Esclarecimento

Licenciamento de Uso

BANCOS DE DADOS

Introdução

Bancos de dados são ferramentas que permitem o armazenamento e manipulação de dados em tabelas (conjuntos de informações com estrutura regular).
Exemplos de bancos de dados: **Sistemas de Processamento de arquivos** (fichas impressas, documentos do Word), **tabelas SQL** armazenadas em um servidor.

1. TIPOS DE BANCOS DE DADOS

• **Banco de Dados Não Relacionais** – Modo regular, os arquivos são escritos de forma sequencial, o acesso geralmente é mais lento em comparação ao banco de dados Relacional.

• **Banco de Dados Relacional** – Os dados são organizados em tabelas permitindo o relacionamento entre as mesmas. Uma relação trata-se de associação entre varias *entidades*[1]

Exemplo: podemos cruzar os dados entre alunos por curso ou turma ao relacionarmos as Tabela Cursos e Tabela Alunos. Em comparação ao Modelo Não Relacional, podemos citar como principais vantagens**:** padrão adotado mundialmente, maior velocidade de acesso aos dados e menor espaço de armazenamento.
A figura 1 representa alguns dos campos que podem estar contidos nas Tabelas Cursos e Alunos.

Tabelas Forma de organizar os dados em linhas e colunas.
Colunas Campos que formam um registro
O Conjunto formado pelo encontro de uma linha/coluna é denominado tupla.

[1] MER (Modelo entidade/relacionamento)

1.1 Estruturas existentes em bancos de dados

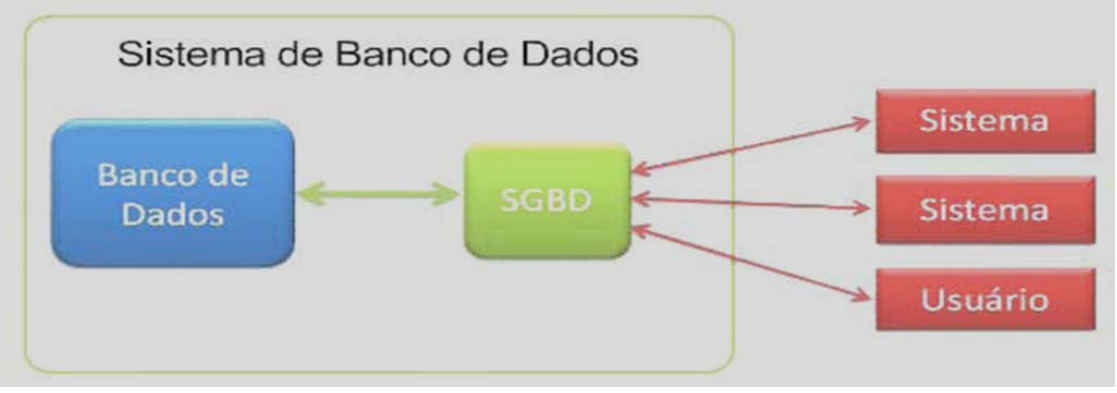
• **Visões** => Consultas SQL previamente programadas disponíveis para rápido acesso, não sevem para armazenar dados, sua função é armazenar critérios de seleção de dados, permitem dados atualizados sempre que as tabelas em questão sofrem alteração.

• **Índices** => Estruturas que gerenciam a ordenação de valores dos campos informados para melhorar a performance de processamento do banco de dados sobre estes campos e seus respectivos registros.

2. DATABASE MANAGEMENT SYSTEM (SISTEMA GERENCIADOR DE BANCO DE DADOS)

O sistema de gerenciamento de banco de dados não deve ser confundido com o próprio banco de dados; a função de gravar uma informação, alterá-la ou até mesmo recuperá-la é do banco de dados, cabe ao sistema de gerenciamento permitir ou não o acesso ao banco de dados. O sistema de gerenciamento pode não trazer grandes benefícios a bancos de dados pequenos, simples e de pouco acesso, ele é vital para bancos de dados com grande volume de informações e com acessos simultâneos por vários usuários, o controlador de acesso gerencia todas estas operações, evitando assim, inconsistência nas informações.

Como podemos perceber, o sistema de gerenciamento é um complemento ao banco de dados, interligando as requisições de conexão dos usuários com o banco de dados.



As requisições podem ser enviadas por usuários específicos, ou através de sistemas online.

3. SQL – STRUCTURED QUERY LANGUAGE

Linguagem Estruturada de Consulta => Desenvolvida originalmente na década de 70, nos laboratórios da IBM, atualmente é o padrão adotado em banco de dados mundialmente. O SQL se diferencia das outras linguagens utilizadas para bancos de dados por especificar a forma do resultado e não o "caminho" para se chegar a ele é uma linguagem declarativa em oposição a outras linguagens procedurais.

Os recursos disponibilizados pelo SQL são agrupados em cinco funcionalidades:

Data Definition Language => Linguagem de Definição de Dados, por meio dela podemos definir estruturas do banco tais como: tabelas, visões, sequenciais e outras estruturas. Comandos da DDL: CREATE (criação de estrutura), ALTER (alterar estrutura) e DROP (permite remover ou excluir uma estrutura).

Data Manipulation Language => Linguagem de Manipulação de Dados, permite inserir, alterar e excluir informações nos objetos construídos pela DDL. A DML possui três comandos: INSERT (inserir dados), UPDATE (alterar informações) e DELETE (excluir dados).

Data Query Language => Linguagem de Consulta de Dados, complementa e permite recuperar e ler dados na estrutura do banco de dados, comando utilizado: SELECT possibilita a ordenação e agrupamento dos dados, cálculos (funções aritméticas) e filtros de seleção.

Data Control Language => Linguagem de Controle de Dados, gerencia as permissões de quem pode acessar o banco de dados, o que cada usuário tem acesso, quem pode ou não alterar e remover dados, dentre outras operações. Comandos da DCL: GRANT (habilita acesso a dados e operações) e REVOKE (revoga o acesso a dados e operações).

Data Transaction Language => Linguagem de Transação de Dados, permite operações conjuntas sendo estas iniciadas pela **START TRANSACTION***, seguidas pela **COMMIT** (concretiza a transação) que informa ao servidor que a transação foi concluída com sucesso ou **ROLLBACK** que anula a transação.

* No SQL Server o comando START deve ser substituído por BEGIN

4. NORMALIZAÇÃO DE DADOS (RELACIONAMENTOS E CHAVES)

Normalização de dados é um termo que está intimamente ligado a Relacionamentos, que por sua vez é ligado a chaves.

• **Relacionamentos** => São ligações entre tabelas onde existes um ou mais campos em comum entre as tabelas relacionadas conhecidos como campos chaves.

• **Campos Chaves** => São valores que apresentam "referência" de uma tabela em outra (chave de identificação).

As tabelas dos bancos de dados são compostas por linhas e colunas, sendo que algumas das colunas podem apresentar características especificas de acordo com a forma com que a tabela foi construída ou da aplicação que será construída sob o banco de dados, geralmente algumas colunas são criadas especificamente para gerar relacionamentos entre duas ou mais tabelas, tais colunas são denominadas "colunas chaves".

4.1 Tipos de Chaves

1. Chaves Primárias (Primary Key – PK), os valores são únicos (não se repetem nos registros da tabela). Esta característica também é denominada de identificador único.

NOME	CPF	ESTADO
Fernando	111.111.111-11	PR
Guilherme	222.222.222-22	SP
Mateus	333.333.333-33	RJ

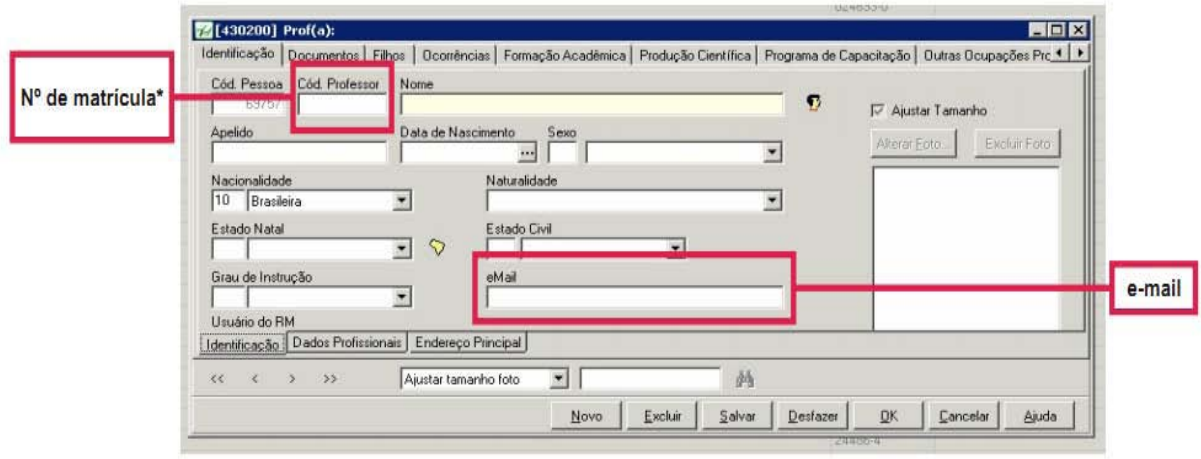
Neste exemplo o nº de CPF é utilizado como chave primária, não pode haver duas pessoas com o mesmo número de CPF. Em uma tabela de cadastro de alunos, o mesmo se aplica ao número de RA (Registro de Aluno)

2. Chave Composta é formada pela composição de duas ou mais colunas para gerar um identificador único, podendo significar que nenhum campo isoladamente possa se tornar a chave primária, sendo necessário dois ou mais campos para gerar uma combinação única de cada registro.

NOME	CPF	ESTADO
Fernando	111.111.111-11	PR
Guilherme	222.222.222-22	SP
Mateus	333.333.333-33	RJ

Usando a mesma tabela do exemplo anterior, neste caso, se o nº de CPF dependesse de cada estado, a identificação seria composta pelas

3. Chave Estrangeira (Foreign Key – FK), modelo adotado onde a coluna armazena a chave primária de outra tabela (referência), não sendo necessário preencher todas as informações que são “buscadas” através do código ou informação atrelada.



Neste exemplo, a **tela** de cadastro de Professor do **RM Classis** utiliza uma tabela que armazena os dados utilizando o número de matrícula como uma chave estrangeira, sendo que, este mesmo campo é a chave primária no cadastro de funcionários (RH)

4.2 Tipos de Relacionamentos

a) Relacionamento 1 para 1 (1:1), para cada registro na primeira tabela existe no máximo um correspondente na segunda tabela, e vice-versa.

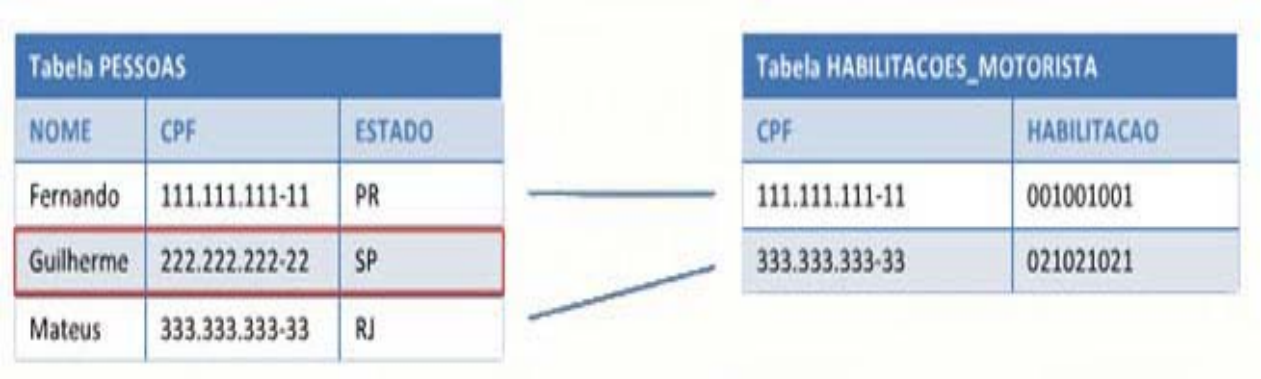


Tabela PESSOAS

NOME	CPF	ESTADO
Fernando	111.111.111-11	PR
Guilherme	222.222.222-22	SP
Mateus	333.333.333-33	RJ

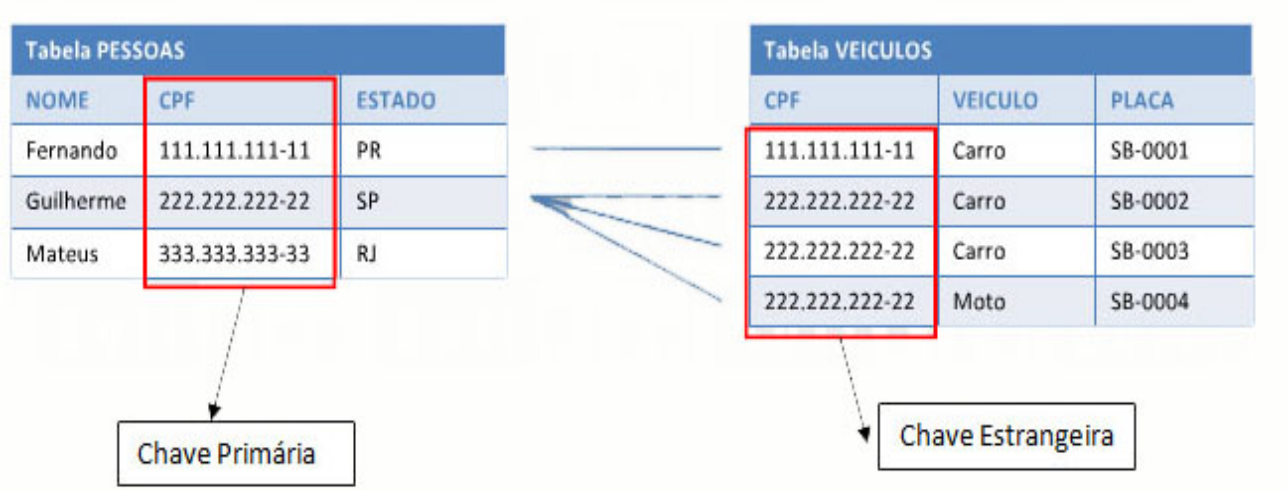
Tabela HABILITACOES_MOTORISTA

CPF	HABILITACAO
111.111.111-11	001001001
333.333.333-33	021021021

Neste exemplo, existe um único correspondente na tabela 2 para a tabela 1 em virtude de cada pessoa possuir um número específico de CPF.

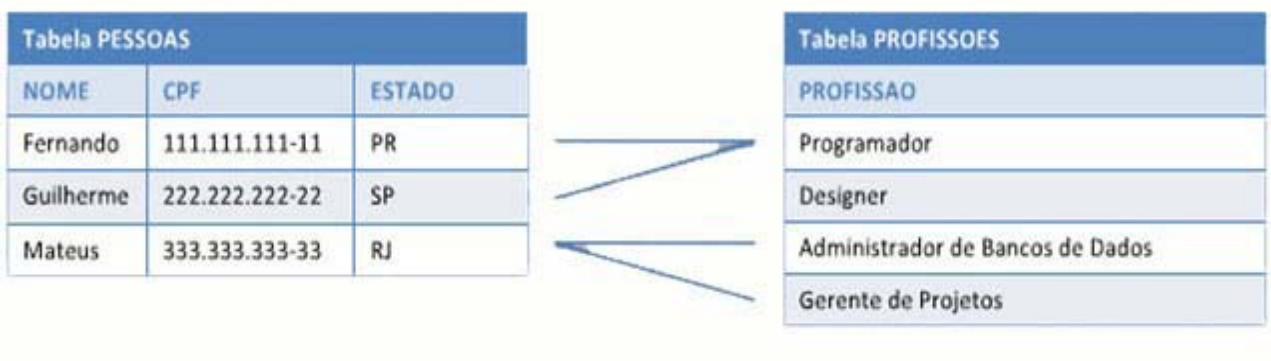
b) Relacionamento 1 para muitos (1:*), neste tipo de relacionamento, para cada registro na primeira tabela podem haver dois ou mais correspondentes na segunda tabela, mas para cada registro na segunda tabela deve haver apenas um correspondente na primeira tabela.

Neste exemplo, um indivíduo pode ser proprietário de mais de um automóvel, sendo possível também que outra pessoa não possua nenhum automóvel registrado em seu número de CPF



c) Relacionamento muitos para muitos (*:*), para cada registro da primeira tabela podem existir um ou mais correspondentes na segunda tabela, e vice-versa.

Neste exemplo, uma pessoa pode exercer mais de uma profissão, assim como, uma determinada profissão possa ser exercida por mais de uma pessoa



4.3 Tipos de Dados SQL

Abaixo segue uma relação dos tipos de dados básicos do SQL Server, sendo que os tipos que estiverem marcados com * somente funcionam a partir do SQL Server 2000

• **TINYINT**: Valores numéricos inteiros variando de 0 até 256

• **SMALLINT**: Valores numéricos inteiros variando de –32.768 até 32.767

• **INT**: Valores numéricos inteiros variando de -2.147.483.648 até 2.147.483.647

• ***BIGINT**: Valores numéricos inteiros variando de –92.23.372.036.854.775.808 até 9.223.372.036.854.775.807

• **BIT**: Somente pode assumir os valores 0 ou 1. Utilizado para armazenar valores lógicos.

• **DECIMAL(I,D) e NUMERIC(I,D)**: Armazenam valores numéricos inteiros com casas decimais utilizando precisão. **I** deve ser substituído pela quantidade de dígitos total do número e **D** deve ser substituído pela quantidade de dígitos da parte decimal (após a vírgula). DECIMAL e NUMERIC possuem a mesma funcionalidade, porém DECIMAL faz parte do padrão ANSI e NUMERIC é mantido por compatibilidade. Por exemplo, DECIMAL(8,2) armazena valores numéricos decimais variando de – 999999,99 até 999999,99

*** Lembrando sempre** que o SQL Server internamente armazena o separador decimal como ponto (.) e o separador de milhar como vírgula (,). Essas configurações INDEPENDEM de como o Windows está configurado no painel de controle e para DECIMAL E NUMERIC, somente o separador decimal (.) é armazenado.

• **SMALLMONEY**: Valores numéricos decimais variando de -214.748,3648 até 214.748,3647

• **MONEY**: Valores numéricos decimais variando de -922.337.203.685.477,5808 até 922.337.203.685.477,5807

• **REAL:** Valores numéricos aproximados com precisão de ponto flutuante, indo de -3.40E + 38 até 3.40E + 38

• **FLOAT**: Valores numéricos aproximados com precisão de ponto flutuante, indo de -1.79E + 308 até 1.79E + 308

• **SMALLDATETIME**: Armazena hora e data variando de 1 de janeiro de 1900 até 6 de junho de 2079. A precisão de hora é armazenada até os segundos.

• **DATETIME**: Armazena hora e data variando de 1 de janeiro de 1753 até 31 de Dezembro de 9999. A precisão de hora é armazenada até os centésimos de segundos.

• **CHAR(N)**: Armazena N caracteres fixos (até 8.000) no formato não Unicode. Se a quantidade de caracteres armazenada no campo for menor que o tamanho total especificado em N, o resto do campo é preenchido com espaços em branco.

• **VARCHAR(N)**: Armazena N caracteres (até 8.000) no formato não Unicode. Se a quantidade de caracteres armazenada no campo for menor que o tamanho total especificado em N, o resto do campo não é preenchido.

• **TEXT**: Armazena caracteres (até 2.147.483.647) no formato não Unicode. Se a quantidade de caracteres armazenada no campo for menor que 2.147.483.647, o resto do campo não é preenchido. Procure não utilizar este tipo de dado diretamente, pois existem funções específicas para trabalhar com este tipo de dado.

• **NCHAR(N)**: Armazena N caracteres fixos (até 4.000) no formato Unicode. Se a quantidade de caracteres armazenada no campo for menor que o tamanho total especificado em N, o resto do campo é preenchido com espaços em branco.

• **NVARCHAR(N)**: Armazena N caracteres (até 4.000) no formato Unicode. Se a quantidade de caracteres armazenada no campo for menor que o tamanho total especificado em N, o resto do campo não é preenchido.

• **NTEXT**: Armazena caracteres (até 1.073.741.823) no formato Unicode. Se a quantidade de caracteres armazenada no campo for menor que 1.073.741.823, o resto do campo não é preenchido. Procure não utilizar este tipo de dado diretamente, pois existem funções específicas para trabalhar com este tipo de dado.

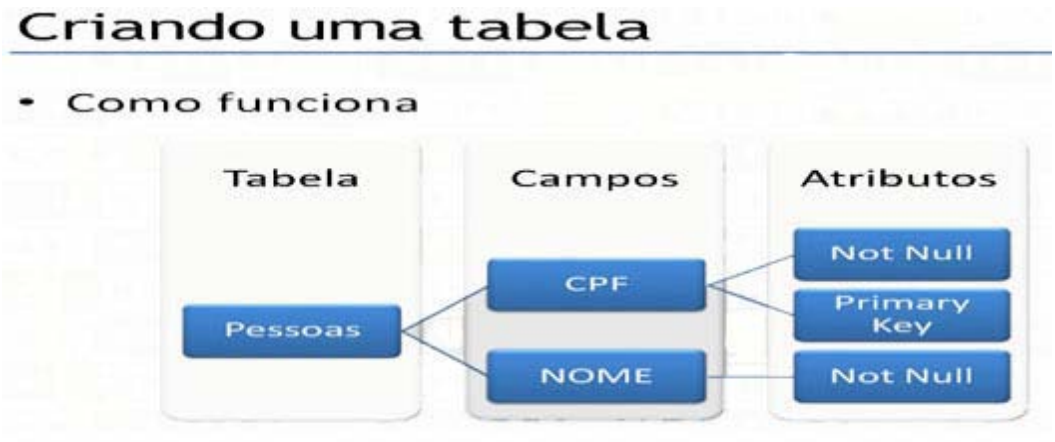
5. SERVIDOR DE BANCOS DE DADOS

Um servidor de Bancos de Dados pode armazenar e gerenciar um ou mais banco de dados, um banco de dados por sua vez, pode possuir uma ou mais tabelas.

5.1 Criando um Banco de Dados

INT - Valores numéricos inteiros variando de -2.147.483.648 até 2.147.483.647

O primeiro comando para criação de um banco de dados em SQL é: **CREATE DATABASE_Nome** (nome sem espaços e caracteres especiais). Os bancos de dados podem utilizar conjuntos de caracteres específicos dependendo da linguagem utilizada sendo que, os mais utilizados são: **Latin1 e UTF-8** (codificação de caracteres específicos: acentos, ç, etc.). Abaixo, exemplo de criação de tabela com padrão UTF-8




```
CREATE TABLE CURSO_SQL (
ID INT UNSIGNED NOT NULL AUTO_INCREMENT, NOME VARCHAR(45) NOT NULL,
ENDERECO VARCHAR(45) NOT NULL,
EMAIL VARCHAR(45) NOT NULL,
PRIMARY KEY (id) ) CHARACTER SET utf8 COLLATE utf8_general_ci;
```

ATRIBUTOS

- **NULL / Not NULL** – Permite ou não valores nulos
- **Unsigned / Signed** – Permite ou não números negativos
- **Auto-increment** – Gera sequencia de contadores (1, 2, 3, ...)

Os tipos de dados char e varchar armazenam dados compostos do seguinte:

- Caracteres maiúsculos e minúsculos, como a, b e c.
- Numerais, como 1, 2 e 3.
- Caracteres especiais, como arroba, (@), E comercial (&) e ponto de exclamação (!).

VARCHAR - Tipo de dado que trabalha sem completar com espaços em branco a área não utilizada da string opostamente ao tipo **VAR**

Inserindo Dados em uma tabela

Campos da Tabela

```
inser into curso_sql (id,nome,endereço,email)
value (1,'Rogerio','Rua Bela Cintra',rogerio@nrsystem.com);
```

Para **alterar o banco** de dados é utilizado o comando ALTER seguido do seu nome e propriedade a ser alterada. Sua sintaxe é: **ALTER DATABASE Nome Propriedade**

A **exclusão definitiva** (irreversível) é possibilitada pelo comando DROP, sendo sua sintaxe: **DROP DATABASE Nome**, a exclusão de tabelas que são referenciadas em outras tabelas pode gerar problemas de inconsistência nas mesmas.

Para deletar colunas de uma tabela usamos a sintaxe: **ALTER TABLE Tabela DROP Campo**

As tabelas seguem uma hierarquia, pois podem conter um ou mais campos, que por sua vez podem possuir um ou mais atributos em seus campos. O exemplo da alteração e exclusão citados no tópico anterior, o comando **ALTER TABLE Nome Propriedade** é usado para alterar os dados da tabela e **DROP TABLE Nome** para exclusão (definitiva e irreversível). Os dados são inseridos nas tabelas por ordem de chegadas, muitas vezes precisamos ordenar alguns campos como, por exemplo, nomes em ordem alfabética, número de CPF em ordem crescente, etc. **Os índices** são estruturas que armazenam de forma isolada e independente o ordenamento de uma tabela baseada no campo em que o índice foi criado para que não seja preciso ordenar os campos sempre que requisitados.

Criar Índice=**CREATE INDEX Nome ON TabelaEColuna**

Alterar Índice= **ALTER INDEX Nome Propriedade**

Exclusão= **DROP INDEX Nome**

OBS: O comando **DROP INDEX** remove o índice, mas não remove os dados no campo em questão.

Exemplo de Criação de índice baseado nos campos CPF e Nome

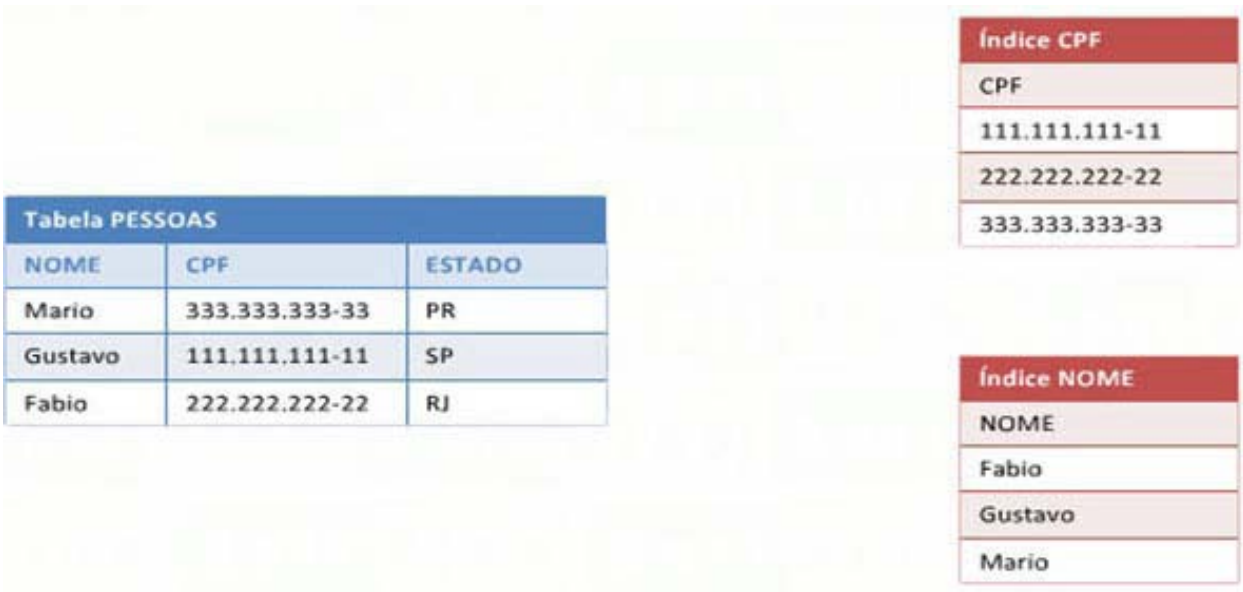


Tabela PESSOAS		
NOME	CPF	ESTADO
Mario	333.333.333-33	PR
Gustavo	111.111.111-11	SP
Fabio	222.222.222-22	RJ

Índice CPF
CPF
111.111.111-11
222.222.222-22
333.333.333-33

Índice NOME
NOME
Fabio
Gustavo
Mario

Sintaxe dos comandos:

CREATE INDEX PESSOAS ON CPF (PESSOAS);

CREATE INDEX PESSOAS ON NOME (PESSOAS);

6. MANIPULAÇÃO DE DADOS – DML E DQL

Linguagem de manipulação de Dados

INSERT – Inserir dados.

UPDATE – Alterar dados.

DELETE – Deletar dados.

Linguagem de Consulta de Dados

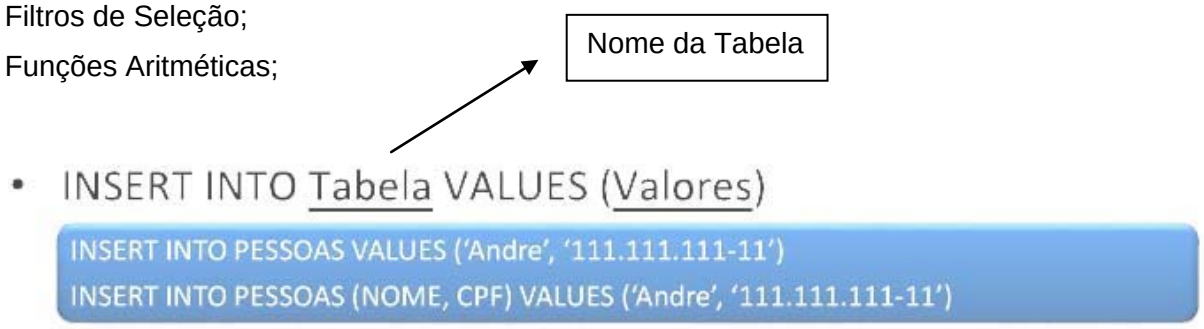
SELECT – Retorna Dados.

Ordenação de Dados;

Agrupamento de Dados;

Filtros de Seleção;

Funções Aritméticas;



Sintaxe do comando INSERT

OBS. O SQL se diferencia das outras linguagens utilizadas para bancos de dados por especificar a forma do resultado e não o "caminho" para se chegar a ele. No exemplo 2 existe um detalhamento para inserção dos dados determinando a ordem Nome e depois CPF independente da criação da tabela.

Para **buscar** ou **recuperar dados** de uma tabela utilize o comando **SELECT**

Sintaxe do comando SELECT:

Se desejar que mais de um campo seja retornado, poderá separar por vírgulas os nomes dos campos, exemplo:

SELECT nome, endereco FROM tblalunos

- SELECT Campos FROM Tabela

```
SELECT * FROM PESSOAS
SELECT NOME FROM PESSOAS
```

OBS. No primeiro comando, o asterisco indica que "todos" os campos da tabela pessoas devem ser retornados, já, no segundo comando, estamos especificando que apenas o campo nome da tabela pessoas deve ser retornado.

Para **atualização** de dados utilizamos o comando **UPDATE** através da sintaxe:

- UPDATE Tabela SET Campo = Valor

```
UPDATE PESSOAS SET NOME = 'Andre Milani'
```

Para **deletar** uma tabela utilize o comando **DELETE**

- DELETE FROM Tabela

```
DELETE FROM PESSOAS
```

7. FILTROS DE SELEÇÃO

Os comandos abordados até então, são utilizados em relação à estrutura das tabelas em sua totalidade. Para busca e seleção de áreas específicas das tabelas devemos acrescentar o complemento **WERE** em alguns dos comandos abordados anteriormente para que a operação somente seja realizada nos registros que atenderem as condições especificadas, neste sentido, faz-se necessário a abordagem do conceito de operadores.

OPERADORES RELACIONAIS:

- Igual (=), Diferente (!=)
- Maior (>), Maior ou igual (>=)
- Menor (<), Menor ou igual (<=)
- Nulo (IS NULL), ou não-nulo (IS NOT NULL)
- Entre intervalo (BETWEEN)
- Valor parcial (like)

OPERADORES LÓGICOS:

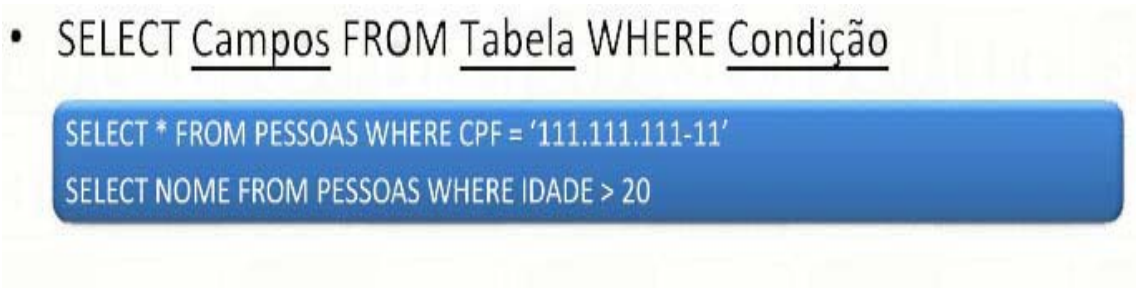
- AND
- OR
- NOT

Para exemplo de aplicação, adotemos duas situações distintas:

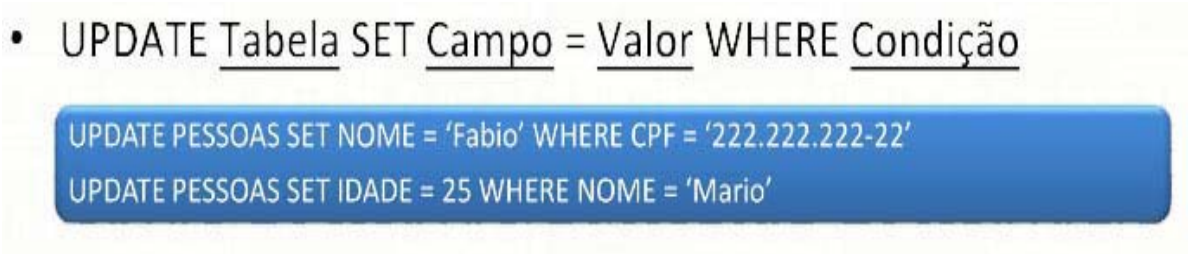
1º Buscar no banco de dados todas as informações referentes a uma determinada pessoa através do nº de CPF;

2º Pesquisar quantas pessoas com idade superior a vinte anos estão cadastradas no sistema

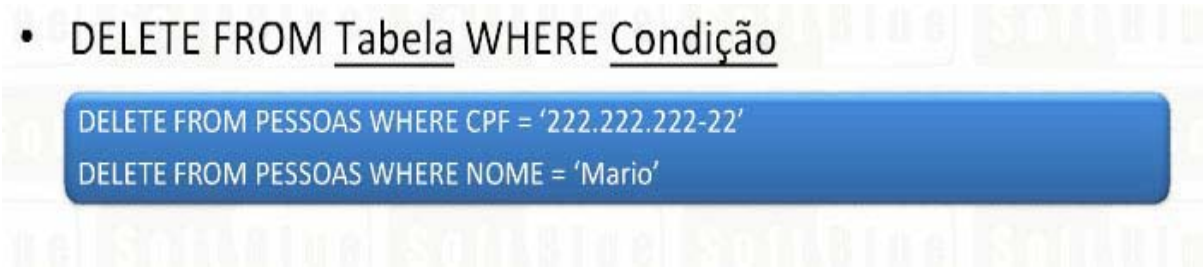
SINTAXE:



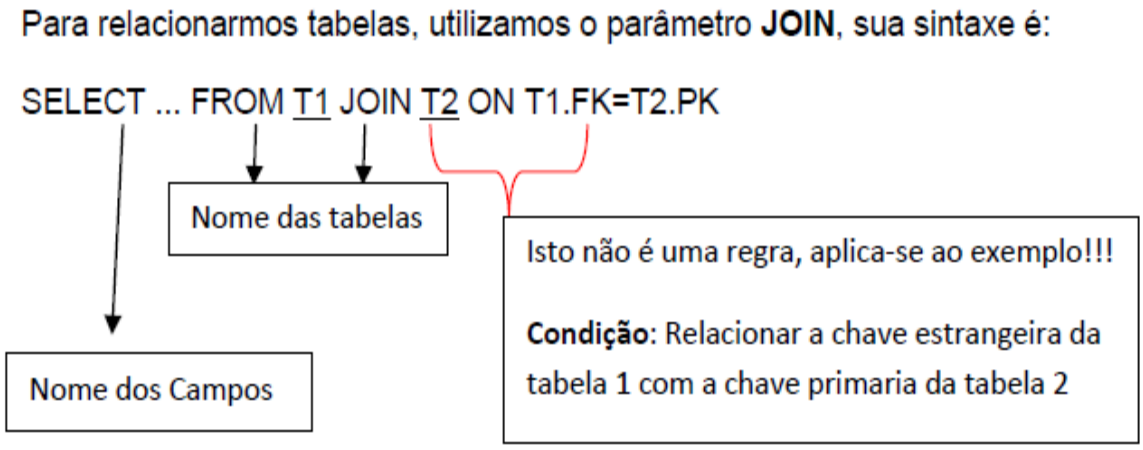
Para atualização de campos específicos, podemos utilizar o comando **UPDATE** seguido do complemento **WERE**, exemplos a seguir:
Se fosse necessário **atualizar** o nome do titular do CPF 222.222.222-22 para Fábio, usaríamos a sintaxe abaixo, o mesmo se aplica a mudança de idade para 25 anos do usuário Mário no segundo exemplo



Para **deletar** campos específicos sem excluir a tabela inteira, podemos usar o comando **DELETE** precedido do complemento **WERE**.

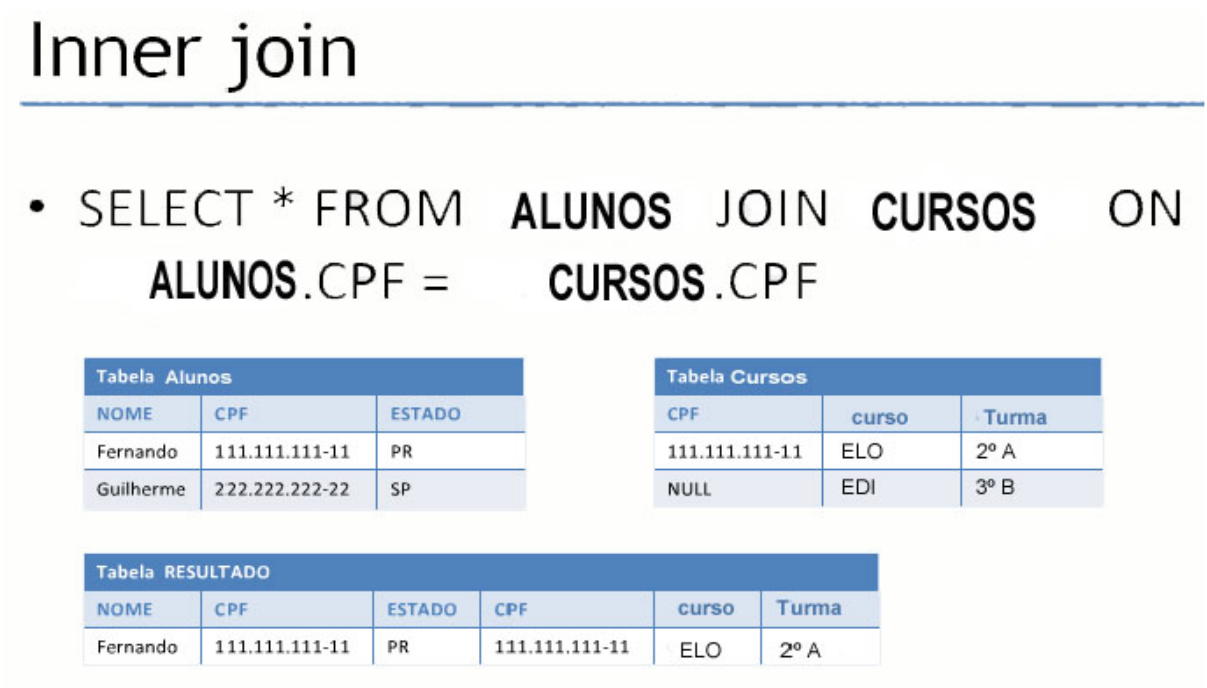


8. RELACIONAMENTOS NO SQL



Se os campos em comum entre as tabelas possuírem o mesmo nome, podemos usar esta sintaxe:

SELECT ... FROM T1 JOIN T2 USING Chave → Nome da Chave



Referência Bibliográfica

COSTA, R.F. Fundamentos de Banco de Dados. Disponível em: http://nrsystem.com.br/Fund_Banco_Dados.pdf

SILBERSCHATZ, A; KORT, F.K. **SISTEMAS DE BANCO DE DADOS.** PLT, Anhanguera Educacional, Programa do Livro-Texto

Professor: Rogério Fernandes
rogerio@nrsystem.com