**CICLO 1:**

**14/08:**

**SISTEMA GERENCIADOR DE BANCO DE DADOS - SGBD (DSMS)**

**Principais funções de um SGBD Relacional:**

* **Gerenciamento do Dicionário de Dados:**
  + Onde encontra-se a estrutura de um objeto, número de campos e de que se trata cada um deles.
  + Guarda a definição de cada campo.
  + Atua em todos os comandos SQL
  + Tudo é criptografado
  + **Metadados:** descrição de como os dados estão organizados
* **Independência dos Dados Armazenados:**
* Não sabe-se o que acontece por trás de cada dado
* **Gerenciamento da Segurança:**
* Depois dos funcionários, os dados são a coisa mais importantes de uma empresa.
* **Gerenciamento de Acesso à Múltiplos Usuários:**
* Número ilimitado de usuários conectados
* Totalmente gerenciada pelo SGBD (sabe quantos usuários estão conectados e organiza-os)
* **Integridade de Dados:**
* Função mais importante
* Quando tem-se chave primária e estrangeira
* Garante que ninguém apagará um dado que está sendo usado e relacionado com outro
* **Gerenciamento de Backup e Recovery**
* Recomenda-se fazer pelo menos 1 vez por mês
* **Linguagem de Acesso**
* **Múltiplas Visões dos Dados:**
* Mesmos conjuntos e diferentes selects

**Arquitetura de um SGBD:**

* Externo: Usuários
* Conceitual: Programadores
* Interno: Administrador

**Memória cash:** é a mais rápida. Acesso rápido. Servidores possuem mais.

**Memória RAM (principal):** Usada pra manter programa em execução, carregar rotinas, guardar dados.

**Memória Secundária:** é o disco SSD

**Memória terciária:** CD, DVD, pen drive... dispositivos externos.

**CICLO 2:**

**21/08:**

**MODELAGEM DE DADOS**

* Mini mundo (cenário): é uma pequena porção que queremos automatizar.
* Coleta e análise de requisitos:
* **Requisitos funcionais:** são os que interferem diretamente no sistema. É o que o sistema tem que fazer. É o foco da aula de BD.
* **Requisitos não funcionais:** não impactam diretamente no funcionamento do sistema. Serve de apoio.
* Modelo entidade-relacionamento: caixa que contém os requisitos e integram um mapa de como os dados devem ser utilizados.
* Depois que o modelo for criado, escolhe-se o que fazer com ele.
* Projeto lógico: quando sabe-se que vai usar BD, mas não sabe-se de que forma
* **MER:** notação do Peter Chen (utilizada academicamente)
* **Entidades:** Coisas que possuem características próprias (substantivos)
* **Atributos:** Caracterizam as entidades (ligados as entidades por arestas com bolinhas na ponta)
* **Relacionamentos:** Forma com que as entidades se relacionam/ligam.
* Há necessidade de guardar dados e informações de cada projeto para facilitar a utilização dos mesmos.

**VERTABELO:**

* **M:** campos obrigatórios;
* **PI:** Primer Key;

**04/09:**

* **Vertabelo:**
* M: preenchimento obrigatório;
* PI: primer Key;
* Char, varchar, date e integer: tipos primitivos;
* Não utilizar caracteres especiais em nenhum momento
* Tabela mãe: não tem fk, ela fornece pk e dados para outras tabelas. É uma tabela base
* No HeideSQL, utiliza-se ponto e não vírgula
* Diferentes padrões de nomenclatura podem apontar para o mesmo local (EX: Blumenau, Blumenal e BNU);

**18/09:**

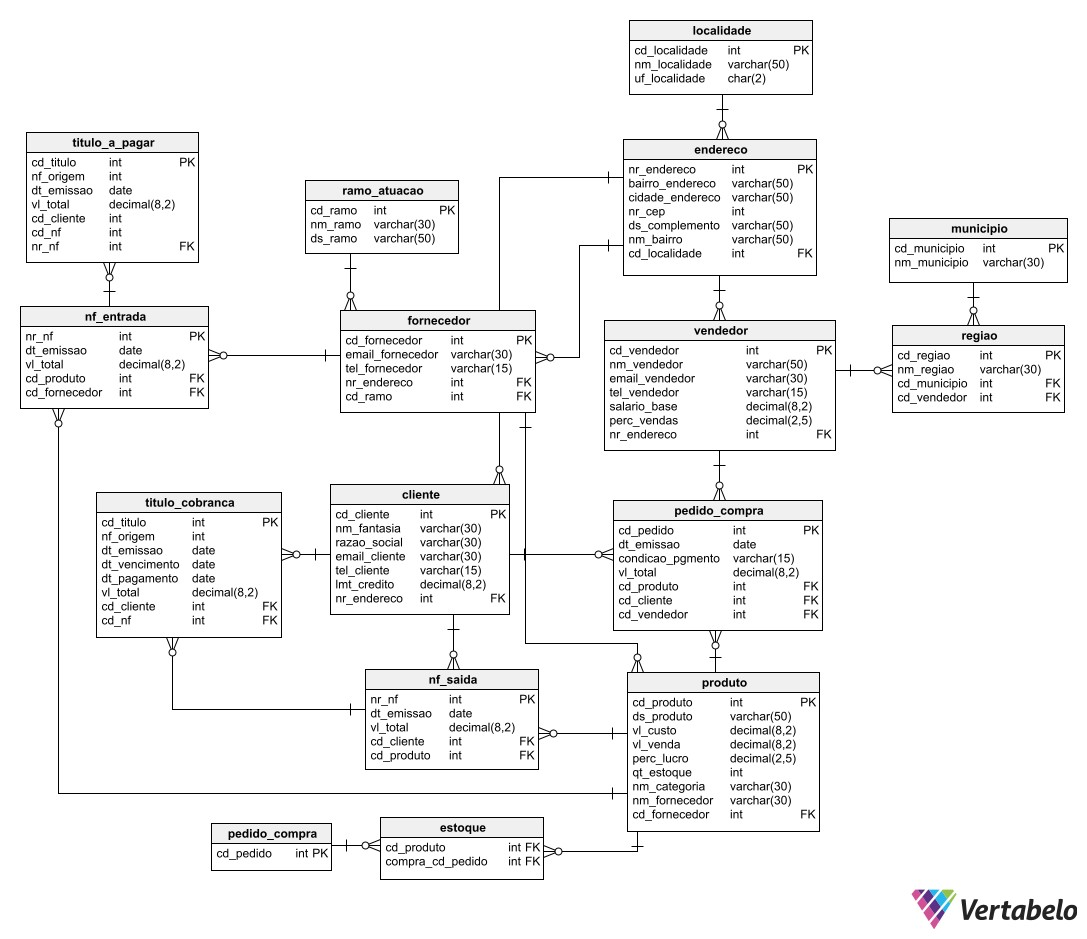
* Cria-se tabelas para casos onde podem haver mais de uma forma de escrever o mesmo dado (EX: Blumenau, Blumenal e BNU);

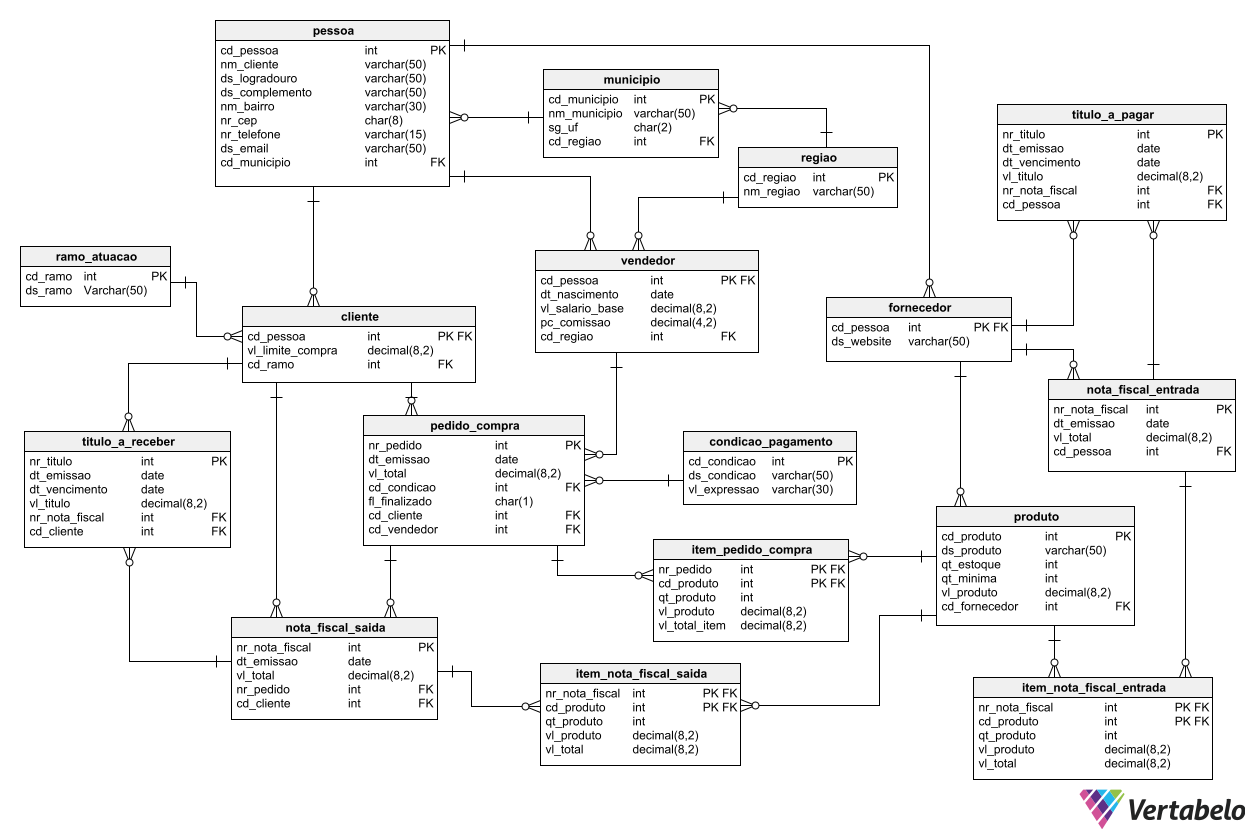
**Estudo de caso**  
O cenário envolve a automação de alguns processos em uma oficina mecânica. A seguir alguns aspectos a serem considerados:

* há um cadastro de clientes com as informações básicas como nome, endereço e dados de contato (telefone e e-mail);
* guarda-se informações básicas do veículo como: placa, marca, modelo, ano fabricação e ano modelo;
* um cliente pode solicitar uma ou mais ordens de serviço sem que precise ser o proprietário de um veículo;
* em uma ordem de serviço poderá haver um ou mais serviços oferecidos pela oficina, assim como peças que venham a ser substituídas;
* as peças são classificadas em categorias (suspensão, elétrica, motor, etc) e apresentam nome, descrição, quantidade em estoque, valor de custo e valor de venda;
* a ordem de serviço é enviada a um ou mais profissionais que atuam na empresa de acordo com sua especialidade para controle da execução dos serviços prestados;
* sobre os serviços prestados são armazenados: descrição e valor;
* a ordem de serviço, após ter sido finalizada, pode gerar uma ou mais notas fiscais (ex. nota fiscal de serviço e nota fiscal de peças);
* a nota fiscal deve contemplar as informações do(s) serviço(s) prestado(s), quantidade(s), valor(es), assim como a relação de peças que porventura tenham sido substituídas.

**01/10:**

* **ESPECIALIZAÇÃO:**
* Quando há atributos que se aplicam a um tipo específico de pessoa;
* Operação segmentada;
* Junção do que é comum, separação do que é específico;
* Herança em POO;
* Conceito mais importante;
* **GENERALIZAÇÃO:**
* Junção de todos os dados em uma “super tabela”;
* Campos não preenchidos ficam nulos;
* Não há diferença/impacto no armazenamento;
* Não é possível cadastrar a pessoa com dois papéis ao mesmo tempo;
* **DEPENDÊNCIA DA EXISTÊNCIA:**
* Ocorre quando uma estrutura depende da outra para existir;
* Nem toda tabela filha (é filha quando possui FK) depende da mãe para existir;
* 1 –> n: um grupo não precisa ter cotas, mas uma cota precisa de um grupo / um grupo pode ter várias pessoas, mas uma pessoa não pode estar em vários grupos;
* Um dependente só existe por causa do colaborador: (EX: colaborador 1 tem 3 dependentes: 1 – 1, 1 – 2, 1 – 3; colaborador 2 tem 1 dependente: 2 – 1;





**16/10:**

* **COMANDOS DDL:**
* **CREATE:** utilizado para definir uma estrutura/tabela;
* dados primitivos: Integer, char (tamanho fixo, ocupa todos os espaços reservados), varchar (tamanho variado, exclui os campos inutilizados), Date;
* **ALTER:** utilizado para modificar uma estrutura/tabela;
* **TRUNCATE:** utilizado para limpar/zerar uma tabela, a estrutura é mantida e os dados são zerados (não existe como retomar os dados);
* **DROP:** utilizado para eliminar a tabela inteira sem deixar vestígios;
* **CONTRAINTS / RESTRIÇÕES:**
* **Primer key:** identificador único, garante que não haverá valores repetidos e o valor é obrigatório, não aceita nulo;
* **Foreing key:** identificador que aponta para uma PK de outra tabela;
* **Not null:** obriga o armazenamento de um valor dentro da coluna que foi criada como obrigatória (ou default = 0 / ‘ ‘, ou seja, não terá um valor nulo);
* **Unique key:** identificador de uma chave única, ou seja, garante que não haverá valores repetidos, mas aceita nulo;
* **Check:** utilizado para validar um campo
* -- comentário de linha
* \*/
* Comentário de bloco
* \*/
* Data e letras sempre entre ‘ ‘;

*-- criando a tabela pessoa*

**CREATE** **TABLE** pessoa

(cd\_pessoa **INTEGER**,

nm\_pessoa **VARCHAR**(10),

ds\_email **VARCHAR**(50)

);

*-- alterando o estrutura de uma tabela para adicionar o valor default*

**ALTER** **TABLE** pessoa

**MODIFY** **COLUMN** nm\_pessoa **VARCHAR**(50) **DEFAULT** 'sem nome';

*-- inserindo valores nos campos código e data nasc*

**INSERT** **INTO** pessoa (cd\_pessoa, dt\_nascimento) **VALUES** (2, '2004/09/28');

*-- adicionando uma nova coluna*

**ALTER** **TABLE** pessoa

**ADD** **COLUMN** dt\_nascimento **DATE**;

*-- removendo uma coluna da tabela pessoa*

**ALTER** **TABLE** pessoa

**DROP** **COLUMN** ds\_email;

**DESCRIBE** pessoa; *-- DESC = DESCRIBRE*

*-- alterando o tamanho de uma coluna*

**ALTER** **TABLE** pessoa

**MODIFY** **COLUMN** nm\_pessoa **VARCHAR**(20);

*-- descrevendo / mostrando a estrutura de uma tabela*

**DESC** pessoa;

*-- inserindo um registro na tabela pessoa*

**INSERT** **INTO** pessoa **VALUES** (3, 'pessoa 3', 'p3@email.com.br', '2023/10/16');

*-- realizando uma busca completa (todos os registros/campos) de pessoa*

**SELECT** \* **FROM** pessoa

**30/10:**

* **COMANDOS DML:**
* **INSERT:** Inserir dados em determinada tabela;
* **UPDATE:** Serve para modificar determinado dado. Utilizar WHERE para mostrar qual o dado a ser modificado;
* **DELETE (TRUNCATE TABLE):** Utilizado para excluir dados selecionados de determinada tabela;
* **SELECT:**
* Toda coluna que tiver mais de uma função de grupo deve estar no group by;
* Não se agrupa dado pelo código
* **WHERE:** mostra onde determinado dado se encontra (não aceita funções de grupo);
* **GROUP BY:** agrupa a tabela de acordo com uma condição (utilizar sempre que há mais e uma função de grupo);
* **HAVING:** normalmente aparece acompanhando o where (faz o que o Where não faz);
* **ORDER** **BY:** ordena os dados de acordo com a forma escolhida, de forma ascendente (se não, utilizar DESC). Não utilizar nome composto, utilizar a posição da coluna na clausula SELECT, ou nome ou apelido da coluna;
* **FROM:** é obrigatório;
* Escrever cada clausula numa linha, cada coluna numa linha e cada tabela em uma linha, para manter a organização do código;
* **%:** é o operador que, juntamente com o LIKE, serve para encontrar determinado caractere ou palavra;
* **IS NULL:** Verifica se o valor é nulo ou não;
* **BETWEEN:** é igual a >= x AND <= y;

**13/11:**

* **JOINS:**
* **JOIN:** intersecção de a e b
* **JOIN RIGHT:** Seleciona a intersecção de a e b e toda a parte da direita (tem relação com a tabela da direita e baseia-se nela)
* **JOIN LEFT:** Seleciona a intersecção de a e b e toda a parte da esquerda (tem relação com a tabela da esquerda e baseia-se nela)
* **INNER**: é a mesma coisa que Join, mas numa literatura mais antiga
* **PRODUTO CARTESIANO:**
* Faz a junção de todos os dados de uma tabela 1 com todos os dados de uma tabela 2
* O número de tabelas na cláusula FROM menos um (n.º tabelas – 1) é a quantidade de junções válidas que devem ser feitas
* Deve-se

**27/11:**

**FUNÇÕES DE GRUPO:**

* **COUNT:** Retorna o número de colunas da tabela (retorna Integer);
* **SUM:** Retorna a soma dos valores (retorna decimal, recebe como parâmetro um decimal – pode ser o \*);
* **AVG:** Retorna a média dos valores (retorna decimal, recebe como parâmetro um decimal, pode ser o \*);
* **MIN / MAX:** Retorna o menor e o maior valor passado como parâmetro (retorna o dado passado como parâmetro);
* **OPs RELACIONAIS:**

**< > <= >= <> =**

* **IN:** Retorna onde está a delimitação da pesquisa
* **ANY:** Indica uma condição (ex: uma data maior do que qualquer um dos valores desejados). Retorna alguns dados de determinada coluna;
* **ALL:** Retorna todos os dados de determinada coluna, é uma condição extrema;
* Não utilizar order by dentro de subconsultas;
* Não utilizar um operador de uma única linha quando a subconsulta retornar mais de um valor;