

<LAB365>

**BANCO DE DADOS INICIAL**

# AGENDA

- Introdução
- Conceitos Básicos
- Modelos de Dados

# BANCO DE DADOS INICIAL

- Por que estudar sobre Banco de Dados?
  - <https://www.menti.com/alnt5zurhpx2>



# BANCO DE DADOS INICIAL

- Armazenamento e Recuperação de Dados
  - Essencial para aplicações dinâmicas
  - Manipulação de Dados
- Performance e Escalabilidade
  - Consultas Eficientes
  - Gestão de Conexões

# BANCO DE DADOS INICIAL

- Modelagem de Dados
- Integridade e Segurança dos Dados
  - Integridade Referencial
  - Segurança dos Dados
- Backup e Recuperação

# BANCO DE DADOS INICIAL

- **Dado:** fato do mundo real que está registrado
  - exemplos: endereço, data
- **Informação:** conjunto de dados inter-relacionados com um sentido em comum
  - exemplos: endereço de entrega, idade
- **Banco de Dados (BD):** coleção de dados inter-relacionados e persistentes que representa um subconjunto dos fatos presentes em um domínio de aplicação(universo de discurso)

# BANCO DE DADOS INICIAL

- **Sistema Gerenciador de Banco de Dados (SGBD)**
  - Um SGBD consiste em uma coleção de dados inter-relacionados e em um conjunto de programas para acessá-los.
  - SGBDs são projetados para gerenciar grandes grupos de informações

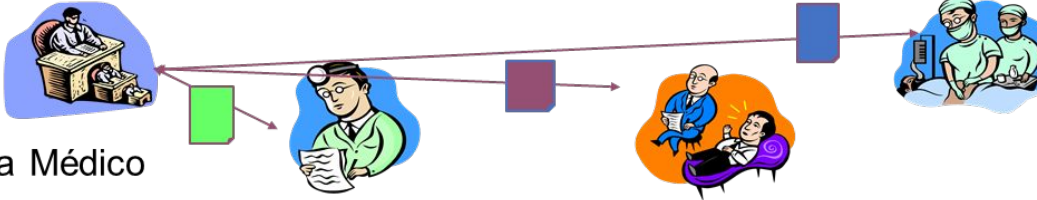
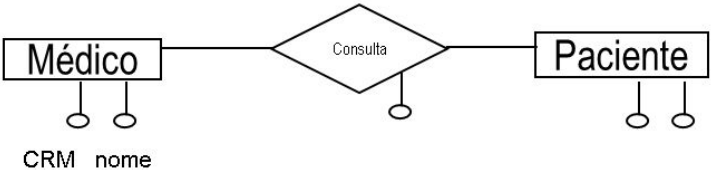
# BANCO DE DADOS INICIAL

- O gerenciamento envolve:
  - A definição de estruturas para o armazenamento da informação
  - O fornecimento de mecanismos para manipular as informações
- O grande objetivo de um SGBD é prover aos usuários uma visão **ABSTRATA** dos dados
- O sistema omite certos detalhes de como os dados são armazenados e mantidos, mas oferece mecanismos eficientes para **BUSCA** e **ARMAZENAMENTO**



# BANCO DE DADOS INICIAL

- **Modelos de Dados**
  - conjunto de conceitos que podem ser usados para descrever a estrutura de um banco de dados. Por estrutura de um banco de dados entendemos os tipos de dados, relacionamentos e restrições que devem suportar os dados.

Mundo Real	 <p>Sistema Médico</p>			
Modelo Conceitual (modelo abstrato dos dados)	<ul style="list-style-type: none"> <li>Independente do modelo de dados</li> <li>Independente do SGBD</li> </ul>			
Modelo Lógico (estrutura dos dados)	<ul style="list-style-type: none"> <li>Dependente do modelo de dados</li> <li>Independente do SGBD</li> </ul> <p>Médico (CRM, Nome)</p>	Relacional	Orientado a Objetos	Objeto-relacional
Modelo Físico	<ul style="list-style-type: none"> <li>Dependente do modelo de dados</li> <li>Dependente do SGBD</li> </ul>	<ul style="list-style-type: none"> <li>Organização física dos dados</li> <li>Estruturas de armazenamento de dados</li> <li>Índices de acesso</li> </ul>		

# BANCO DE DADOS INICIAL

- **Modelo Entidade-Relacionamento**

- O modelo Entidade-Relacionamento foi definido por Peter Chen em 1976, e teve como base a teoria relacional criada por E.F.Codd(1970). Segundo Chen, a visão de uma dada realidade, baseia-se no relacionamento entre conceitos desta realidade, os quais retratam os fatos que governam esta mesma realidade, e que cada conceito (entidade ou relacionamento) pode possuir atributos (qualificadores desta realidade).



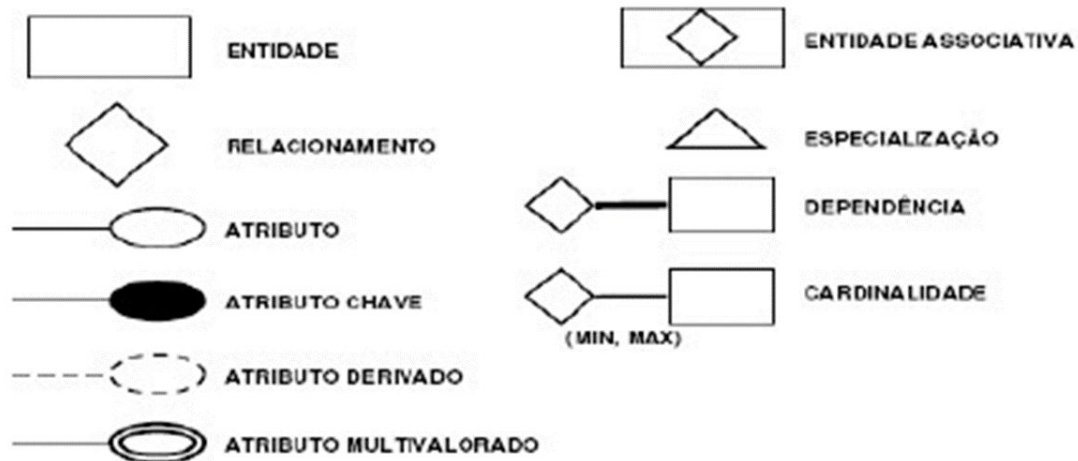
# BANCO DE DADOS INICIAL

- **Modelo Entidade-Relacionamento**

- O modelo ER propõe que a realidade seja visualizada sob três pontos de vista:
  1. Os objetos que compõe a realidade
  2. Os tipos de informação ou características que se deseja conhecer sobre os objetos que compõem a realidade e
  3. A forma como estes objetos se relacionam

# BANCO DE DADOS INICIAL

- **Modelo Entidade-Relacionamento**
  - Desta forma o MER é composto por três conceitos: *Entidade*, *Atributo* e *Relacionamento*.



# BANCO DE DADOS INICIAL

- **Modelo Entidade-Relacionamento**

- **Entidade:** Nomeamos as entidades usando substantivos que representam de forma clara e objetiva sua função. Por exemplo, Livro, Funcionário, Venda e etc.
- **Atributos:** É uma característica ou qualidade de uma entidade que tem valor para o negócio, que deve ser observada pelo usuário.
- **Relacionamentos:** Associação entre duas ou mais entidades.
  - **Cardinalidade:** É o número máximo e mínimo de ocorrências de uma entidade que estão associadas às ocorrências de outra entidade que participa do relacionamento.

## BANCO DE DADOS INICIAL

# HANDSON

# BANCO DE DADOS INICIAL

- **BrModelo:** <https://sourceforge.net/projects/brmodelo/>



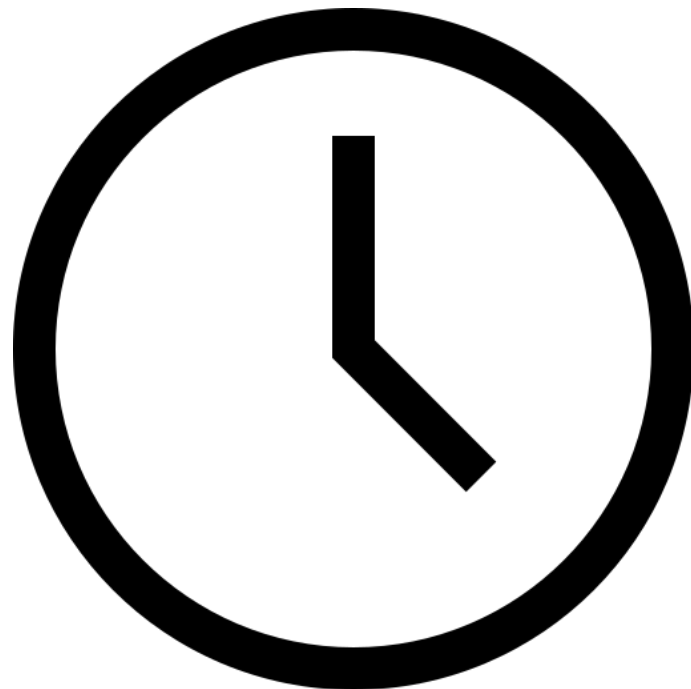
## INTERVALO!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

**Início:** 20:20

**Retorno:** 20:40



## BANCO DE DADOS INICIAL

- Uma floricultura deseja informatizar suas operações. Inicialmente, deseja manter um cadastro de todos os seus clientes, mantendo informações como: RG, nome, telefone e endereço. Deseja também manter um cadastro contendo informações sobre os produtos que vende, tais como: nome do produto, tipo (flor, vaso, planta,...), preço e quantidade em estoque. Quando um cliente faz uma compra, a mesma é armazenada, mantendo informação sobre o cliente que fez a compra, a data da compra, o valor total e os produtos comprados.

# BANCO DE DADOS INICIAL

- **Modelo de Dados Relacional**

- Em um projeto de banco de dados, a segunda etapa é construir um modelo lógico. O modelo lógico é desenvolvido a partir do modelo conceitual. O modelo traz uma visão de como será implementado no BD.

# BANCO DE DADOS INICIAL

- **Modelo de Dados Relacional**

- Nessa etapa, já é necessário se preocupar com aspectos de nomenclatura e restrições mais completas de consistências e integridade. O modelo lógico também é chamado de modelo relacional.

# BANCO DE DADOS INICIAL

- **Modelo de Dados Relacional**
  - Conversões de Nomenclatura

Modelo Conceitual	Modelo Lógico
Entidade	Tabelas
Atributo	Coluna
Relacionamentos	Chave Estrangeira

# BANCO DE DADOS INICIAL

- **Modelo de Dados Relacional**

- Dentro do modelo conceitual é utilizado o conceito de atributos identificadores. No modelo relacional, esse conceito se transporta para **chave primária**.
- A **chave primária** é utilizada para identificar unicamente cada registro de dados de uma tabela. Também é utilizada para formalizar os relacionamentos entre as tabelas, portanto a definição correta é muito importante!

# BANCO DE DADOS INICIAL

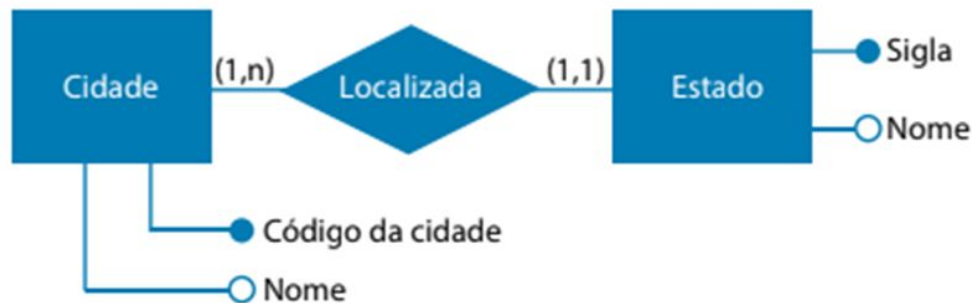
- **Modelo de Dados Relacional**

- É necessário reforçar algumas características essenciais das chaves primárias:
  - ser unívoca, ou seja, ter um único valor para cada registro.
  - ser não nula, ou seja, a coluna não pode estar vazia.
  - ser não redundante, no caso de uma chave primária composta, não devem ser incluídos mais atributos do que o mínimo necessário.

# BANCO DE DADOS INICIAL

- **Modelo de Dados Relacional**

- Para entendermos o conceito de chave estrangeira vamos usar o modelo abaixo.



Davi Leon (2019)

Figura 16 - Representação gráfica do modelo conceitual de duas entidades

Fonte: Do autor (2019)



# BANCO DE DADOS INICIAL

- **Modelo de Dados Relacional**

- No modelo lógico, o relacionamento irá se transformar em duas entidades ligadas por uma chave estrangeira.

ESTADO		CIDADE		
SIGLA	NOME	CÓDIGO	NOME	SIGLA_ESTADO
RJ	Rio de Janeiro	1	Búzios	RJ
PR	Paraná	2	Fortaleza	CE
AC	Acre	3	Rio Branco	AC
AM	Amazonas	4	Manaus	AM
CE	Ceará	5	Niterói	RJ
		6	Pato Branco	PR

Dani Leon (2019)

Figura 17 - Representação do modelo lógico de duas tabelas relacionadas

Fonte: Do autor (2019)

## BANCO DE DADOS INICIAL

# HANDSON

<LAB365>

**BANCO DE DADOS INICIAL**

<LAB365>



# AGENDA

- Modelagem de dados (continuação)
- Linguagem SQL

# BANCO DE DADOS INICIAL

- **Modelo de Dados Relacional**

- Integridade Referencial: são regras checadas a todo o momento que um dado vinculado a um relacionamento é modificado, seja na sua tabela de origem ou na tabela de destino.

# BANCO DE DADOS INICIAL

- **Linguagem SQL**

- A linguagem SQL é o recurso mais conhecido por DBAs e programadores para a execução de comandos em bancos de dados relacionais. É por meio dela que criamos tabelas, colunas, índices, atribuímos permissões a usuários, bem como realizamos consultas a dados. Enfim, é utilizando a SQL que “conversamos” com o banco de dados.

# BANCO DE DADOS INICIAL

- **Linguagem SQL**
  - A linguagem SQL é organizada em subconjuntos, cada um com propósitos bem definidos.
  - **DQL** - *Linguagem de Consulta de Dados* - Define o comando utilizado para que possamos consultar (SELECT) os dados armazenados no banco;

# BANCO DE DADOS INICIAL

- **Linguagem SQL**

- **DML** - *Linguagem de Manipulação de Dados* - Define os comandos utilizados para manipulação de dados no banco (INSERT, UPDATE e DELETE);
- **DDL** - *Linguagem de Definição de Dados* - Define os comandos utilizados para criação (CREATE) de tabelas, views, índices, atualização dessas estruturas (ALTER), assim como a remoção (DROP);



# BANCO DE DADOS INICIAL

- **Linguagem SQL**

- **DCL** - *Linguagem de Controle de Dados* - Define os comandos utilizados para controlar o acesso aos dados do banco, adicionando (GRANT) e removendo (REVOKE) permissões de acesso;
- **DTL** - *Linguagem de Transação de Dados* - Define os comandos utilizados para gerenciar as transações executadas no banco de dados, como iniciar (BEGIN) uma transação, confirmá-la (COMMIT) ou desfazê-la (ROLLBACK).

# SQL

```
graph TD; SQL[SQL] -.-> SUBCONJUNTOS_SQL[SUBCONJUNTOS SQL]; SUBCONJUNTOS_SQL -.-> DQL[DQL]; SUBCONJUNTOS_SQL -.-> DML[DML]; SUBCONJUNTOS_SQL -.-> DDL[DDL]; SUBCONJUNTOS_SQL -.-> DCL[DCL]; SUBCONJUNTOS_SQL -.-> DTL[DTL]; DQL -.-> COMANDOS_SQL[COMANDOS SQL]; DML -.-> COMANDOS_SQL; DDL -.-> COMANDOS_SQL; DCL -.-> COMANDOS_SQL; DTL -.-> COMANDOS_SQL; COMANDOS_SQL -.-> SELECT[SELECT]; COMANDOS_SQL -.-> INSERT_UPDATE_DELETE[INSERT<br/>UPDATE<br/>DELETE]; COMANDOS_SQL -.-> CREATE_ALTER_DROP[CREATE<br/>ALTER<br/>DROP]; COMANDOS_SQL -.-> GRANT_REVOKE[GRANT<br/>REVOKE]; COMANDOS_SQL -.-> BEGIN_COMMIT_ROLLBACK[BEGIN<br/>COMMIT<br/>ROLLBACK];
```

SUBCONJUNTOS SQL

**DQL**

**DML**

**DDL**

**DCL**

**DTL**

COMANDOS SQL

SELECT

INSERT  
UPDATE  
DELETE

CREATE  
ALTER  
DROP

GRANT  
REVOKE

BEGIN  
COMMIT  
ROLLBACK

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DDL**
  - Uma tarefa indispensável, porém realizada com menos frequência, é criar o banco de dados. Em seguida, é natural a criação das tabelas a ele relacionadas. Para isso, lidamos com os comandos CREATE DATABASE e CREATE TABLE da SQL.

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DDL**

- CREATE DATABASE nome\_do\_banco\_de\_dados
- DROP DATABASE nome\_do\_banco\_de\_dados
- CREATE TABLE nome\_da\_tabela(
  - campo1 Tipo,
  - campo2 Tipo,
  - campo3 Tipo)

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DDL**

- ALTER TABLE Nome\_Da\_Tabela ADD Nome\_Coluna Tipo\_Coluna
- ALTER TABLE Nome\_Da\_Tabela MODIFY Nome\_Coluna  
Tipo\_Coluna
- ALTER TABLE Nome\_Da\_Tabela DROP Nome\_Coluna Tipo\_Coluna

## BANCO DE DADOS INICIAL

# HANDSON

## BANCO DE DADOS INICIAL

- Escrever um comando de SQL que permita criar uma tabela com o nome Caixa\_Postal, capaz de armazenar um inteiro de até quatro dígitos e uma string com 45 caracteres.
- Escrever um comando de SQL que permita criar uma tabela com o nome Pessoa, com o seguintes atributos: ID, Nome, Idade, Salario, Telefone e Código Postal).

# BANCO DE DADOS INICIAL

- **Linguagem SQL - Constraints**

- As restrições SQL são usadas para especificar regras para dados em uma tabela. As restrições podem ser especificadas quando a tabela é criada com a instrução CREATE TABLE ou depois que a tabela é criada com a instrução ALTER TABLE.



# BANCO DE DADOS INICIAL

- **Linguagem SQL - Constraints**

- *NOT NULL* - Garante que uma coluna não pode ter um valor NULL
- *UNIQUE* - Garante que todos os valores em uma coluna sejam diferentes
- *PRIMARY KEY*- Uma combinação de NOT NULL e UNIQUE. Identifica exclusivamente cada linha em uma tabela
- *FOREIGN KEY* - Evita ações que destruiriam links entre tabelas

# BANCO DE DADOS INICIAL

- **Linguagem SQL - Constraints**
  - *CHECK* - Garante que os valores em uma coluna satisfaçam uma condição específica
  - *DEFAULT* - Define um valor padrão para uma coluna se nenhum valor for especificado

## BANCO DE DADOS INICIAL

# HANDSON

# BANCO DE DADOS INICIAL

- Considere o seguinte exemplo: Criar um banco de dados Clínica com as seguintes características:
  - **Tabela Médicos**
    - Atributo CRM: caractere, único e não vazio
    - Atributo Nome: caractere e não vazio
    - Atributo Idade: inteiro e não poderá ser maior que 23 e menor que 70
    - Atributo Especialidade: caractere e não poderá possuir especialização em Ortopedia
  - **Tabela Paciente**
    - Atributo CPF: caractere e único
    - Atributo Nome: caractere e não vazio
    - Atributo Doença: caractere e não poderá ter valores como fratura e torção

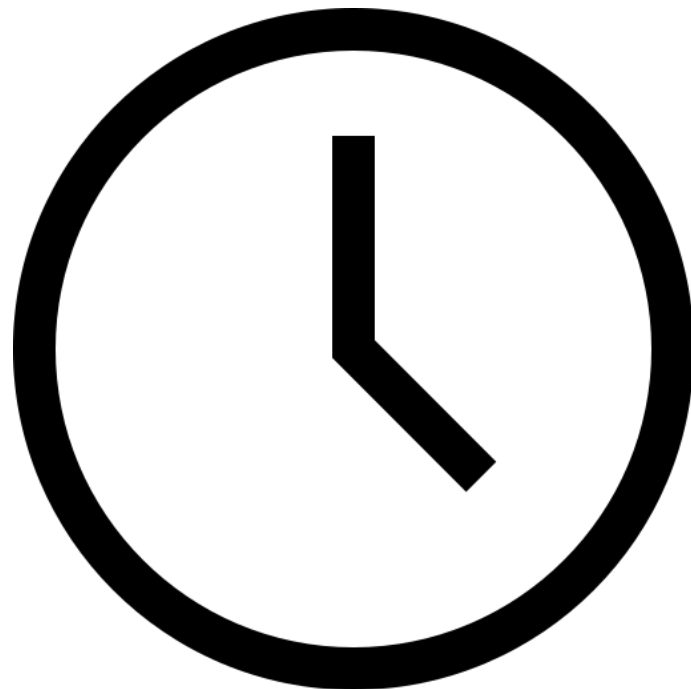
## INTERVALO!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

**Início:** 20:20

**Retorno:** 20:40



# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**

- Os comandos de manipulação de dados(DML) em SQL são representados por:
- *INSERT*: permite a inclusão de novas linhas nas tabelas
- *UPDATE*: altera os valores de dados já cadastrados
- *DELETE*: remove dados já cadastrados

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**

- INSERT INTO NOME DA TABELA(coluna1,coluna2,coluna3) VALUES (valor1, valor2, valor3)
- Exemplos:
  - INSERT INTO Cliente(codigo,nome,sexo) VALUES("200810", "Regilan Meira", "Masculino")
  - INSERT INTO Disciplina(codigo,nome,ementa) VALUES("01", "Banco de Dados", "DER,Modelo Relacional,SQL")

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**

- O comando UPDATE é usado para mudar valores de linhas de dados que já foram cadastrados anteriormente e que obedecem a determinados critérios, especificados em condições. Este comando pode alterar mais de uma linha ao mesmo tempo, caso mais de uma linha obedeça a determinada condição. As condições podem também ser representadas utilizando os operadores: AND, OR e NOT.



# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**
  - O comando UPDATE, contém a cláusula WHERE, de forma a restringir o conjunto dos registros que serão processados pelo comando. Se não for colocada a cláusula WHERE no comando UPDATE, as alterações serão realizadas em todos os registros da tabela.

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**
  - UPDATE NOME DA TABELA SET coluna1 = valor1, coluna2 = valor2  
WHERE condições
  - Exemplo
    - UPDATE Avaliacao SET media = 10

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**

- UPDATE Avaliacao SET media = 10 WHERE nome\_aluno = "João"
- UPDATE Compras SET preco = 105, forma\_pagamento = "Cartão de Crédito" WHERE numero\_compra = "2008708"

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**
  - **Situação 01: Aumentar o salário de todos os funcionários em 10%.**
    - Como se pretende aumentar o salário de todos os elementos da tabela FUNCIONÁRIO, o comando UPDATE não usará a cláusula WHERE.

```
UPDATE Funcionario SET salario = salario * 1.1
```

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**
  - **Situação 02: Aumentar o salário do funcionário Regilan Meira e adicionar 1 ano ao tempo de serviço.**
    - Nessa situação, estamos restringindo a atualização para o funcionário REGILAN MEIRA, sendo assim faz-se necessário o uso da cláusula WHERE.

```
UPDATE Funcionario SET salario = salario * 1.1, idade =  
idade + 1 WHERE nome = "Regilan Meira"
```

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**
  - **Situação 03: Adicionar R\$ 150 no salário das mulheres que possuem filhos, ou homens que são casados.**
    - Nessa situação, utilizaremos a cláusula WHERE , juntamente com o operador AND e OR.

```
UPDATE Funcionarios SET Salario = Salario + 150 WHERE  
(Sexo = "F" and Filhos > 0) OR (Sexo = "M" and EstadoCivil =  
"Casado")
```

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**

- O comando DELETE é usado para remover linhas de uma tabela. Este comando pode remover mais de uma linha ao mesmo tempo, caso mais de uma linha obedeça a uma certa condição. As condições podem ser representadas utilizando os operadores AND, OR e NOT.

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**
  - O comando DELETE, contém a cláusula WHERE, de forma a restringir o conjunto dos registros que serão processados pelo comando. Se não for colocada a cláusula WHERE no comando DELETE, serão apagados todos os registros de uma tabela.



# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**

- Assim como no comando UPDATE, podemos utilizar os operadores relacionais (>, >=, <, <=, =, <>, like) e os operadores lógicos (AND, OR) para especificar as condições de exclusão de dados.

# BANCO DE DADOS INICIAL

- **Linguagem SQL - DML**

- DELETE FROM NOME DA TABELA WHERE <condições>
- Exemplos
- DELETE FROM ESCOLA
- DELETE FROM ESCOLA WHERE ALUNO = "TIAGO PEREIRA"
- DELETE FROM PRODUTOS WHERE NOME Like "LEITE%"
- DELETE FROM CLIENTES WHERE QuantidadeCompras <= 3

<LAB365>

**BANCO DE DADOS INICIAL**

<LAB365>



# AGENDA

- Linguagem SQL

# BANCO DE DADOS INICIAL

- **SELECT - DQL**

- Visualizando os dados inseridos dentro de uma tabela
  - Utilizaremos o comando:
  - **SELECT** *column1, column2, ...* **FROM** *table\_name*;
- Podemos visualizar apenas colunas específicas, passando o nome das suas referências ou podemos escolher todas as colunas usando o “\*”
- Podemos passar “apelidos” para as nossas colunas, chamado de “alias” através do “as”
  - exemplo:
  - **select** login **as** username, ucloudIdentifier **as** ucloud\_id **from** user;

# BANCO DE DADOS INICIAL

- **SELECT - DQL**

- Podemos filtrar a busca pelos dados do Select
  - where
  - like (%texto%)
  - IgnoreCase SQL -> `SELECT * FROM people WHERE nome LIKE 'RAFAEL%';`

# BANCO DE DADOS INICIAL

- **Operador 'LIKE' e patterns**

- Exemplos

```
SELECT * FROM pessoas p  
WHERE p.nome LIKE 'Michael' ;
```

```
SELECT * FROM pessoas p  
WHERE p.nome LIKE 'Mi%' ;
```

```
SELECT * FROM pessoas p  
WHERE p.nome LIKE '%ch%' ;
```

# BANCO DE DADOS INICIAL

- A instrução **SELECT** possui as seguintes cláusulas:
  - Selecione linhas distintas usando o operador **DISTINCT**.
  - Classifique as linhas usando a cláusula **ORDER BY**.
  - Filtre linhas usando a cláusula **WHERE**.
  - Agrupe linhas em grupos usando a cláusula **GROUP BY**
  - Filtre grupos usando a cláusula **HAVING**.



# BANCO DE DADOS INICIAL

- Operadores de comparação
- Operadores Lógicos
  - AND, OR, NOT

Operadores Relacionais	
Operador	Operação
=	Igual a
<>	Diferente de
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

# BANCO DE DADOS INICIAL

- **ORDER BY**

- Após uma consulta com **SELECT**, o SGBD retorna as linhas em uma ordem não especificada. Para classificar o conjunto de resultados, use a cláusula **ORDER BY** na instrução **SELECT**.
- A cláusula **ORDER BY** permite que você classifique as linhas retornadas de uma instrução **SELECT** em ordem crescente ou decrescente com base nos critérios especificados.
- Exemplo:
  - **SELECT DISTINCT** nome, celular **FROM** Aluno **ORDER BY** nome **ASC**;

# BANCO DE DADOS INICIAL

- A seguir, a sintaxe da cláusula ORDER BY:

```
SELECT
    column_1,
    column_2
FROM
    table_name
ORDER BY
    column_1 [ASC | DESC],
    column_2 [ASC | DESC];
```

- Primeiro, especifique uma coluna ou expressão que deseja classificar na cláusula ORDER BY. Se você classificar o conjunto de resultados com base em várias colunas ou expressões, use uma vírgula para separar duas colunas ou expressões.
- Segundo, use o ASC para classificar o conjunto de resultados em ordem crescente e DESC para classificar o conjunto de resultados em ordem decrescente.
- Se pular a opção ASC ou DESC, o ORDER BY utilizará o ASC por padrão.

# BANCO DE DADOS INICIAL

- **Funções de Agregação**

- Agrupam as linhas encontradas pela consulta e retorna apenas uma linha como resultado.

Vamos aprender as seguintes funções de agregação:

- 1) **AVG**: Calcula média aritmética;
- 2) **COUNT**: retorna o total de linha de acordo com a QUERY realizada;
- 3) **MAX**: retorna o registro de maior valor de acordo com a QUERY realizada;
- 4) **MIN**: retorna o registro de menor valor de acordo com a QUERY realizada;
- 5) **SUM**: retorna a soma de valores de acordo com a QUERY realizada.

# BANCO DE DADOS INICIAL

- **Funções de Agregação - AVG**

- Argumento numérico;
- Retorno numérico decimal;
- Calcula a média aritmética sobre o conjunto de linhas;

```
SELECT AVG(nota) FROM Nota
```

# BANCO DE DADOS INICIAL

- **Funções de Agregação - COUNT**

- Argumento qualquer;
- Retorno numérico inteiro;
- Retorna o número de linhas de entrada para o qual a expressão é não nula ou simplesmente o total caso seja especificado "\*";

`SELECT COUNT(*) FROM Cliente`

# BANCO DE DADOS INICIAL

- **Funções de Agregação - MAX**

- Argumento numérico, texto ou data/hora;
- Retorno mesmo tipo de dado do argumento;
- Retorna o maior valor entre todos os fornecidos;

`SELECT MAX(nascimento) FROM Cliente`

# BANCO DE DADOS INICIAL

- **Funções de Agregação - MIN**

- Argumento numérico, texto ou data/hora;
- Retorno mesmo tipo de dado do argumento;
- Retorna o menor valor entre todos os fornecidos;

`SELECT MIN(nascimento) FROM Cliente`



# BANCO DE DADOS INICIAL

- **Funções de Agregação - SUM**

- Argumento numérico;
- Retorno numérico inteiro ou decimal;
- Retorna o soma da expressão sobre as linhas de entrada;

`SELECT SUM(valor) FROM Venda`

# BANCO DE DADOS INICIAL

- **GROUP BY**

- Conseguimos agrupar determinados dados para possibilitar que funções de agregação resultem em mais de uma linha
- Exemplo: extrair relatório de gasto de folha de pagamento por departamento de uma empresa
  - Precisamos agrupar os funcionários dentro do seu departamento
  - Precisamos somar os salários dos funcionários

Group by: Departamento.id

Sum: Funcionario.salario

# BANCO DE DADOS INICIAL

```
SELECT working_area, COUNT(*)  
FROM agents  
GROUP BY working_area;
```

agents

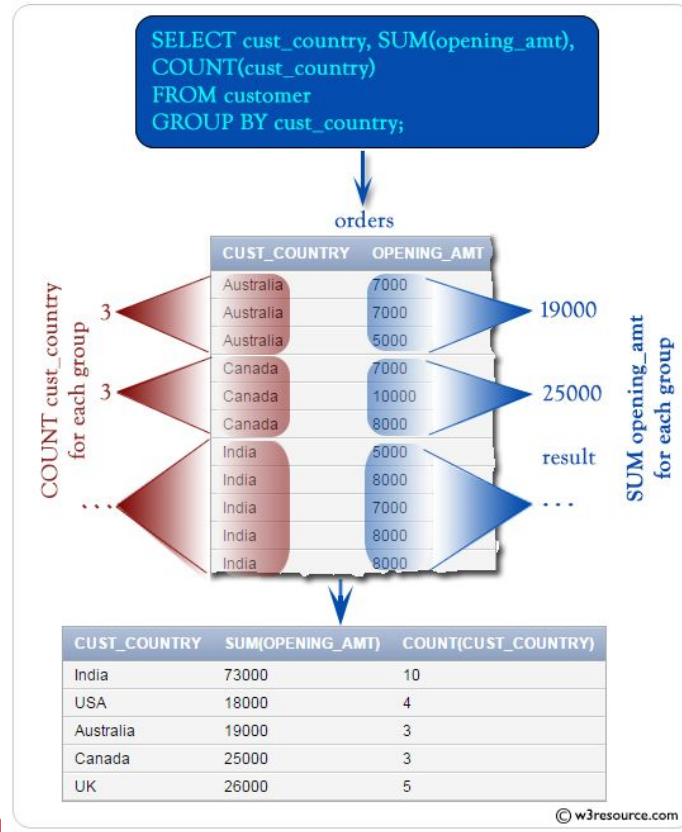
the working\_area have  
been grouped and appearing  
once

AGENT_NAME	WORKING_AREA
Alex	London
Subbarao	Bangalore
Benjamin	Hampshair
Ramasundar	Bangalore
Alford	New York
Ravi Kumar	Bangalore
Santakumar	Chennai
Lucida	San Jose
Anderson	Brisban
Mukesh	Mumbai
McDen	London
Ivan	Toronto

WORKING_AREA	COUNT(*)
San Jose	1
Toronto	1
London	2
Hampshair	1
New York	1
Brisban	1
Bangalore	3
Chennai	1
Mumbai	1

result

# BANCO DE DADOS INICIAL



# BANCO DE DADOS INICIAL

- **ORDEM DE ESCRITA**

- 1- SELECT
- 2- FROM
- 3- WHERE
- 4- GROUP BY
- 5- HAVING
- 6- ORDER BY

- **ORDEM DE EXECUÇÃO**

- 1- FROM
- 2- WHERE
- 3- GROUP BY
- 4- HAVING
- 5- SELECT
- 6- ORDER BY

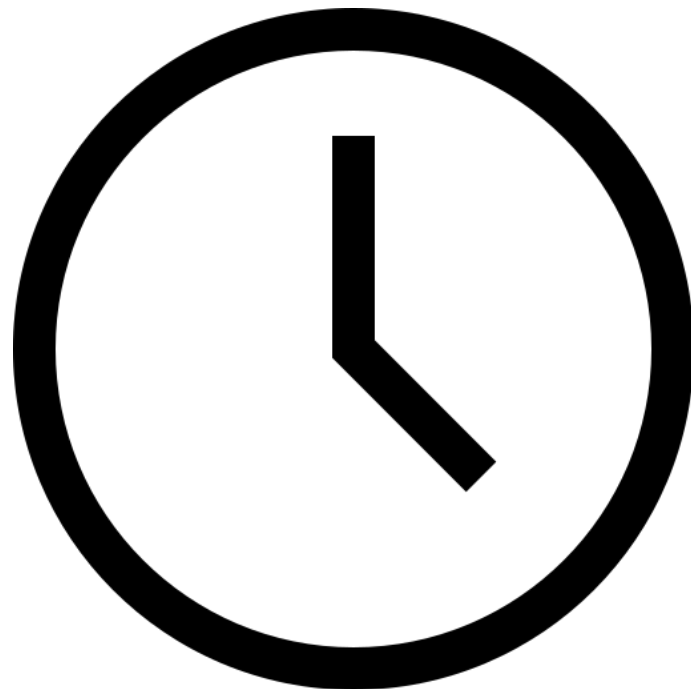
## INTERVALO!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

**Início:** 20:20

**Retorno:** 20:40



**BANCO DE DADOS INICIAL**

**PLANTÃO TIRA-DÚVIDAS**

## AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique [aqui](#) ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.





<LAB365>

<LAB365>

**SENAI**