacionalidade não tivesse sido preenchida com uma informação (como no exemplo, Japão), ela automaticamente teria sido preenchida por “Brasil”, como definida a propriedade DEFAULT ‘Brasil’ no campo da nacionalidade, na hora da criação da tabela

* ALTER TABLE nome\_tabela; Define que você quer realizar uma alteração na tabela que você mencionou. Esse comando sempre vem acompanhado de algum outro comando como ADD ou DROP.
* ADD COLUMN nome\_nova\_coluna TIPO AFTER nome\_coluna\_existente; Adiciona uma nova coluna, após a coluna referenciada. Precisa especificar o tipo e os atributos dela. Caso deseje acrescentar uma coluna como a primeira coluna do seu BD, você tem que trocar o AFTER por FIRST e não precisa mencionar uma coluna existente.
* DROP COLUMN nome\_coluna; Deleta a coluna.

//DROP pode ser utilizado para deletar a tabela ou o BD.

* ADD PRIMARY KEY (nome\_coluna); Cria uma chave primaria, caso não exista uma chave primaria.
* ADD FOREIGN KEY (nome\_coluna); Cria uma chave estrangeira, e precisa ter especificado o nome da coluna e o nome da tabela que ela vai vir. Por exemplo:

ALTER TABLE pessoa

ADD FOREIGN KEY (nome)

REFERENCES outra\_tabela(nome\_coluna);

* JOIN: Ligação após o FROM nome\_tabela, que serve para ligar uma tabela a outra, mas ela liga totalmente, e não somente os que tem ligação com as duas tabelas.
* INNER JOIN: Ligação após o FROM nome\_tabela, que liga uma tabela a outra, somente os registros que têm ligação nas duas tabelas, por meio da FOREIGN KEY.

SELECT nome\_tabela.nome\_coluna, nome\_outra\_tabela.nome\_outra\_coluna FROM nome\_tabela INNER JOIN nome\_outra\_tabela

ON nome\_outra\_tabela.nome\_outra\_coluna = nome\_tabela.nome\_coluna

\\Após o ON, é especificado o relacionamento da chave estrangeira com a chave primeira

* UPDATE nome\_tabela; Altera um ou mais campos de um ou mais registros. Tomar bastante cuidado com esse comando, pois ele pode modificar todos os registros do BD, caso as condições que você especificou, sejam atendidas. Exs. Abaixo.

UPDATE pessoas

SET nome = ‘Maria’

WHERE id = ‘1’;

\\Caso eu tivesse mais de um id = 1, todas as linhas que tem id = 1, o campo nome seria mudado para ‘Maria’ e os dados não poderiam ser recuperados, exceto você tiver feito um backup antes.

\\O WHERE pode-se utilizar operadores relacionais para otimizar as buscas, como os sinais >, <, >=, =<, =, (!= ou <>).

UPDATE pessoas

SET nome = ‘Luiz’, sexo = ‘M’

WHERE sexo = ‘F’;

LIMIT 1;

\\Nesse caso, eu estou especificando que eu quero mudar apenas um único registro pelo comando LIMIT 1, então meu SQL verificaria do primeiro ao último registro, e o primeiro registro que ele encontrasse o sexo = ‘F’, ele mudaria nome = ‘Luiz’, sexo = ‘M’. Caso não tivesse o comando LIMIT, todos os registros que o sexo = ‘F’ se tornariam sexo = ‘M’ além de modificar o nome de todos os nomes para “Luiz”, onde o sexo = ‘F’.

* TRUNCATE TABLE nome\_tabela; Limpa todos os registros, sem perder as propriedades e tipo das colunas.
* SELECT nome\_coluna FROM nome\_tabela; Seleciona linhas de registro, podendo ser especificado várias colunas por meio da separação de virgulas, ou selecionado todas as colunas usando \* no lugar do nome das colunas. Exs. Abaixo.

SELECT nome, ano FROM pessoas

WHERE ano IN (2001, 2004, 2005)

ORDER BY nome; \\Estou selecionando os campos “nome” e “ano”, onde o ano é 2001, 2004 ou 2005, ordenando-o pelo nome de forma ascendente (ASC - padrão), mas posso colocar após o nome, para ele order por forma descendente com o comando DESC. Caso eu troque o IN por BETWEEN eu vou procurar todos os valores entre dois parâmetros, por exemplo, todos os registros que estão entre 2001 e 2004 (WHERE BETWEEN(2001 AND 2004))

* LIKE; Operador para buscas que significa “parecido com”, com “%” ou “\_” onde esses dois caracteres especiais fazem a total diferença, dizendo a que eu quero alguma coisa que começa com X, e tem ou não, quaisquer caracteres depois dele, ou tem alguma coisa obrigatoriamente depois ou antes, em sua ordem respectiva. por exemplo:

SELECT \* FROM pessoas

WHERE nome LIKE “M%”;

\\O sistema me retornaria todas as pessoas que começam com nome M, como Marcos, Maria, Mauro

WHERE nome NOT LIKE “M%”;

\\O Sistema me retornaria todos os nomes que não começam com M, como Ana, Claudia, Luiz.

WHERE nome LIKE “%M%”;

\\Tem “M” em qualquer lugar, como Ana Maria, Marcos, Nathalia Mayer.

WHERE nome NOT LIKE “%M%”;

\\Não tem “M” em nenhum lugar, como Ana, Laura, Luiz.

WHERE nome LIKE “M\_r”;

\\Que começa com M, tem algum caractere logo em seguida, e o R depois desse caractere, como por exemplo, Marcos, Marcio, Marta.

* SUM; Operador que trabalha com o SELECT, realizando a somatória de todos os números da coluna, com os filtros aplicados, por exemplo, filtrar em um ano específico, um numero de carga horaria dos cursos ministrados nesse ano.
* AVG; Operador que trabalha com o SELECT, que realiza a média da coluna, podendo ser aplicado filtros para saber, por exemplo, a media da carga horaria das aulas ministradas, entre 2 anos.
* COUNT(nome\_coluna); Operador que trabalha com o SELECT, que cria uma coluna na visualização, para mostrar quantos elementos existem. Ao invés do nome da coluna, pode-se colocar \*.
* MIN; Operador que trabalha com o SELECT, realiza a busca em toda a coluna e retorna o valor mínimo do primeiro registro que ele encontrar, mesmo que haja 2 registros com o mesmo valor. O mesmo vale para o operador MAX.
* DISTINCT; Operador que trabalha com o SELECT, que realiza a busca da coluna desejada, e retorna um valor de cada tipo, sem que haja duplicidades, por exemplo, ele me retornou o numero 40, então ele ira ignorar todos os outros números 40 na hora de realizar a busca, encontrou o numero 20 logo em seguida, ele vai retornar o 40 e o 20, pra mim, e ignorar agora os 2 números na hora de realizar a busca.
* GROUP BY nome\_coluna; Operador de agrupamento trabalha com o SELECT, que agrupa todos os dados, mostrando o que mais reincide para o que menos reincide, de cima para baixo, e se feito junto com o operador COUNT(nome\_coluna), ele cria uma outra coluna de visualização, mostrando quantas vezes aquele dado foi reincidido.