7 - BANCO DE DADOS I

7.1 - MÓDULO 01 - QUAL SERIA A MELHOR FORMA DE CARLOS ORGANIZAR ESSA BIBLIOTECA?

7.1.1 - Compreender conceitos básicos de Banco de dados (BD)

7.1.1.1 - Banco de Dados

Os bancos de dados podem ser dados de cadastros, históricos de compras, de filmes vistos, de músicas tocadas etc. Dificilmente, há uma aplicação que não precise guardar algum tipo de informação.

Assim, o `database`, ou `banco de dados`, **é onde conseguimos guardar informações que fazem o nosso sistema funcionar.**

Um `banco de dados` **é definido como uma coleção de dados persistentes**, pois ele *continua armazenando os dados mesmo se a máquina em uso for desligada*.

> Diferente da Memória RAM, que perde os dados quando a máquina é encerrada.

1960, pela IBM, foi o primeiro banco de dados. Visava otimizar o armazenamento de dados que era feito através do acesso sequencial e do arquivo de acesso direto

Qualuer coisa que colete informações é um banco de dados.

Heuser (2009) define "dado" como sendo um **fato registrado do mundo real, que possui um significado implícito no contexto de domínio de aplicação.**

Um banco de dados é um **conjunto de dados inter-relacionados** (SILBERSCHATZ, 2006), podendo ser classificado como relacional ou não relacional

7.1.1.2 - Banco de Dados Relacional

Esse tipo de banco de dados possui os dados armazenados em tabelas **separadas**, sendo possível relacioná-las. Nos bancos relacionais, as tabelas e os dados *ocupam menos espaço e são mais eficientes*.

O modelo mais comum, o modelo relacional, classifica dados em tabelas, também **conhecidas como relações**, cada uma das quais consiste em colunas e linhas. *Cada coluna lista um atributo* da entidade em questão, como preço, código postal ou data de nascimento. **Juntos, os atributos em uma relação são chamados de domínio**. Um determinado atributo ou combinação de atributos é escolhido como uma `chave primária` que pode ser consultada em outras tabelas, **quando é chamada de chave estrangeira.**

Cada linha, também chamada de `tupla`, *inclui dados sobre uma instância específica da entidade em questão*, como um determinado colaborador.

O modelo também explica os tipos de relações entre essas tabelas, incluindo relações `uma para uma`, `uma para muitas` e `muitas para muitas`.

Student ID	First name	Last name
52-743965	Charles	Peters
48-209689	Anthony	Sondrup
14-204968	Rebecca	Phillips

ProviderID	Provider name
156-983	UnitedHealth
146-823	Blue Shield
447-784	Carefirst Inc.

Student ID	ProviderID	Type of plan	Start date
52-743965	156-983	HSA	04/01/2016
48-209689	146-823	HMO	12/01/2015
14-204968	447-784	HSA	03/14/2016

7.1.1.3 - Abstração de Dados

A abstração de dados garante uma visão intangível de um banco de dados para seu usuário. Não importa para o usuário saber a maneira como esses dados estão armazenados, **o importante é tê-los disponíveis quando solicitados**.

Exitem 3 níveis de abstração de dados, de acordo com Silberschatz(2006)

1. **Físico**, é a abstração mais baixa, mostra **COMO SÃO ARMAZENADOS**.
2. **Lógico**, é de abstração intermediária, **mostra os DADOS ARMAZENADOS** no banco de dados **E AS RELAÇÕES existentes** entre eles.

3. **Vizualização**, é o nível mais alto, apresenta a parte do banco de dados de **maior interesse para o usuário final.**

7.1.1.4 - Modelos de dados

Silberschatz (2006) diz que um **modelo de dados é uma coleção de ferramentas conceituais para descrever dados, relações de dados, semântica de dados e restrições de consistência**. Ou seja, um modelo de dados descreve a estrutura lógica e física de um banco de dados.

Para Elmasri (2008), existem *dois níveis* de modelos de dados:

1. **alto nível**, modelos de dados conceituais que *descrevem os dados como os usuários os percebem*;
2. **baixo nível**, modelos de dados físicos, *descrevem os detalhes de como os dados estão armazenados* no computador.

7.1.1.5 - Sistema de Gerenciamento de Banco de Dados (SGBD)

É pelo SGBD que o usuário cria e manipula o BD.

> O SGBD é um conjunto de softwares formado por um ou vários bancos de dados e possui aplicações capazes de manipular esses bancos.

Ele possui um catálogo de dados que tem uma descrição completa de como os dados estão armazenados. Essas informações recebem o nome de metadados.

O SGBD ainda *proporciona a separação* entre **programas e dados**. Assim, qualquer modificação feita na estrutura não vai alterar o código fonte como era feito no processamento tradicional de arquivos, a estrutura é modificada **só no catálogo**, sem alterar o programa.

.	VANTAGENS	DESVANTAGENS
SGBD	Controle de redundância, Compartilhamento de dados, restrição ao acesso não autorizado, padronização, redução do tempo de desenvolvimento de aplicações	**alto custo** de implementação, já que hardwares são necessários. Além disso, mão de obra, uma vez que demanda gerenciamento e manutanção.

> Leitura Complementar

[ALVES, Gustavo. O que é um banco de dados?. Dicas de programação, [s/d].]
https://dicasdeprogramacao.com.br/o-que-e-um-banco-de-dados/

> Referência Bibliográfica

ELMASRI, Ramez E.; NAVATHE, Shamkant B. Sistemas de Banco de Dados. 4 ed. Boston: Pearson Addison-Wesley, 2008.

HEUSER, Carlos Alberto. Projeto de banco de dados. Vol. 4. Porto Alegre: Bookman Editora, 2009.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. Sistema de Banco de Dados. 5 ed. Rio de Janeiro: Campus, 2006.

7.1.1.6 - Anotação Exercícios

1. Sobre a evolução dos mecanismos de armazenamento, quais foram os primeiros modelos de bancos de dados desenvolvidos?
    1. O modelo hierárquico e o modelo em redes.
        1. Resposta correta!O modelo hierárquico foi desenvolvido pelo Information Management System (IMS) e o modelo em redes pelo Comittee for Data Systems Language (CODASYL).
2. Assinale a alternativa que apresenta o nome do modelo de banco de dados introduzido por E.F. Codd em 1970, considerado padrão mundial e que possui seus dados armazenados em tabelas.
    1. Modelo relacional.
        1. Resposta correta!O modelo relacional é um padrão utilizado no mundo inteiro, armazena seus dados em tabelas e proporciona o relacionamento entre elas.
3. Os bancos de dados são coleções de registros armazenados em algum dispositivo. Eles possuem modelos, como o hierárquico, o em redes, o relacional e o Orientado a Objetos. Sobre esses modelos, assinale a resposta correta.
    1. O modelo relacional representa os dados como uma coleção de linhas e colunas organizadas dentro de uma tabela.
        1. Resposta correta!No banco de dados relacional, as tabelas e os registros são formados através de colunas e linhas.
4. O Sistema de Gerenciamento de Banco de Dados (SGBD) é um conjunto de programas que serve para gerir e manipular os bancos de dados. Identifique a alternativa que apresenta apenas SGBDs.
    1. Oracle, MySQL e SQL Server.
        1. Resposta correta!Resposta correta!
5. Na época do surgimento dos computadores, como os dados passaram a ser armazenados?
    1. Em armários, através de fichas de papel preenchidas.
        1. Resposta incorreta!Esse modo de armazenamento ocorria quando os computadores ainda não existiam.
    2. Em sistemas de armazenamentos sequenciais.
        1. Resposta correta!Isso mesmo! Nesse tipo de sistema, se houvessem cinquenta arquivos guardados, seria preciso passar por todos os quarenta e nove anteriores para acessar o 50º arquivo.

7.1.2 - Modelar banco dados relacionais: Conceito relacional, lógico e físico

7.1.2.1 - Vídeo - Etapas da modelagem de banco de dados

Modelagem pode ser *dividida em duas etapas*.

1. Análise de Requisitos;
    1. Possível identificar os dados a serem armazenados no banco;
    2. Descarte dos dados não necessários;
2. Modelagem propriamente dita;
    1. Feita a partir do modelo entidade-relacionamento-`MER`;
        1. Utilizar o `MER` é importante para separar as informações necessárias na construção do banco, identificando diferentes tipos de dados e como eles se relacionam;
        2. Por meio do MER podemos construir o Diagrama entidade-relacionamento-`DER`

    1. `DER`, apresenta **graficamente** a estrutura do banco de dados e sua relação com os elementos.
3. `MER` também é composto por `Atributos`, assim como `Entidades` e `Relacionamentos`. Tem sua representação construída por **formas geométricas**
    1. RETÁNGULO - Entidades;
        1. identificar de acordo com a **existência no mundo real**.
            1. LÓGICA;
                1. Ex: Dicíplica é lógica, pois não ocupa lugar no espaço.
            2. FÍSICA;
                1. Ex: aluno e prof. são entidades físicas, porque são tangíveis e visíveis no mundo real.
        2. Os objetos das entidades ainda podem ser:
            1. FORTES;
                1. Sua existência **independe** de outra entidade;
            2. FRACAS;
                1. Sua existência **depende** de outra pra existir;
            3. ASSOSSIATIVAS;
                1. Surgem quando existe a necessidade de associar uma entidade a um relacionamento.
    2. CÍRCULOS - Atributos;
        1. Servem para caracterizar e escrever a entidade dentro de um **domínio**, e possuem algumas **funções**:
            1. Descritivas;
                1. Nome ou cor;
            2. Nominativas;
                1. Nome, código ou número;
            3. Referênciais;
                1. CPF do cliente na venda;
        2. **Estrutura** dos atributos podem ser:
            1. Simples;
                1. Apenas 1 atributo define a caracteristica da entidade;
            2. Composto;
                1. São utilizados vários atributos para definir as informações da entidade.
    3. LOZANGULOS - Relacionamentos;
        1. identificar os relacionamentos entre elas:
            1. `Um pra Um`;
                1. As duas entidades envolvidas tem referência OBRIGATÓRIA a apenas uma unidade da outra.

2. `Um pra Muitos`
    1. Uma entidade pode fazer referência a **VÁRIAS unidades** da outra, enquanto essas só podem fazer referência a UMA.

    Observamos que Estado pode fazer referência a várias cidades, e cidades só podem fazer referência a um Estado.
3. `Muitos pra Muitos`
    1. Uma entidade pode fazer referência a várias Unidades da outra e vice e versa;

    Uma entidade funcionário pode participar mais de um projeto, assim como um projeto pode ter mais de um funcionários envolvido.

Bancos de dados relacionais são fundamentais na programação e também os mais utilizados.

7.1.2.2 - Modelagem de BDs relacionais - Fases

> Fase inicial - Análise de requisitos

Nela, é **realizada a identificação dos requisitos que constam em um banco de dados**. Entrevistas são feitas e o minimundo é desenvolvido com a descrição textual geral do projeto.

As regras de negócios desse documento devem ser identificadas para que as informações sejam bem executadas, pois informações erradas afetam todo o projeto.

> Projeto conceitual

O projeto conceitual *tem base nas informações coletadas pela análise de requisitos*. Assim, um esquema é gerado, utilizando um modelo de dados conceitual de **alto nível**, o Modelo Entidade-Relacionamento (**MER**). Esse modelo é simplificado para aproximar a composição dos bancos de dados aos usuários, *facilitando a compreensão das informações*. Por isso, a representação dos dados armazenados não é tão relevante.

O MER é representado através de diagramas. As `entidades` são **retângulos**, os `atributos` são **elipses** e os `relacionamentos` são **losangos**.

Nesta etapa, os três tipos de relacionamentos de entidades são fundamentais, sendo delas:

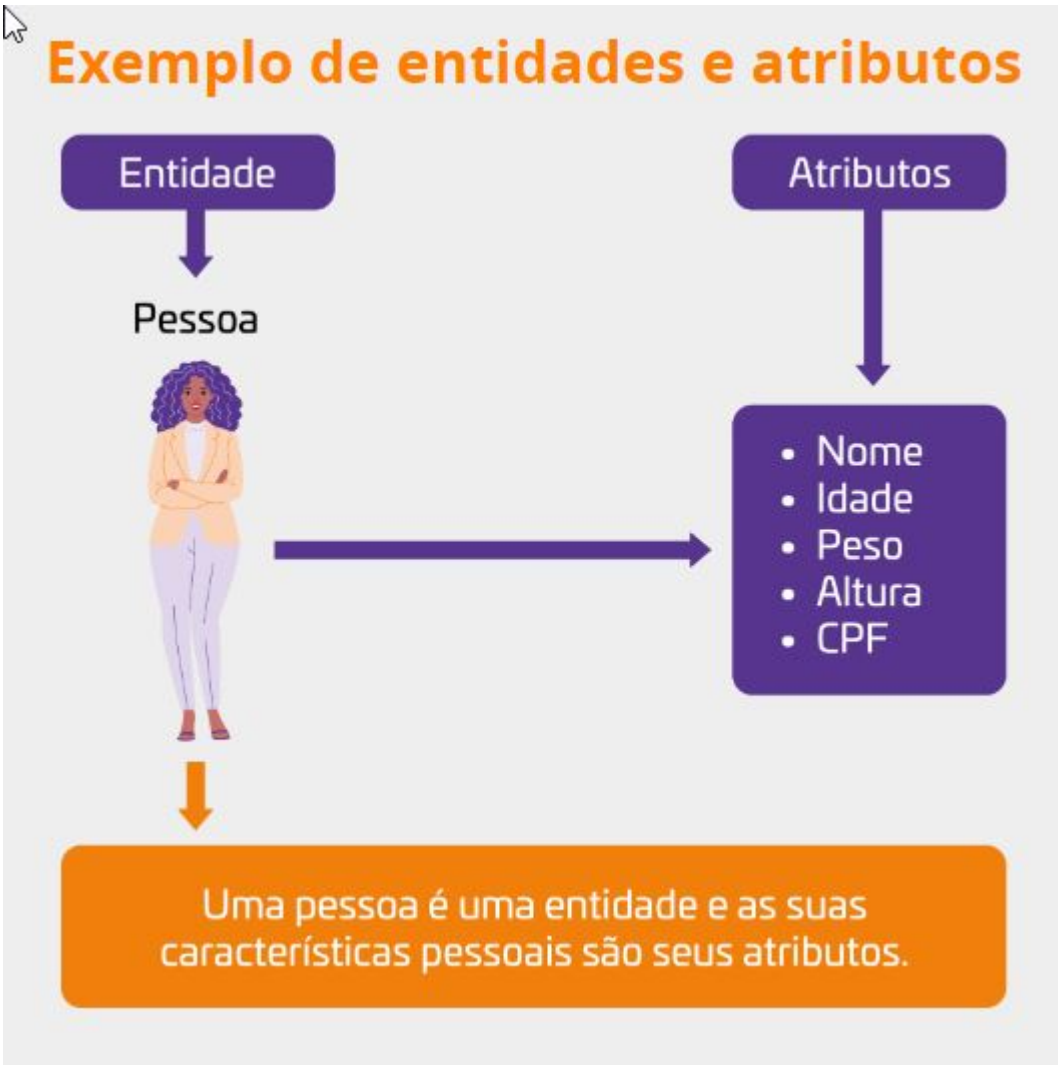
- um para um (1, 1);
- um para muitos (1, n) ou muitos para um ou (n, 1);
- muitos para muitos (n, n);

> Entidade e atributos

A `entidade` pode ser definida como **um objeto do mundo real, concreto ou abstrato, com existência independente**.

Os `atributos` c**ompõem um conjunto particular de propriedades atreladas à entidade.**

> ```
> ATENÇÃO:
> ```
>
> Atributos são características das entidades.



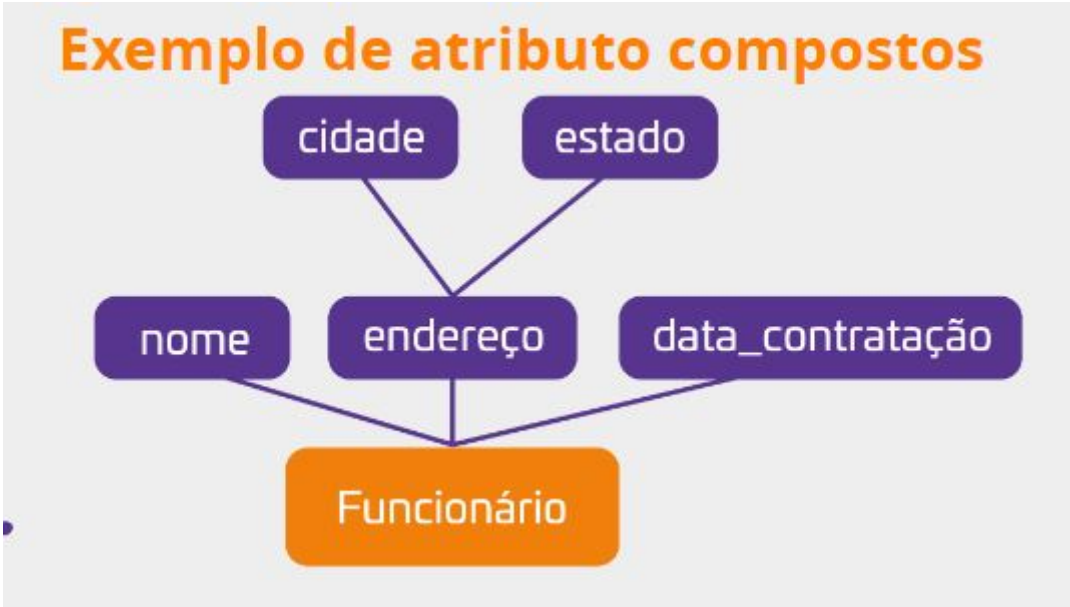
> Atributos

Eles podem ser classificados como:

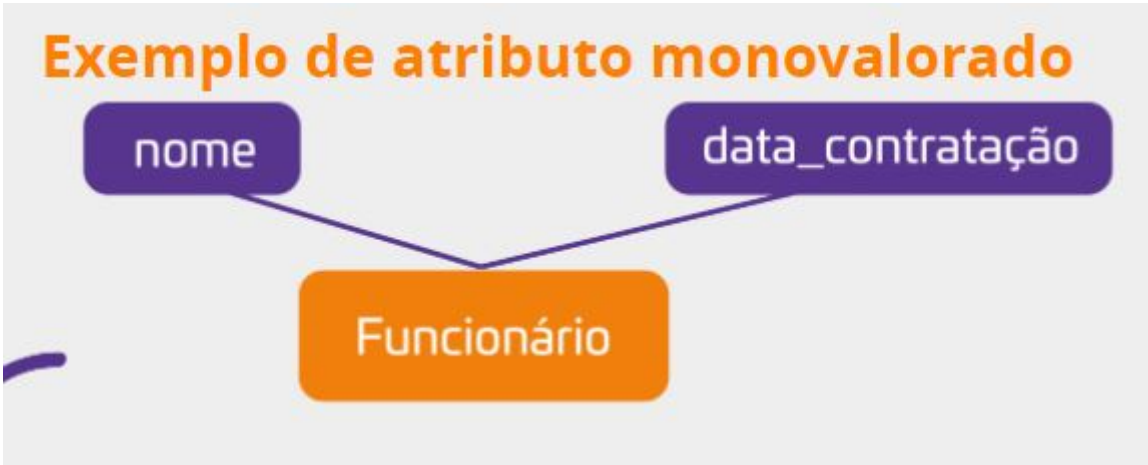
- Atributo **simples** - não pode ser subdividido;

- Ex: CPF, um funcionário tem esse atributo simples, pois esse atributo não pode ser dividido;

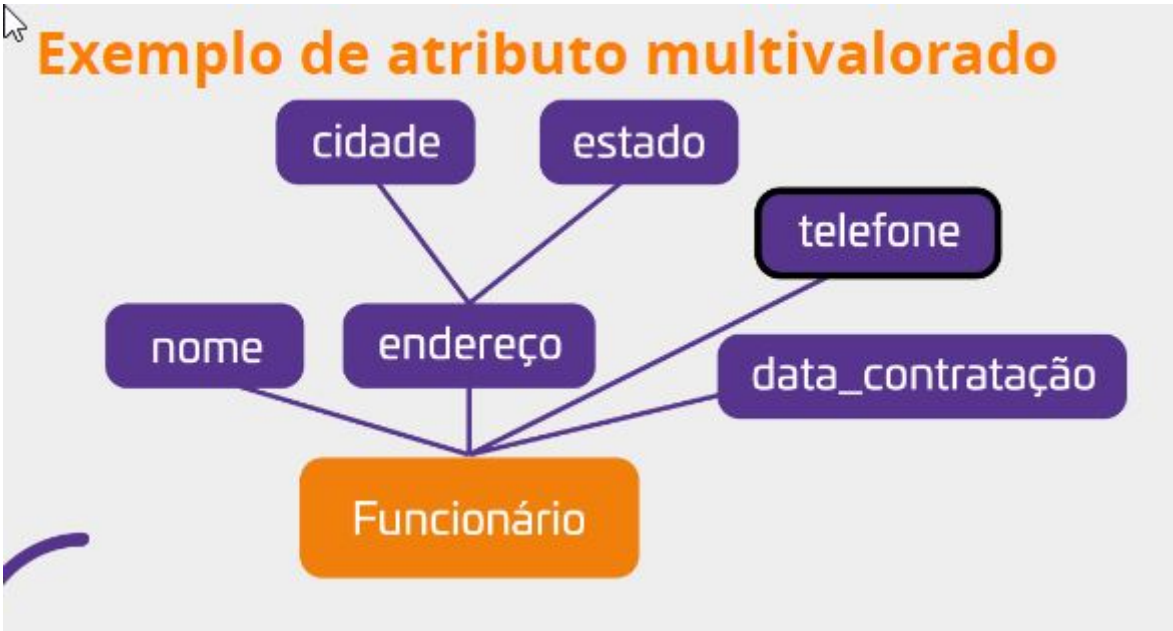
- Atributo **composto** - pode ser dividido em diversas partes, com significados independentes entre si;
  - Ex: para um funcionário, ele tem um atributo composto que é o endereço, que possuem subprodutos como cidade, estado, endereço, data da contratação;



- Atributo **monovalorado** - ou simplesmente valorado, só assume um valor em uma determinada instância;
  - Ex: Na entidade `funcionário`, temos dois atributos, nome e data_contratação, que são atributos monovalorados, ou seja, cada um com um único valor.

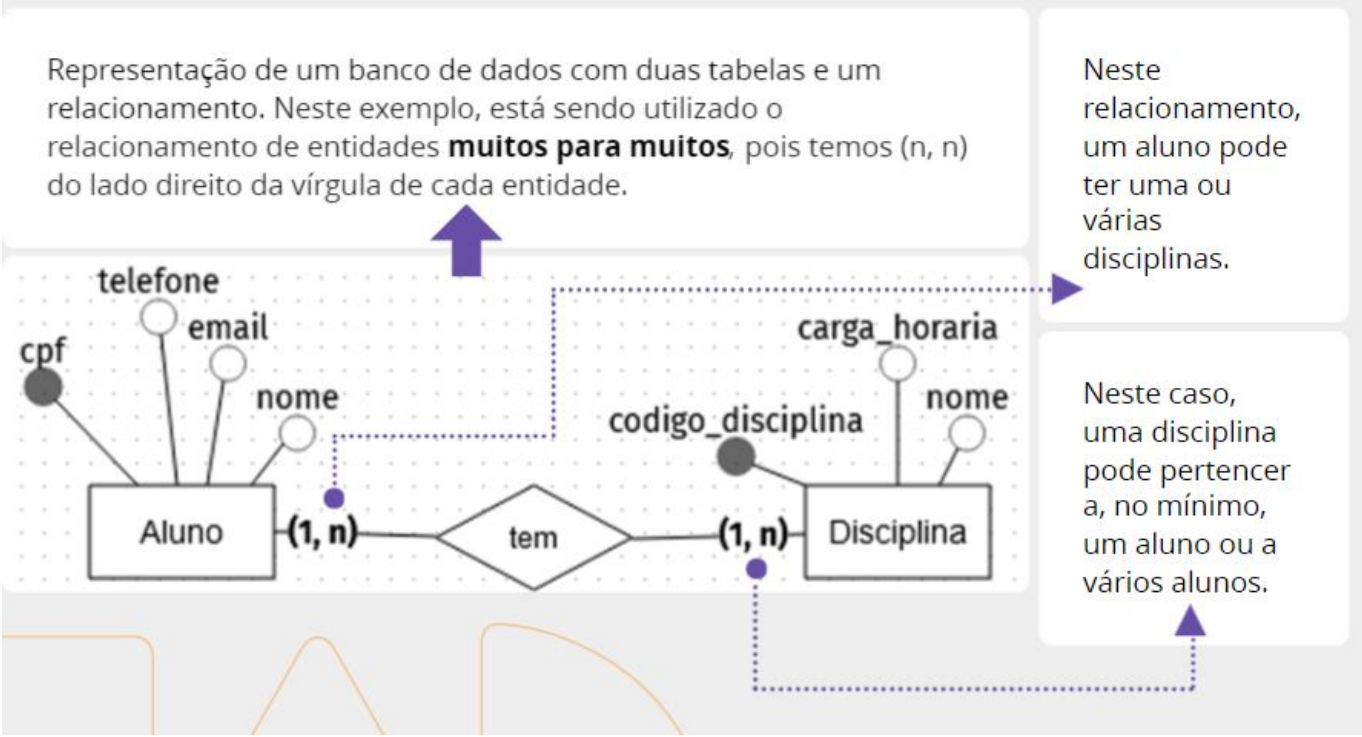


- Atributo **multivalorado** - pode assumir diversos valores em uma mesma instância;
  - Entidade `funcionário` tem um abriuto `telefone`, ele é multivalorado, pois pode ter mais de um número de telefone cadastrado;



- Atributo **derivado** - é gerado a partir de outro atributo.
  - Ex: entidade `funcionário`, o atributo `tempo_de_casa` precisa do atributo `data_contratação` para calcular seu valor.

> Exemplo Projeto Conceitual (n, n) "muitos para muitos"

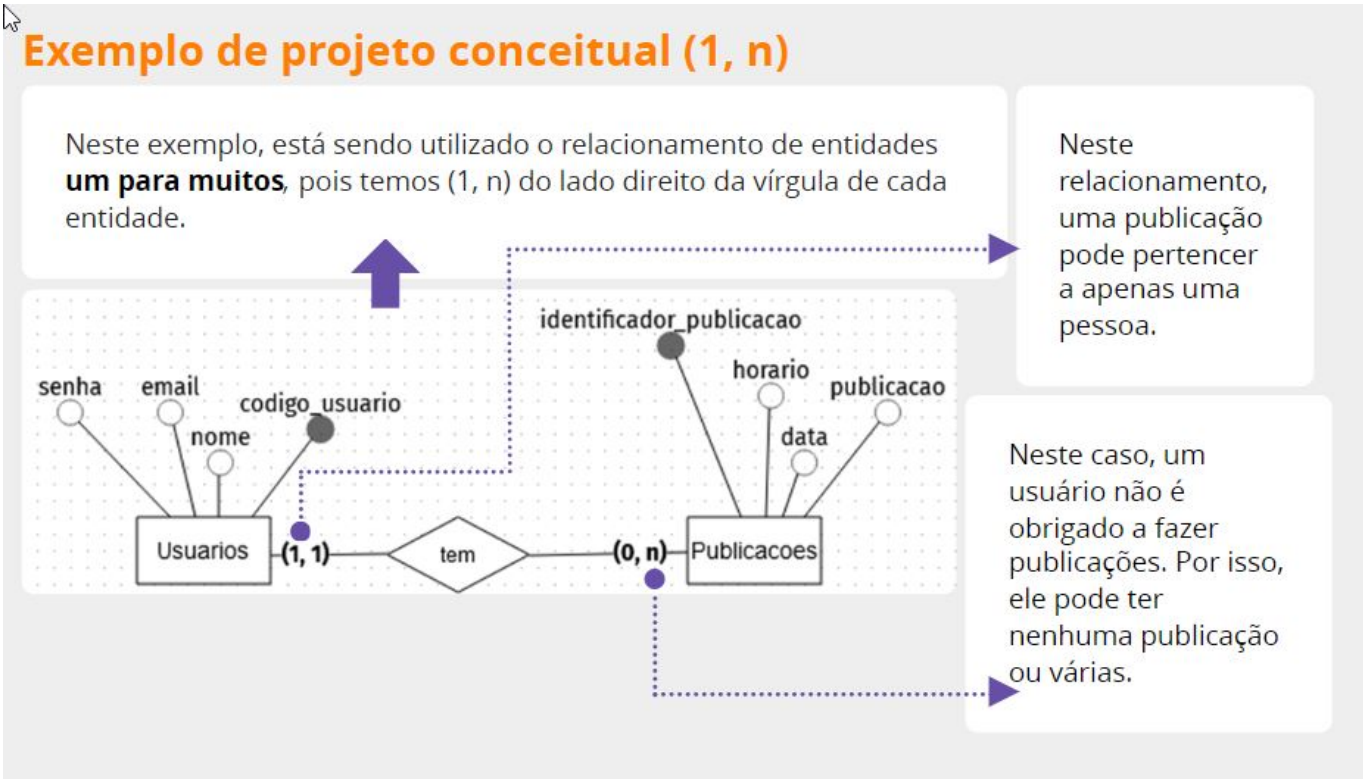


Neste exemplo, está sendo utilizado o relacionamento de entidades muitos para muitos, **pois temos (n, n) do lado direito da vírgula de cada entidade.**

Toda tabela possui uma chave primária, que se trata de um *identificador único*e **não pode ter mais de um valor**.

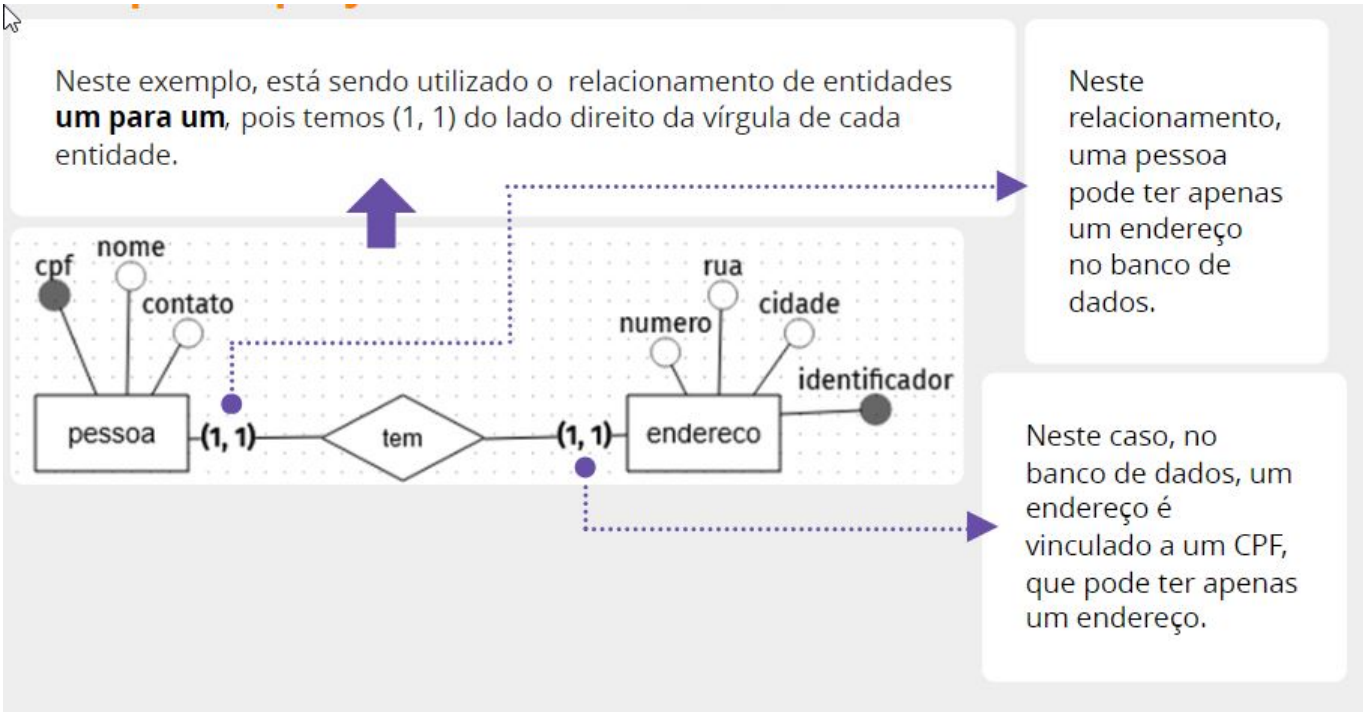
Neste caso, as chaves primárias são os atributos referenciais, que são o CPF do aluno e o código da disciplina. Isso porque cada aluno tem seu próprio CPF e cada disciplina tem um código para representá-la.

> Exemplo Projeto Conceitual (1, n) - "um para muitos"



Neste exemplo, está sendo utilizado o relacionamento de entidades um para muitos,**pois temos (1, n) do lado direito da vírgula de cada entidade**.

> Exemplo Projeto Conceitual (1, 1) - "um pra um"



Neste exemplo, está sendo utilizado o relacionamento de entidades um para um, **pois temos (1, 1) do lado direito da vírgula de cada entidade**.

> Projeto Lógico

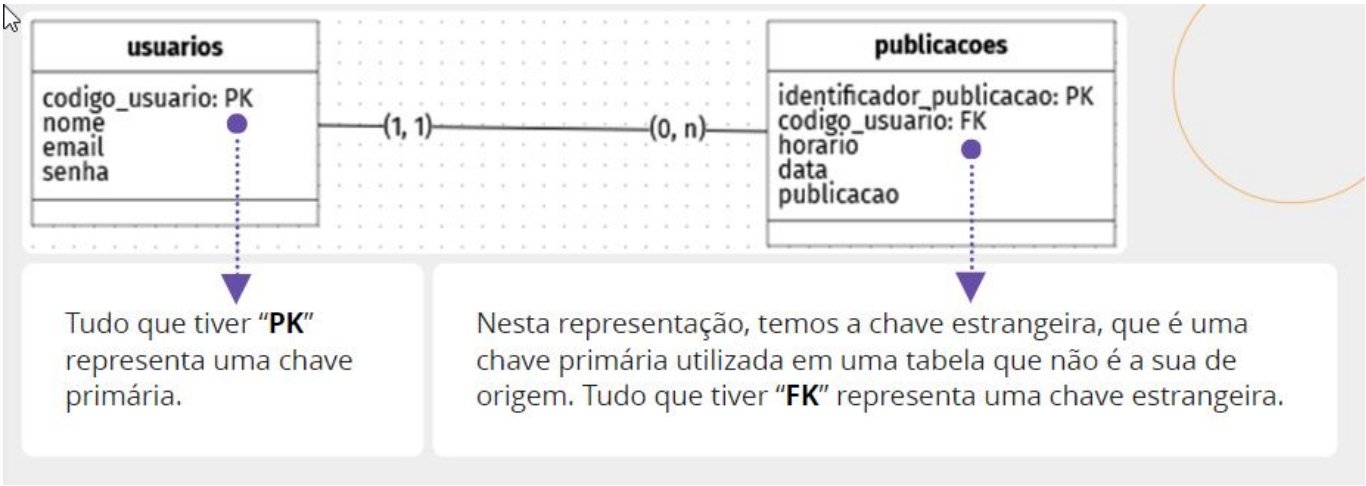
O projeto lógico **é uma etapa do processo do MER**. Nele, o *mapeamento do projeto conceitual é elaborado*.

A construção dos modelos internos é realizada detalhando tabelas, regras, relacionamentos, dados das colunas (tamanho e tipo), entre outros.

O projeto lógico **tem como resultado o esquema detalhado do banco de dados**.

```
Exemplo de Projeto Lógico
```

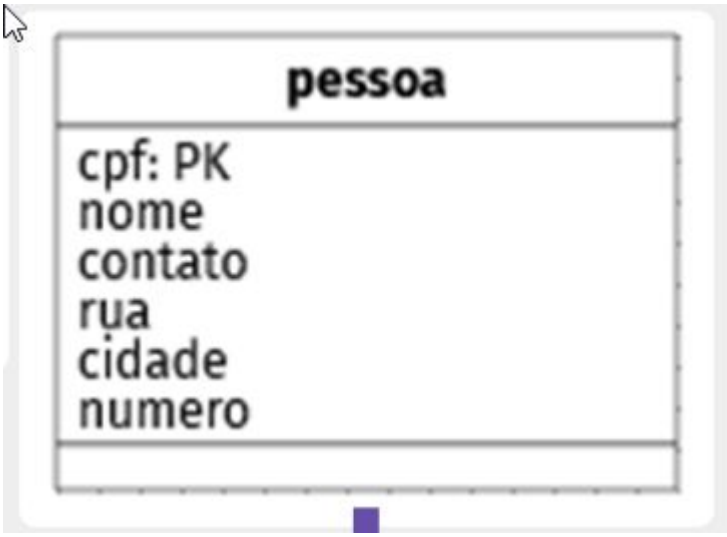
Aqui, temos a conversão do modelo conceitual um para muitos (1, n) para o modelo lógico. Neste modelo, cada entidade se torna uma tabela e seus atributos, os dados de cada tabela.



Aqui, temos a conversão do modelo conceitual muitos para muitos **(n, n)** para o modelo lógico. Neste modelo, cada entidade se torna uma tabela, seus atributos se tornam os dados de cada tabela e o relacionamento se torna uma nova tabela.

Uma nova tabela é criada para que ocorra o relacionamento entre as outras tabelas. Por isso, **ela possui apenas as chaves estrangeiras.**

Aqui, temos a conversão do modelo conceitual um para um **(1,1)** para o modelo lógico. Neste modelo, todas as entidades se tornam uma única tabela, ou seja, ocorre uma união dos dados.



Não existe chave estrangeira nesse tipo de tabela. Por isso, ela é considerada a mais simples.

> Projeto Físico

O projeto físico **está no nível mais baixo em relação ao usuário final.** Nessa etapa, são definidas:

- as estruturas de armazenamento,
- os índices e a organização de arquivos do banco de dados.

Ela é considerada a **parte final do projeto**,*pois apresenta a forma de armazenamento, as permissões de acesso ao usuário e os scripts para criação dos objetos, tabelas, colunas, visões, funções e entre outros*

A seguir veremos a conversão do modelo lógico um para muitos (1, n) para o modelo físico. Neste modelo, os dados são representados em linhas de código SQL, que são utilizadas para a criação das tabelas em um banco de dados.

```
CREATE TABLE usuarios
(
 codigo_usuario INT PRIMARY KEY,
 nome VARCHAR(n) DEFAULT '30',
 email VARCHAR(n) DEFAULT '30',
 senha VARCHAR(n) DEFAULT '30',
 CHECK (codigo_usuario = undefined)
);

CREATE TABLE publicacoes
(
 identificador_publicacao INT PRIMARY KEY,
 codigo_usuario INT,
 horario VARCHAR(n) DEFAULT '5',
 data DATE,
 publicacao VARCHAR(n) DEFAULT '1000',
);

ALTER TABLE publicacoes ADD FOREIGN KEY(codigo_usuario) REFERENCES usuarios
(codigo_usuario)
```

Cada tabela é criada através do comando *create table* e os seus atributos são descritos juntamente com a descrição do seu respectivo tipo. Neste exemplo, temos dados do tipo *varchar*, que guardam textos. Também há o número de caracteres que esse dado pode guardar, neste caso, *varchar default 30*.

Exemplo de projeto físico

```
CREATE TABLE usuarios
(
 codigo_usuario INT PRIMARY KEY,
 nome VARCHAR(n) DEFAULT '30',
 email VARCHAR(n) DEFAULT '30',
 senha VARCHAR(n) DEFAULT '30',
 CHECK (codigo_usuario = undefined)
);

CREATE TABLE publicacoes
(
 identificador_publicacao INT PRIMARY KEY,
 codigo_usuario INT,
 horario VARCHAR(n) DEFAULT '5',
 data DATE,
 publicacao VARCHAR(n) DEFAULT '1000',
);

ALTER TABLE publicacoes ADD FOREIGN KEY(codigo_usuario) REFERENCES usuarios
(codigo_usuario)
```

Quando temos uma chave estrangeira, é necessário descrever qual é a sua origem.

> Bancos de Dados

Existem ferramentas on-line destinadas à criação de cada um desses modelos vistos. Uma dessas é o brModelo, que é simples e de fácil uso.

> Leitura Complementar

AULA 02 - Modelo de Entidade-Relacionamento (MER). Metrópole Digital, [s/d]. Disponível em: https://materialpublic.imd.ufrn.br/curso/disciplina/3/45/2/7

> Referências Bibliográficas

ELMASRI, Ramez E.; NAVATHE, Shamkant B. Sistemas de Banco de Dados. 4 ed. São Paulo: Pearson Addison-Wesley, 2008.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. Sistema de Banco de Dados. 5 ed. Rio de Janeiro: Campus, 2006.

> ATENÇÃO, A CARDINALIDADE MÍNIMA(A ESQUERDA) DA NOTAÇÃO `MIN MAX` SÓ ACEITA `ZERO` OU `UM`
> Por exemplo: (0, 1) ou (1, n)
> **zero**: NÃO É OBRIGATÓRIO
> **1** : É OBRIGATÓRIO

7.1.2.2 - Anotação exercício

1. Marque a alternativa correta sobre o conceito de Modelagem Entidade-Relacionamento.
    1. O Modelo Entidade-Relacionamento é um modelo de dados conceitual de alto nível.
        1. Resposta correta!Com esse tipo de modelagem, o usuário não precisa da representação dos dados e de como eles serão armazenados.
2. Sobre relacionamento um para um, marque a alternativa correta.
    1. Para cada registro da primeira tabela existe um correspondente na segunda tabela, e vice-versa.
        1. Resposta correta!No relacionamento um a um, os registros sempre terão o seu correspondente na outra tabela.
3. Qual é a importância da análise de requisitos no início de um projeto?
    1. A análise de requisitos identifica quais dados devem ser arquivados no banco de dados, evitando o armazenamento de informações redundantes.
        1. Resposta correta!Essa análise coleta dados indispensáveis para que o usuário possa solucionar problemas e alcançar objetivos.
4. Antes da implementação do banco de dados, algumas etapas de modelagem de dados são necessárias. Elas são importantes para a identificação de erros e, por isso, devem ser seguidas à risca. Marque a alternativa que identifica essas etapas.
    1. Modelagem Conceitual, Modelagem Lógica e Modelagem Física.
        1. Resposta correta!Um bom desenvolvedor não deve ignorar essas etapas.
5. Sobre os objetos do modelo entidade-relacionamento, marque a resposta correta.
    1. Uma entidade pode ser definida como um objeto do mundo real, um objeto concreto ou um abstrato, possuindo existência independente.
        1. Resposta correta!Entidade é o objeto básico tratado pelo Modelo ER.

7.1.3 - Normalizar um esquema de banco de dados

7.1.3.1 - Normalização de dados e aplicação das formas normais

> CONTEÚDO VÍDEO INTRODUTÓRIO

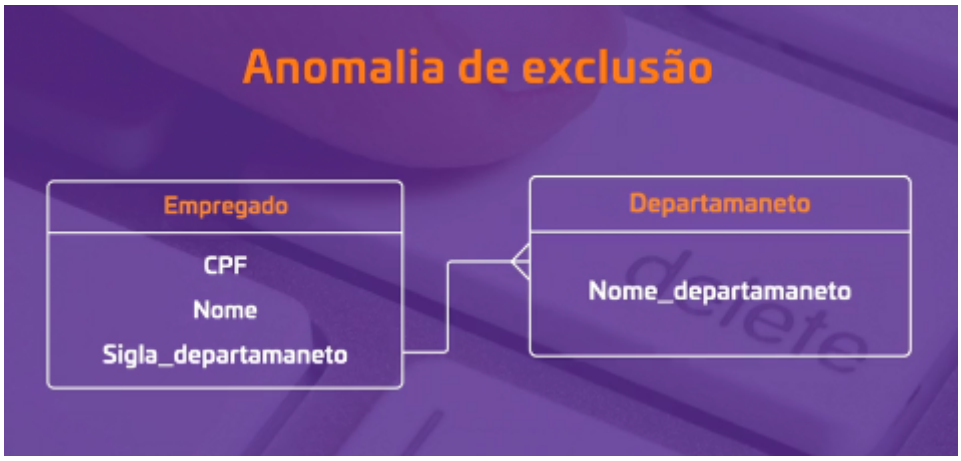
Normalização de dados é um processo que serve para evitar o surgimento de anomalias nas tabelas, surgem em três situações:

- Na `Inserção` dos dados;

- Quando adicionamos um registro na tabela sem que outro registro já esteja disponível;

Não poderia ser informado em qual departamento tal funcionário está alocado se não houver a `entidade Departamento`. Sem ela, haveria uma anomalia de inserção.

- Na `Exclusão` dos dados;
  - Acontece ao excluirmos um registro de uma tabela que acaba por afetar diretamente os registros de outra tabela;



`Entidade Empregado` depende da `Entidade Departamento`, se a `Sigla_departamento` for excluída da `Entidade Departamento` deverá ocorrer a exclusão do atributo `Sigla_departamento` de todos os registros da `Entidade Empregado`, para não incorrer em uma anomalia de exclusão.

- Na `Alteração` dos dados;
- Ocorre na modificação de um registro, fazendo com que registros em outras tabelas também sejam modificados.

Se a

`Sigla_departamento` for alterada na entidade/Tabela `Departamento` ela também deve ser alterada na entidade/Tabela `Empregado`. Ou ocorrerá anomalia.

VANTAGENS DA NORMALIZAÇÃO

- Previnir anomalias;
- Facilitar a manutenção;
- Aumentar a performace;
- Manter a integridade dos dados;

A normalização é baseada em `Formas normais`, que são REGRAS, que devem ser **seguidas por uma tabela** para que ela seja considerada eficiênte.

DEPENDÊNCIA FUNCIONAL

Revela qual será a relação **entre** *os atributos dentro de uma tabela*; A `dependência funcional` é a base na NORMALIZAÇÃO DE DADOS, uma vez que é através dela que é realizada a **conversão de dados** de *estruturas complexas de dados* em *estruturas estáveis e simples* utilizando as `Formas normais`.

Conceito de BD e fundamental na programação.

> CONTEÚDO HYPERLINK

A `normalização de dados` https://www.devmedia.com.br/guia-simplificado-para-as-5-formas-normais-artigo-revista-sql-magazine-87/21043 é um processo criado em 1970, logo após o reconhecimento das `formas normais` https://learn.microsoft.com/pt-br/office/troubleshoot/access/database-normalization-description

As `formas normais` são passos descritivios sobre a organização dos dados do banco de dados.

O ato de `Normalização os dados` é desmembrar as tabelas em tabelas mais específicas, gerando um aumento do número de tabelas, por consequência o aumento de manutenção e **comprometer a performance inicial**
Normalização deve ser realizada por bom senso, e o horizonte de melhorias é de médio longo prazo.

Link slide Proz https://slides.com/administrador/hipertexto02_banco_de_dados_i_proz?token=k-RpWtJf#/1

7.1.3.2 - Primeira Forma Normal - 1FN

> Normalização tem como objetivo evitar anomalias nas tabelas.

1FN;

- **não existem grupos de atributos repetidos**;
- suas relações **não possuem atributos multivalorados**;
- cada *linha* **deve** representar **um registro** e cada *célula* **deve** conter um **valor único**;

Uma tabela 1FN deve estar de acordo com as seguintes informações:

- os dados **devem estar em valores atômicos**, ou seja, indivisíveis;
- ter apenas **um dado por coluna**;
- deve existir **pelo menos uma chave primária**;

- se existirem atributos `multivalorados`, uma **nova tabela será criada**;
- se existirem atributos `compostos`, eles serão **desmembrados em novas colunas** de dados.

Exemplo da 1FN

Exemplo da 1FN

Pessoa			
CPF	Nome	Endereço	Contato
123.456.789-10	Maria Dias	Rua Lima, 42, São Paulo, SP, Brasil	21 1234-5678
123.456.789-20	Carlos Monteiro	Rua do Barbalho, 56, Recife, PE, Brasil	35 0125-8745 35 8930-4567
123.456.789-30	Kauan Queiroz	Avenida Brasil, 123, São Paulo, SP, Brasil	41 5644-2310 41 0011-2233

Para que essa tabela esteja na 1FN, precisamos organizar alguns dados. Percebemos que nela existem atributos multivalorados, que são aqueles que podem ter mais de um valor, como o **Contato**, e atributos compostos, que são aqueles que podem ser divididos em atributos menores, como o **Endereço**. Então, é preciso modificar isso, pois não é permitido nesse tipo de forma normal.

Para que essa tabela esteja na 1FN, precisamos organizar alguns dados. Percebemos que nela existem atributos `multivalorados`, *que são aqueles que podem ter mais de um valor*, como o **Contato**, e atributos `compostos`, *que são aqueles que podem ser divididos em atributos menores*, como o **Endereço**. Então, é preciso modificar isso, pois nã**o é permitido** nesse tipo de *forma normal*.

Nos atributos `compostos` ocorre a *decomposição dos dados*, em que **cada dado é adicionado em uma coluna**. Assim, o atributo endereço será subdividido em atributos menores.

Exemplo da 1FN

Logradouro	Número	Cidade	UF	País
Rua Lima	42	São Paulo	SP	Brasil
Rua do Barbalho	56	Recife	PE	Brasil
Avenida Brasil	123	São Paulo	SP	Brasil

No caso dos atributos `multivalorados`, **criamos uma nova tabela** com os dados. Na nova tabela **Contato**, a `chave primária-PK` é o *identificador* e a `chave estrangeira-FK` é o *CPF* correspondente da tabela **Pessoa**. Se um CPF possuir dois contatos, eles devem ser apresentados em linhas diferentes na tabela Contato, como é o caso dos CPFs dos identificadores 4 e 5.

Exemplo da 1FN

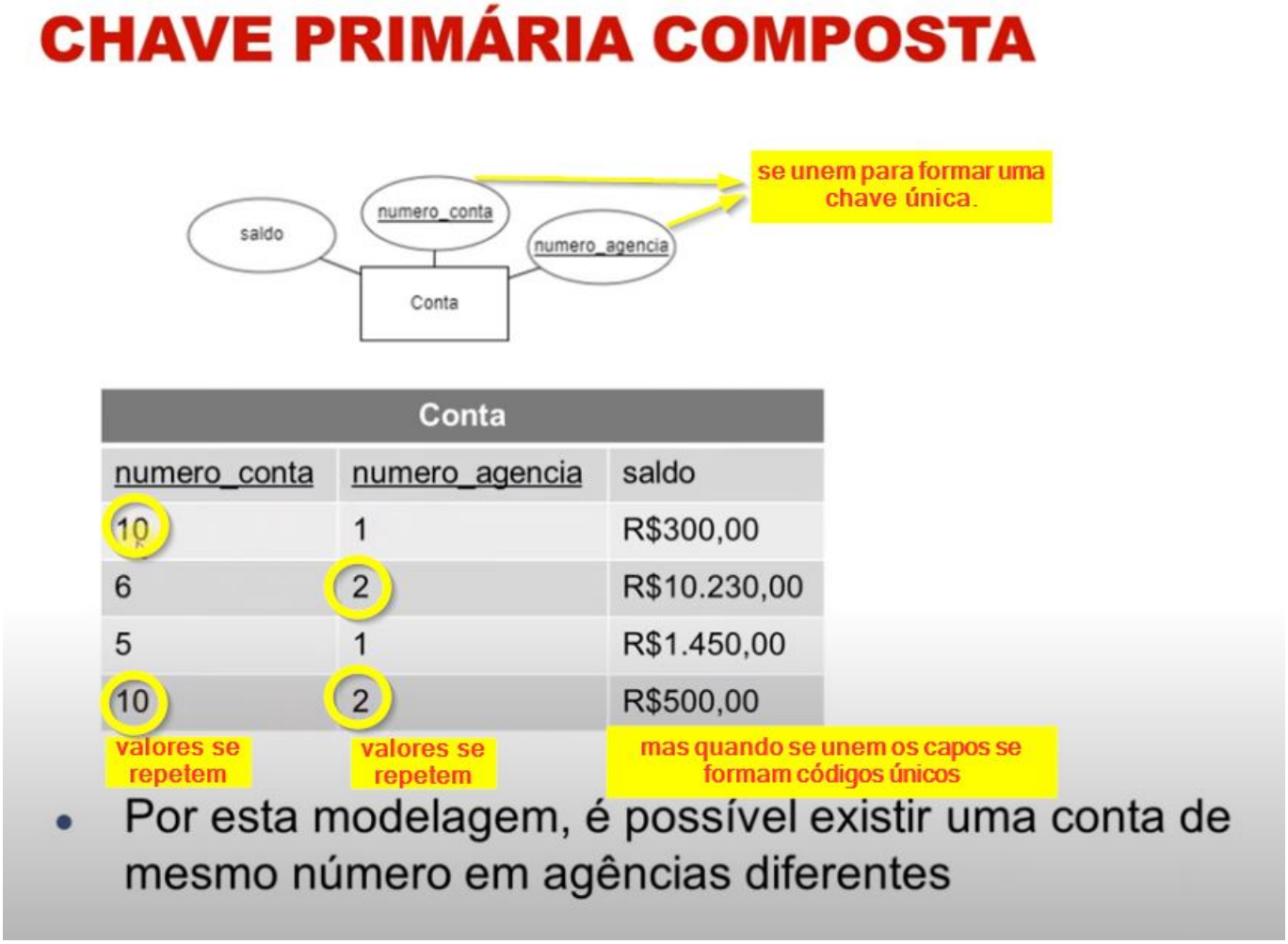
Contato		
Identificador	CPF	Contato
1	123.456.789-10	21 1234-5678
2	123.456.789-20	35 0125-8745
3	123.456.789-20	35 8930-4567
4	123.456.789-30	41 5644-2310
5	123.456.789-30	41 0011-2233

7.1.3.3 - Segunda Forma Normal - 2FN

A 2FN, tem que ter tudo que a 1FN exige.

- não deve ter atributos compostos ou multivalorados e nem dependências parciais.

Explicação sobre chave primária composta:



No caso das chaves `primárias compostas`, a 2FN **exige que todas as relações tenham dependência total**. Ou seja, uma chave primária composta **determina** funcionalmente os atributos, *não podendo existir um atributo que depende apenas de uma parte dessa chave.* Não pode ser composta.

Portanto, na 2FN, os dados:

- já estão na 1FN;
- a chave primária precisa ser simples.

Exemplo da 2FN

Identificador	Indentificador_ função	Nome	Função	Descrição	Data_inicio
1	103	Carla	Programadora	Desenvolvedora Python	08/03/2020
2	87	Rian	Designer	Comunicação visual	12/03/2021
1	104	Carla	Programadora	Desenvolvedora Java	08/03/2020
4	01	Camila	Gerente	Planejamento	18/11/2021
5	54	Pedro	Analista	Analisa resultados	01/04/2022
4	54	Camila	Gerente	Analisa resultados	01/04/2022

Essa tabela está na 1FN, **não possuindo dados multivalorados e nem atributos compostos.** Portanto, como não está na 2FN, ela precisa de ajustes.

Nesta tabela, temos chaves primárias compostas, o `IDENTIFICADOR` e o `IDENTIFICADOR_FUNÇÃO`. Com o atributo NOME, conseguimos saber qual é o nome da pessoa, sendo um atributo que depende parcialmente da chave primária.

Também conseguimos saber qual é a função da pessoa pelo número do identificador e a descrição pelo identificador da função. Com isso, podemos ter duas tabelas ao invés de apenas uma.

Para transformar essa tabela na 2FN, será necessário desmembrar alguns dados e construir outra tabela. Assim, nenhuma coluna depende parcialmente de outra.

Para que a tabela original fique na 2FN, ela precisa ser dividida em duas. Com esse diagrama, temos a organização de como as duas tabelas seriam criadas com os seus devidos atributos para que não haja chave primária composta. No relacionamento das entidades, uma pessoa pode possuir uma ou várias funções e uma função pode ter uma ou várias pessoas envolvidas.

7.1.3.4 - Terceira Forma Normal - 3FN

A Terceira Forma Normal (3FN) tem que estar, obrigatoriamente, na 2FN e possuir dependência exclusiva da chave primária da tabela.

> Todos os seus atributos devem ser funcionalmente independentes.

Na 3FN, os dados:

- devem estar na 2FN;

- Os atributos não podem depender de outros que não são chaves.

Exemplo da 3FN

Nota_fiscal	Identificador_vendedor	Nome_vendedor	Codigo_Produto	Quantidade_vendido
12345	001	Carol	0123	200
65789	010	Ruan	9012	50
32456	100	José	3210	120
87689	001	Carol	5643	30

Essa tabela não está na 3FN, pois os seus atributos não são todos dependentes. A Nota_fiscal é a chave primária e sabemos o Nome_vendedor através do Identificador_vendedor. Com isso, temos uma dependência.

Para resolver isso, precisamos criar uma nova tabela com esses atributos que são dependentes.

Exemplo da 3FN

Vendedor	
Identificador_vendedor	Nome_vendedor
001	Carol
010	Ruan
100	José
001	Carol

Agora, temos uma tabela `Vendedor` com a sua chave primária `Identificador_vendedor` e os nomes de cada vendedor.

Exemplo da 3FN

Nota_fiscal	Identificador_vendedor	Nome_vendedor	Codigo_Produto	Quantidade_vendido
12345	001		0123	200
65789	010		9012	50
32456	100		3210	120
87689	001	Carol	5643	30

Os nomes dos vendedores *são excluídos desta tabela*, porém ela continua com o `Identificador_vendedor`, que **se torna uma chave estrangeira** nesta tabela, pois esse atributo é uma `chave primária` na tabela `Vendedor`.

7.1.3.5 - Forma Normal de Boyce-Cood - FNBC

Cada forma normal é estritamente mais forte do que a anterior, então:

- toda relação da 2FN encontra-se na 1FN;
- e toda relação na 3FN encontra-se na 2FN.

O mesmo acontece com a Forma Normal de Boyce-Codd (FNBC). Toda relação FNBC está na 3FN.

Escola		
Aluno	Disciplina	Professor
Mel	Introdução a Programação	Rafael
Pietro	Programação Orientada a Objetos	Cintia
Carlos	Teoria da Computação	Queiroz
Rita	Introdução a Programação	Maria
Mel	Programação Orientada a Objetos	Tiago
Carlos	Introdução a Programação	Rafael

Analisando essa tabela, conseguimos entender que:

- descobrimos o professor se soubermos quem é o aluno e qual é a disciplina;
- descobrimos qual é a disciplina se soubermos quem é o professor;

Assim, existem dependentes entre os atributos.

Para transformar essa tabela na FNBC, é necessário decompor as relações que são dependentes. Neste caso, essa tabela é decomposta da seguinte forma:

- Professor, Disciplina;
- Aluno, Professor.

Forma Normal de Boyce-Codd (FNBC)

Professor:PK	Disciplina
Rafael	Introdução a Programação
Cintia	Programação Orientada a Objetos
Queiroz	Teoria da Computação
Maria	Introdução a Programação
Tiago	Programação Orientada a Objetos
Rafael	Introdução a Programação

Após a decomposição, temos uma tabela com a `chave primária` Professor e sua respectiva Disciplina.

Aluno:PK	Professor:FK
Mel	Rafael
Pietro	Cintia
Carlos	Queiroz
Rita	Maria
Mel	Tiago
Carlos	Rafael

Após a decomposição, temos outra tabela com a chave primária Aluno e Professor como chave estrangeira, pois, a partir do professor, podemos saber qual disciplina o aluno irá cursar.

Vimos no exemplo anterior como transformar uma tabela para a FNBC, utilizando a função de decompor os atributos, construindo novas tabelas para eliminar as dependências.

> leitura Complementar

HEUSER, Carlos Alberto. Projeto de Banco de Dados. 2 ed. Rio de Janeiro: Campus, 2008.

MONTEIRO, Emiliano Soares. Projetos de Sistemas e Bancos de Dados. Rio de Janeiro: Brasport, 2004.

> Referência Bibliográfica

DATE, Christopher J. Introdução a Sistemas de Bancos de Dados. Rio de Janeiro: Campus, 2004.

ELMASRI, Ramez E., NAVATHE, Shamkant B. Sistemas de Banco de Dados. 4 ed. São Paulo: Pearson Addison-Wesley, 2008.

OPPEL, Andy. Banco de Dados Desmistificado. Rio de Janeiro: Alta Books, 2004.

SILBERSCHATZ, Abraham, KORTH, Henry F., SUDARSHAN, S. Sistema de Banco de Dados. 5 ed. Rio de Janeiro: Campus, 2006.

7.1.3.6 - Anotação Exercícios

1. O que causaria uma anomalia de atualização nessa tabela?

AGÊNCIA		
NOME	NUMERO_AGÊNCIA	ENDEREÇO_AGÊNCIA
MARIA	060	RECIFE
PAULO	135	CARUARU
FABIANA	060	RECIFE
ERIK	060	RECIFE
RONALD	135	CARUARU

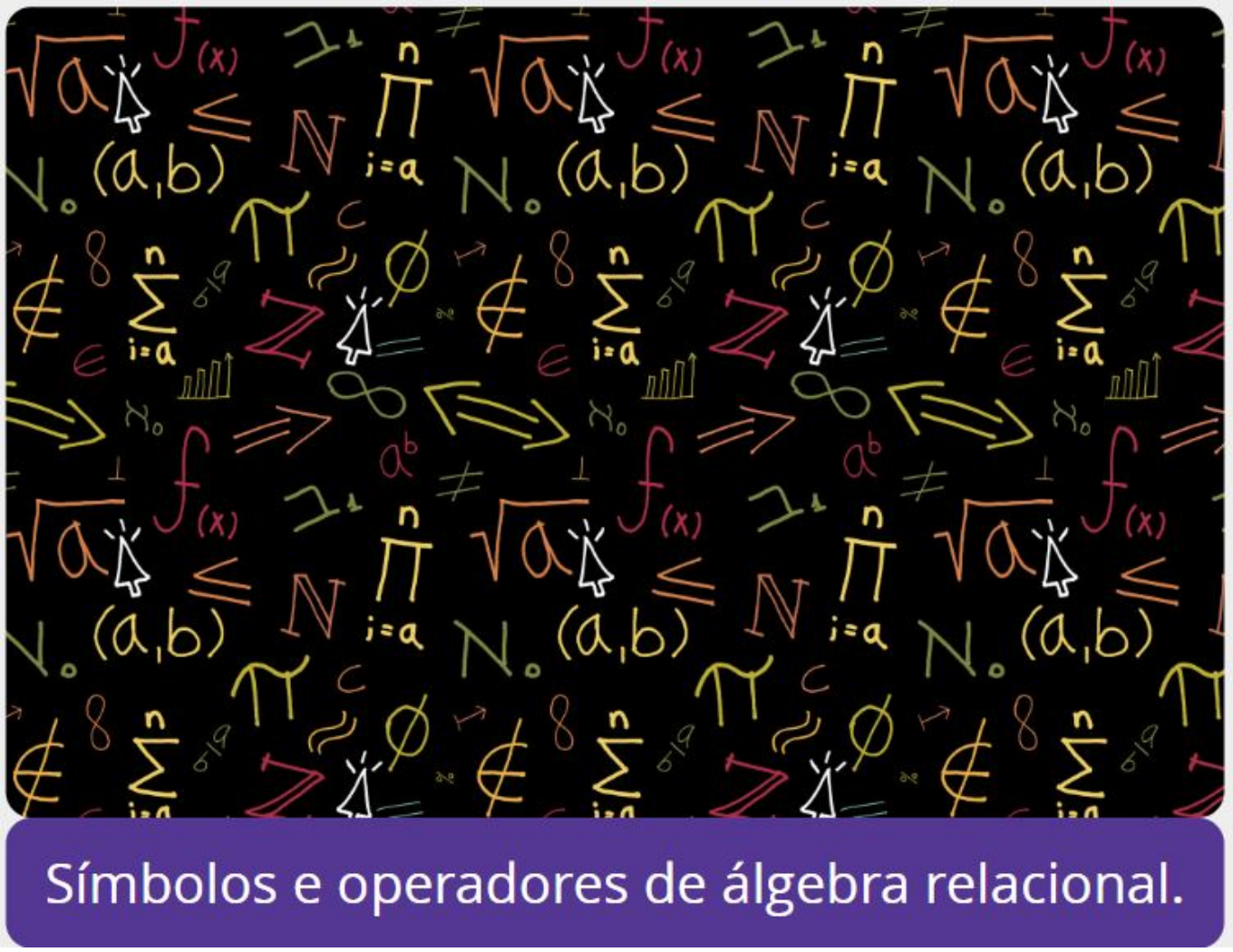
    1. Se o endereço de uma das agências for atualizado, ele precisa ser atualizado para todos os funcionários ou ocorrerá uma anomalia de atualização.
        1. Resposta correta!Isso mesmo! Se o local não for atualizado para os funcionários, eles podem ter o mesmo número de agência com endereços diferentes.
2. Podem surgir diversas anomalias em um banco de dados. Analisando as tabelas, o que causaria uma anomalia de exclusão?

    1. Excluir NUMERO_AGENCIA da tabela Agência, pois, assim, o funcionário não teria vínculo com a agência.
        1. Resposta correta!Isso mesmo! A tabela Funcionário depende de um registro da tabela Agência, que, caso seja excluído, tudo que estiver relacionado a ele também deve ser excluído para a anomalia não ocorrer.
3. Qual é a função da normalização no desenvolvimento de um banco de dados?
    1. Prevenir anomalias e aumentar o desempenho do banco de dados.
        1. Resposta correta!Isso mesmo! Com o uso da normalização, a manutenção do banco de dados se torna mais prática.

7.2 - MÓDULO 02

7.2.1 - Linguagens formais de consulta a bancos de dados - Álgebra relacional e Cálculo relacional

Na *criação de um banco de dados*,**o principal objetivo é o armazenamento.** Os dados devem ser persistentes e íntegros, pois as suas informações serão trabalhadas.
Usando as consultas que são implementadas na linguagem SQL, **esses dados podem ser recuperados para o usuário ou para a aplicação** do banco de dados.
Essa linguagem foi desenvolvida com base em outras de `consultas formais` como a **álgebra relacional**.



7.2.2 - Álgebra relacional

A `álgebra relacional` **é uma linguagem de consulta**, onde *os usuários passam instruções* para o sistema executar e o mesmo executa o conjunto de instruções *a fim de se obter um resultado desejado pelo usuário*.
O **objetivo das instruções** é *acessar o banco de dados e coletar a informação* que o usuário deseja.
A álgebra relacional é a base formal para IMPLEMENTAÇÃO da linguagem `SQL`. Essa que é uma linguagem comercial usada para **aplicar estratégias disponíveis** no módulo de processamento de consultas interno dos Sistemas de Gerenciamento de Banco de Dados (SGBDs).
Esses `módulos` **incluem a funcionalidade da otimização de consultas**, as quais são realizadas com o SGBD para extrair os dados que estão no banco.
[Otimização de consultas]https://www.devmedia.com.br/10-tecnicas-de-otimizacao-de-consultas-sql/39499

Nomeclaturas encontradas na Algebra Relacional

Algebra	MER
Tuplas	Linhas da Tabela
Atributos	Colunas da Tabela
Relação	A Tabela

> LEMBRE-SE
> O **DER** é uma representação mais simples e visual das relações entre entidades, enquanto o **MER** é mais detalhado e abstrato, descrevendo a estrutura do banco de dados de uma maneira mais técnica. Ambos são usados no processo de design de banco de dados para ajudar na compreensão e implementação de um sistema de banco de dados.

7.2.2.1 - Operação *select*

O operador `select`, ou a o`perações de seleção`, seleciona tuplas das tabelas para atender a uma solicitação e extrair informações.

[Operador select]
https://www.devmedia.com.br/algebra-relacional-tutorial/2663

Simbologia	σ
Sintaxe	σ (relação)
Exemplo	σ salario>=2000 (EMPREGADO)

No exemplo, podemos entender que em uma tabela de nome `EMPREGADO`, serão selecionadas as tuplas que possuem o valor de salário maior que dois mil.

A letra grega σ (sigma) representa a operação select, e a sua representação é da seguinte forma :

$$\sigma_{\text{<condição da seleção>}}(\text{<nome da relação>})$$

A expressão booleana `<condição da seleção>` é aplicada sobre atributos da tabela e `<nome da relação>` é em qual tabela a operação select será aplicada.
Neste exemplo, temos um operador `select` para selecionar as tuplas de uma tabela `Aluno` que possuem a nota do aluno **maior que 7,0.** Observe como o comando é utilizado em álgebra relacional:

$$\text{Consulta1} = \sigma_{\text{NOTA} > 7{,}0}(\text{ALUNO})$$

Como resultado da extração, temos a seguinte tabela:

Consulta 1			
Nome	Matrícula	Disciplina	Nota
João	33333	História	8,0
Joaquim	56544	Geografia	7,5

Neste caso, o operador `select` seleciona as tuplas de uma tabela de nome `DEPENDENTE`. Os dados extraídos devem ser de filho e que seja do *sexo masculino*, utilizando o camando `AND`, pode-se consultar uma informação juntamente com outra, sendo *filho* e do sexo masculino.

$$\text{Consulta2} = \sigma_{(\text{relação} = \text{“Filho”}).\text{and}.(\text{sexo} = \text{“Masc”})}(\text{DEPENDENTE})$$

Resultado:

Consulta 2				
Reg_Resp	Nome_Dep	Data_Nasc	Relação	Sexo
54215	Flávio	12/12/2014	Filho	Masc

Os **operadores relacionais** utilizados nas operações select são `<`, `>`, `≤`, `≥`, `=` e `≠`, e os **operadores booleanos** são `and`, `or` e `not`.

7.2.2.2 - Operação *project*

A operação de **projeção**, ou `project`, *permite **escolher e separar** um conjunto determinado de **colunas** de uma tabela*. Observe a representação:

Simbologia	π
Sintaxe	π (relação)
Exemplo	NOME, SALARIO (EMPREGADO)

No exemplo, podemos entender que em uma tabela de nome `EMPREGADO`, **serão selecionadas as colunas** que possuem o `nome` e o `salário` dos empregados.

[Operação de projeção] https://www.youtube.com/watch?v=v-CaKZG8k_s

A operação `project` é representada pela letra grega π (pi), e a sua representação é da seguinte forma:

π <lista de atributos> (<nome da relação>)

A parte `<lista de atributos>` representa a lista de atributos que o usuário selecionará na tabela.

A parte `<nome da relação>` é em qual tabela vai ser aplicada a operação project.

Neste exemplo, temos um project para selecionar as colunas **Nome** e **Nota** da tabela **Aluno**. Então, para que as colunas sejam extraídas, **a nota do aluno deve ser maior que 7,0**:

$$Consulta3 = \pi_{NOME,\ NOTA > 7,0}(ALUNO)$$

Consulta 3	
Nome	Nota
João	9,0
Joaquim	7,5
Felipe	8,0
Francisco	9,5
Luiz	8,5

7.2.2.3 - Sequencialidade de operações

Na sequencialidade de operações, podemos combinar as operações de project e select. Assim, é possível extrair tuplas e colunas ao mesmo tempo.

Sua forma geral é:

$$\pi_{<lista\ de\ atributos>}(\sigma_{<condições\ de\ seleção>}(nome\ da\ relação))$$

[Sequencialidade de operações] https://www.youtube.com/watch?v=E-tIwv8jDhE&t=15s

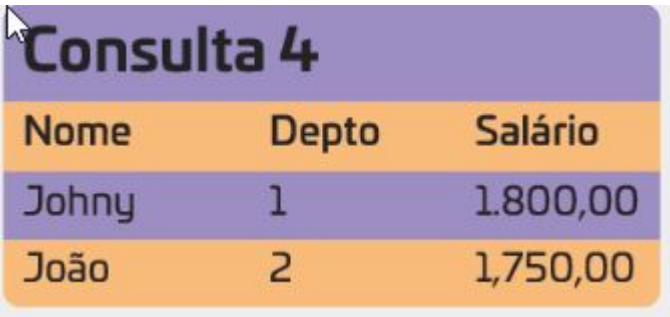
Neste exemplo, temos uma **combinação de operações**, onde dados são extraídos de uma tabela (func)Funcionário. As **colunas** desta tabela a serem extraídas são:

- Nome,
- Depto(departamento) e
- Salário.

as **tuplas** que serão extraídas deve apresentar o *valor de salário menor que dois mil*

$$Consulta\ 4 = \pi_{nome,\ depto,\ salario}(\sigma_{salário < 2000}(func.))$$

Ao realizar essa extração, será apresentada a seguinte consulta, com o nome de dois funcionários, o departamento em que trabalham e o salário que recebem.



Consulta 4

Nome	Depto	Salário
Johny	1	1.800,00
João	2	1,750,00

7.2.2.4 - Como funcionalm as operações matemáticas dentro dos conjuntos

As relações também **podem ser tratadas como conjuntos**, logo, pode-se aplicar operações matemáticas sobre elas.

As operações são:

- **União** - representada pelo símbolo `U`;
- **Interseção** - representada pelo símbolo `∩`; e
- **diferença** - representada pelo símbolo `≠`.

Esses operadores são **NÃO UNÁRIOS**, isso quer dizer que eles **podem ser utilizados em mais de uma tabela**.

7.2.2.5 - Operações não unárias

As operações não unárias **precisam**, **necessariamente**, **de mais de uma relação para acontecer**.

Elas `atuam em relações compatíveis`, ou seja, **aquelas que possuem o mesmo** grau, ou mesmo **número**, de **atributos** que têm o mesmo domínio.

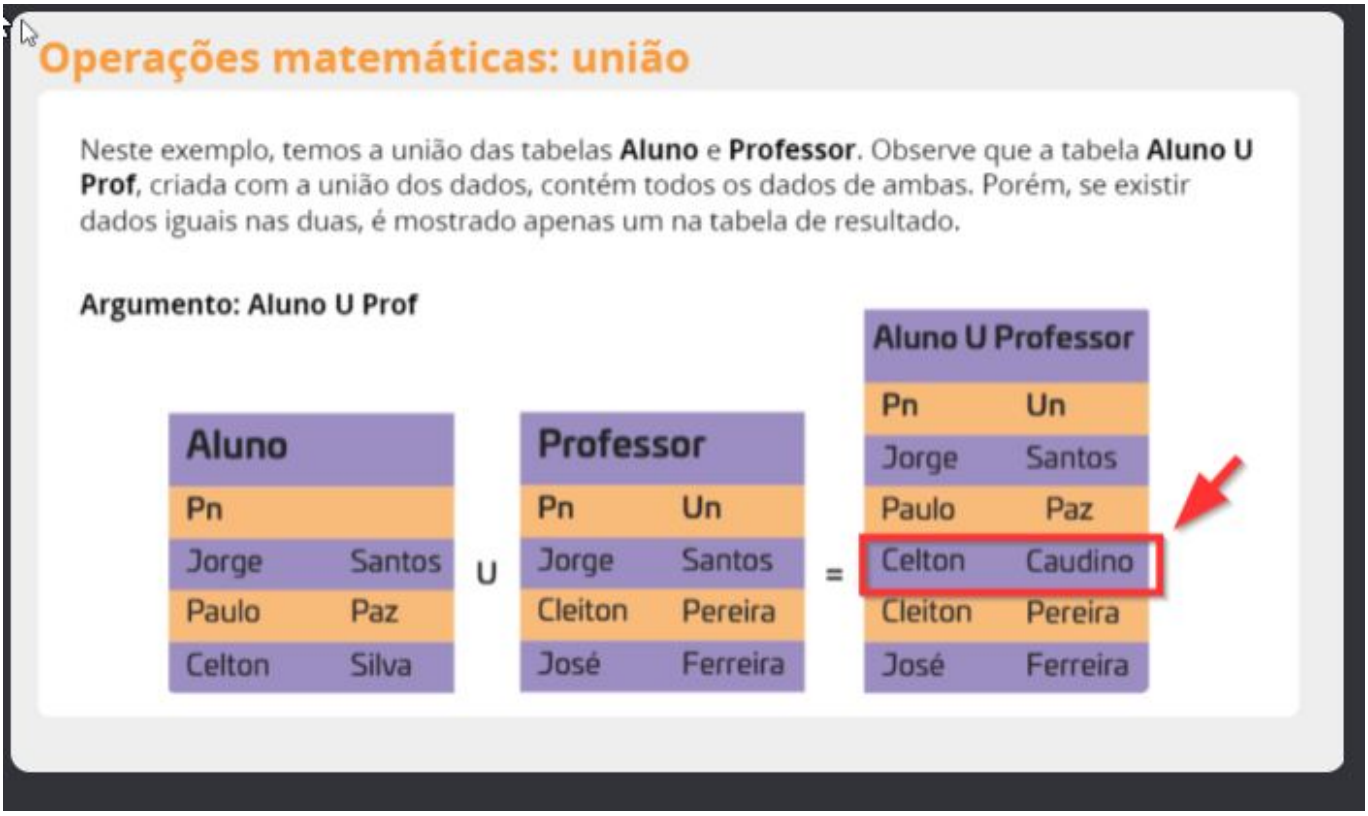
7.2.2.6 - Operações matemáticas: União

A união entre os conjuntos **R** `U` **S**, por exemplo, vai **retornar todas** as `tuplas de R`, `de S` e as em `comum entre R e S`, ou seja, a `intersecção`.

[União entre Conjuntos] https://www.youtube.com/watch?v=J3BlrfLkC0A

Neste exemplo, temos a união das tabelas `Aluno` e `Professor`. Observe que a tabela `Aluno` **U** `Prof`, criada com a união dos dados, contém todos os dados de ambas. **Porém, se existir dados iguais** nas duas, é mostrado **apenas um** na tabela de resultado.

OBERVAÇÃO



Não entendi o motivo desse resultado. Entrei em contado com o grupo.

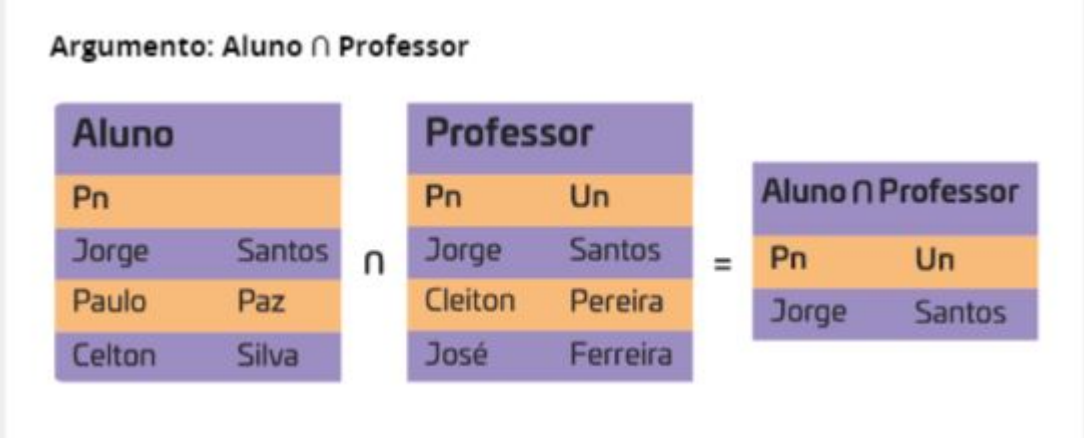
7.2.2.6 - Operações matemáticas: Intersecção

A operação de `intersecção` **retorna apenas o que existe em comum entre duas tabelas** `R ∩ S`.

[Operação de Intersecção] https://www.youtube.com/watch?v=I93r3XilNxA

Então, o seu **resultado** é uma `nova relação` que inclui todas as tuplas que estão em R, em S e na intersecção entre ambas, **retornando apenas o que existe em comum entre as duas relações**.

Aqui, temos a intersecção das tabelas Aluno e Professor. Observe que a tabela `Aluno ∩ Professor`, criada com a intersecção dos dados, **contém todos os dados que são iguais em ambas as tabelas**.



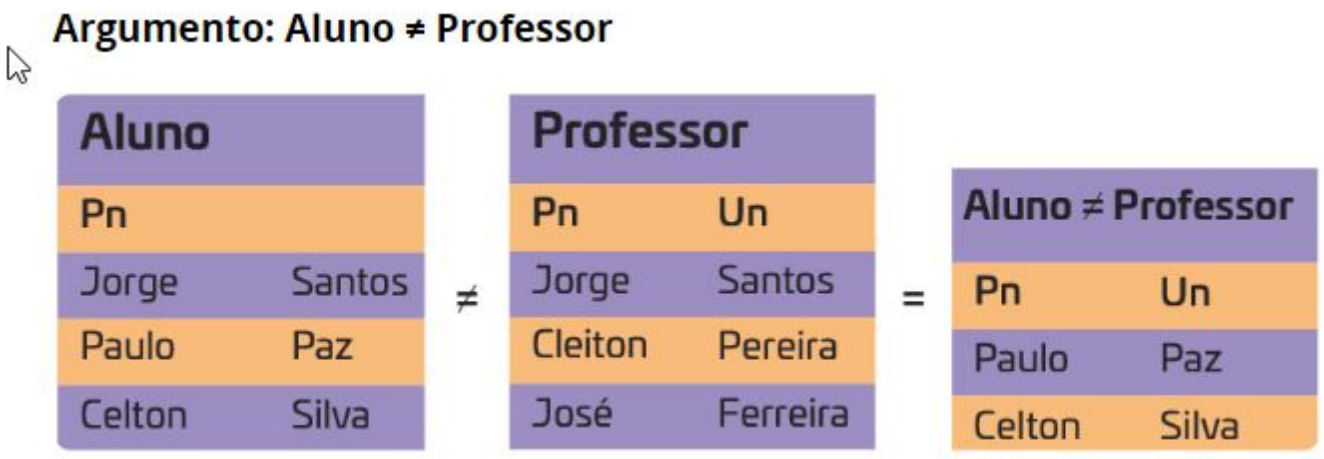
7.2.2.7 - Operações matemáticas: Diferença

A operação de `diferença` também é não unária, ou seja, precisa de mais de uma tabela. Ela **retorna** `todas as tuplas pertencentes a R` mas que **não pertencem a S**.

[Operação de diferença] https://www.youtube.com/watch?v=fr29QmtyPlw

> É importante saber!
> que ela **NÃO É COMUTATIVA**, diferente das operações de **união** e de **intersecção**. logo R - S é Diferente de S - R.
> **RELEMBRE**: Comutatividade : É a propriedade que diz que a ordem em que você coloca os blocos sobre a mesa não importa, o resultado é o mesmo. Igual a os números, independente da ordem na adição ou multiplicação é o mesmo.

Aqui, temos a diferença das tabelas `Aluno` e `Professor`. Observe que a tabela `resultado da diferença` **contém todos os dados que pertencem à tabela Aluno** e `não pertencem à tabela Professor`.



7.2.2.8 - Cálculo relacional

`Cálculo relacional` **é uma linguagem formal de consultas** para banco de dados *relacionais*. Ela é `não procedimental`, ou seja, **a ordem não importa**, diferentemente da **álgebra relacional** que as operações **devem ser executadas em uma determinada ordem**.

[Cálculo relacional] https://www.youtube.com/watch?v=doJ1dI00Xbg

A expressão criada para executar uma operação em cálculo relacional **independe da ordem e específica** o que deve ser recuperado, invés de como recuperar.

Além disso, **toda expressão criada em cálculo relacional** `consegue se expressar na álgebra relacional`, o mesmo acontece ao contrário.

O `cálculo` e a `álgebra relacionais` **são linguagens que definiram a base de consultas para bancos de dados relacionais**.

7.2.2.9 - Álgebra relacional e o cálculo relacional

A `álgebra relacional` é responsável por **definir as operações do modelo relacional** e o `cálculo relacional`, por **especificar uma consulta declarativa**, já que *é uma linguagem de consulta formal e utiliza a expressão declarativa*.

[Álgebra relacional]
https://sites.google.com/site/fkbancodedados1/algebrarelacional

[Cálculo relacional]
https://www.ime.usp.br/~jef/calcrelac.pdf

[Cálculo]
https://www.youtube.com/watch?v=4B5PJREk2YI&list=PLz7W_DVMdJkHUmwwzgi8ZgiYSkA7zFLD7

> Leitura Complementar

[DESCOMPLICA, OLIBA!. Álgebra Relacional em Bancos de Dados - Operações Unárias (Seleção, Projeção). YouTube, 6 jul. 2020.] https://www.youtube.com/watch?v=E-tlwv8jDhE&list=PLsVF4k1EODtGJcnizYVGjgAexY9idkEYH

[DESCOMPLICA, OLIBA!. Operações sobre Conjuntos de Dados - União, Intersecção e Diferença - Álgebra Relacional de Dados. YouTube, 20 jul. 2020.] https://www.youtube.com/watch?v=J3BlrfLkC0A&list=PLsVF4k1EODtGJcnizYVGjgAexY9idkEYH&index=3

> Referências Bibliográficas

DATE, Christopher J. Introdução a Sistemas de Bancos de Dados. Rio de Janeiro: Campus, 2004.

ELMASRI, Ramez E., NAVATHE, Shamkant B. Sistemas de Banco de Dados. 4 ed. São Paulo: Pearson Addison-Wesley, 2008.

HEUSER, Carlos Alberto. Projeto de Banco de Dados. 2 ed. Rio de Janeiro: Campus, 2008.

MONTEIRO, Emiliano Soares. Projetos de Sistemas e Bancos de Dados. Rio de Janeiro: Brasport, 2004.

OPPEL, Andy. Banco de Dados Desmistificado. Rio de Janeiro: Alta Books, 2004.

SILBERSCHATZ, Abraham, KORTH, Henry F., SUDARSHAN, S. Sistema de Banco de Dados. 5 ed. Rio de Janeiro: Campus, 2006.

7.2.2 - CONHECER OS SGBDs MAIS UTILIZADOS NA ATUALIDADE

7.2.2.1 - Vídeo - Conceitos básicos de SGBD - Conhecendo Ferramentas

SGBDs, FERRAMENTAS ESSENCIAIS PARA IMPLEMENTAR BD, uma vez que é por ele que é criada e manipulada os dados, criar relações entre tabelas, copiar e eliminar arquivos, alterar as estruturas dos campos, criar e gerir usuários e importar e exportar dados. E muito mais.

Alguns SGBDs:

- **ORACLE**, mais utilzado atualmente nas corporações, utiliza PLSQL. Funções: Acelar o tratamento e a persistência dos dados, seguro, confiável, robusto, alta qualidade.
- **MySQL**, muito popular, usado principalmente na internet, utilizado em projetos simples e complexos, utiliza linquagem SQL. licença paga e gratuita. Utilizado pela NASA, BRADESCO, YOUTUBE.
- **SQL SERVER**, criado e mantido pela Microsoft, robusto e atente uma demanda de negócios eimples e complexos, utilizado em grande volumes de dados, uma vez que usa IA para proporcionar detecções mais certeir4as de tendências e comportamentos por meio de dados.
- **Postgree SQL**, `SGBD de código aberto`, Executada pela linguagem PSQL, suporta grande parte do padrão SQL, além de ter features como visualizações atualizaveis, integridade transacional, controle de multvesão, consultas complexas, chaves estrangeiras, e muito mais.
- **FireBird**, `SGBD de código aberto`, atende clientes e servidores, compatibilidade SQL-92, funciona em mais de 10 sistemas operacionais. Foi desenvolvido apartir do código fonte do interbase6.
- **MariaDB**, `SGBD de código aberto`, construído pelos desenvolvedores do MySQL. A diferença entre MariaDB e MySQL é a licença, Maria é GPL, enquanto MySQL tem dupla, paga e gratuita.
- **MongoDB**, armazena seus registros sem nenhuma preocupação com a estrutura dos dados, orientado a documentos, é um Banco de Dados NoSQL. Dados NÃO ARMAZENADOS EM TABELAS, como é o caso dos bancos relacionais.

7.2.2.2 - Conceitos básicos de SGBD

O SGBD é um conjunto de aplicações que possibilita a manipulação e a criação de bancos de dados.

Ele tem a função de manter os dados com uma descrição completa e se responsabiliza pela forma como eles são armazenados.

[SGBD]https://dicasdeprogramacao.com.br/o-que-e-um-sgbd/

No SGBD, há um `catálogo de dados` cuja **função é armazenar informações** contendo a estrutura de cada arquivo, seu **tipo**, **formato** de armazenamento e **restrições**. Elas ficam **armazenadas no dicionário de dados** e são chamadas de `metadados`.

Uma representação conceitual dos dados deve ser fornecida, sem apresentar muitos detalhes do armazenamento, pelo SGDB ao usuário.

No **dicionário de dados, ou catálogo de dados**, **encontra-se a organização e a estrutura detalhada de cada arquivo**. As consultas complexas são realizadas pelos usuários técnicos já familiarizados com o SGBD.

[Modelagem conceitual]https://www.dpi.inpe.br/livros/bdados/cap3.pdf

As principais **características** do SGBD são:

- gerenciar grandes volumes de dados;
- eliminar a redundância e a inconsistência de dados;
- facilitar o armazenamento e acesso aos dados;
- garantir múltiplos acessos ao banco;
- garantir segurança aos dados; e
- garantir a integridade dos dados.

[Principais características SGBD]https://treinamento24.com/library/lecture/read/130632-quais-sao-as-principais-caracteristicas-de-um-sgbd

7.2.2.3 - Vantagens do SGBD

Uma **vantagem** do SGBD é o `controle da redundância`. Esse problema ocorre no processamento tradicional, em que o *mesmo dado pode ser guardado em diferentes tabelas*, gerando, assim, redundância e prejudicando o sistema.

[Vantagens do SGBD]https://ehgomes.com.br/banco-de-dados/sgbd-sistema-gerenciador-de-banco-de-dado-o-que-e/

Por exemplo, três tabelas com o mesmo dado gera excesso de informações no armazenamento. Com o SGBD, essa redundância é eliminada, o que **evita o desperdício** de espaço e passa a armazenar um único valor, não o mesmo valor três vezes. Observe na imagem.

Outras **vantagens** do SGBD são:

- compartilhamento de dados;
- restrição ao acesso não autorizado;
- representação de relacionamentos complexos entre dados;
- padronização, flexibilidade e redução de tempo do desenvolvimento de aplicações;
- disponibilidade de informações atualizadas; e
- economia de escala.

Desvantagens:

- alto custo;
- gerenciamento;
- manutenção.

Deve-se analisar se há a necessidade de adotar o SGBD em uma empresa pequena, já que é preciso adquirir outros softwares de alto custo para serem implementados.

7.2.2.4 - Arquitetura do SGBD

Arquiteturas do SGBD, são maneiras de intermediar a manipulação de dados para as aplicações utilizadas pelos usuários.

- **Plataformas centralizadas**
  - Essa arquitetura é voltrada para mainframes.
  - Alto custo
  - Soluções Centralizadas
  - SGBD armazenado em um computador de grande capacidade de processamento, onde ficam os emuladores para os aplicativos.
  - Uma grande vantagem é permitir a manipulação de um grande volume de dados por parte dos usuários.

[Mainframes]https://tecnoblog.net/responde/o-que-e-mainframe/

[Emuladores]https://pt.wikipedia.org/wiki/Emulador#:~:text=Na computação%2C um emulador é,de outros softwares sobre ele.

- **Sistemas de computador pessoal**

    - PC fazem sozinhos seus processamentos, utilizando o sistema `stand-alone`
    - Antes eram limitados, mas com a evulução dos hardwares já possuem uma excelente capacitdade de processamento.
    - SGBDs atuam como hospedeiros e terminais, usando o padrão `Xbase`. Arquitetura bem simples e é executada na máquina com um único aplicativo.

[Computador-Pessoal]https://blog.portaleducacao.com.br/#:~:text=Arquitetura Sistemas de Computador Pessoal Atualmente os computadores,trata de SGBD's%2C funcionam como hospedeiros e terminais.

- **Arquitetura Cliente-Servidor**
    - Processamento dividido entre dois sistemas. Reduz tráfego de rede.
    - Bastante popular. Nessa arquitetura o cliente solicita um determinado serviço para o servidor, enquanto o servidor trabalha para encontrar o dado solicitado o clinete está livre para realizar outras tarefas.
    - Envio de solicitações pelo usuário e retorno por parte do servidor com o dado. Tudo por meio de um hardware(Computador por exemplo).

[Arquitetura-Cliente-Servidor]https://www.canalti.com.br/arquitetura-de-computadores/arquitetura-cliente-servidor/#:~:text=Definição. Arquitetura cliente-servidor ou modelo cliente-servidor é uma,outro responsável pela obtenção dos dados (os clientes).

> Texto do CodePark Arquitetura cliente servidor é aquela em que o processamento da informação ocorre por meio de módulos divididos ou processos distintos. Nessa arquitetura há um processo reponsável pela manutenção da informação(servidor) e outro responsável pela obtenção dos dados (os clientes). **Cliente**, solicita um determinado serviço por intermédio de uma mensagem ao Servidor. Enquanto o servidor trabalha a mensagem o Cliente está disponível para realizar outras tarefas.

Servidor, oferece serviços a processos demandados pelo Cliente, executando a tarefa e retornando via resposta a tarefa, que se resume em dados.

Vantagens da arquitetura Cliente-Servidor:

- Recursos centralizados, essa vantagem se traduz pelo gerenciamento comum de recursos o qual tem como principal reflexo a minimização de redundâncias.
- Facilidade de manutenção, por meio dela é possivel substituir, reparar, atualizar/ realicar um servidor de seus clientes enquanto que não serão afetados por essa mudança.

Desvantagens da arquitetura Cliente-Servidor:

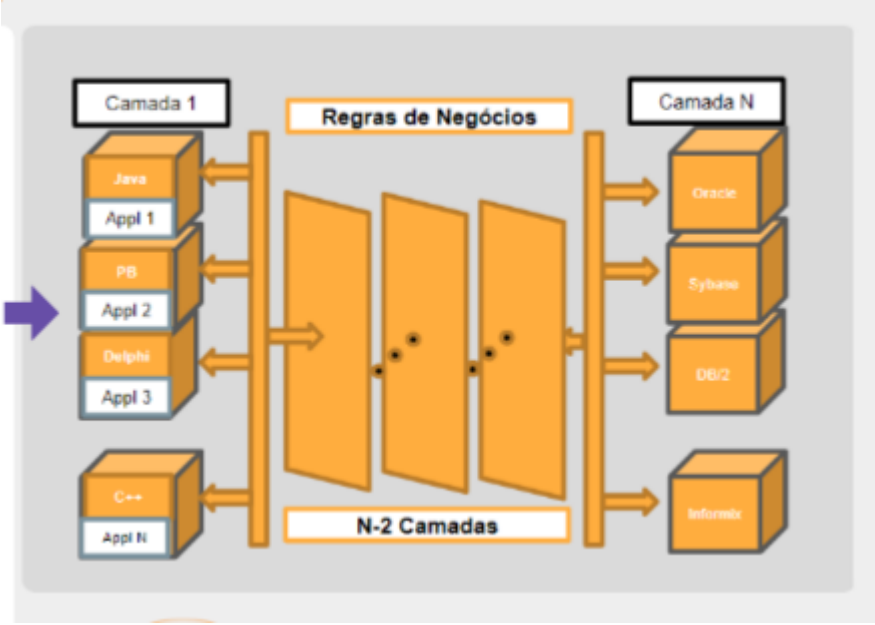
- Sobrecarga, ficará ele sobrecarregado caso receba solicitações simultâneas acima do suportado.

- Único nó, se torna sensível a falhas. Caso um servidor crítico deixe de funcionar os pedidos dos clientes não serão cumpridos. O que não ocorre em situações como P2P, onde os dados são compartilhados por enúmeros nós, e caso um servidor falhe, terá o mesma informação em outros.

- **Banco de Dados distribuído**

    - São necessários diversos servidores para distribuir as informações. Recebem e distribuem informações para diversas aplicações.

Neste exemplo, as informações estão distribuídas em vários servidores, ou seja, em N camadas. Na imagem, as caixas representam os dados. Quando um usuário fizer uma solicitação, como uma busca de dados, essa consulta é feita a qualquer um dos servidores e o sistema se encarrega de obter a informação, independente do servidor em que ela estiver.



Com isso, podemos entender que **a escolha de qual arquitetura SGBD para aplicar depende bastante do que precisamos armazenar**.

> Leitura Complementar [CURSO EM VÍDEO. Curso MySQL #01 - O que é um Banco de Dados?. 15 fev. 2016]https://www.youtube.com/watch?v=Ofktsne-utM&list=PLHz_AreHm4dkBs-795Dsgvau_ekxg8g1r Referências Bibliográficas DATE, Christopher J. Introdução a Sistemas de Bancos de Dados. Rio de Janeiro: Campus, 2004.

ELMASRI, Ramez E., NAVATHE, Shamkant B. Sistemas de Banco de Dados. 4 ed. São Paulo: Pearson Addison-Wesley, 2008.

HEUSER, Carlos Alberto. Projeto de Banco de Dados. 2 ed. Rio de Janeiro: Campus, 2008.

MONTEIRO, Emiliano Soares. Projetos de Sistemas e Bancos de Dados. Rio de Janeiro: Brasport, 2004.

OPPEL, Andy. Banco de Dados Desmistificado. Rio de Janeiro: Alta Books, 2004.

SILBERSCHATZ, Abraham, KORTH, Henry F., SUDARSHAN, S. Sistema de Banco de Dados. 5 ed. Rio de Janeiro: Campus, 2006.

7.2.2.5 - Anotações Exercícios

- Identifique o SGBD não relacional nas alternativas.
  - MongoDB
    - Resposta correta!MongoDB é orientado a um documento multiplataforma NoSQL.
- Sobre o SGBD, marque a alternativa correta.
  - Nesse sistema, existe um catálogo de dados que armazena informações com a estrutura de cada arquivo, o tipo, o formato de armazenamento e suas restrições.

        - Resposta correta!Essas informações estão armazenadas no dicionário de dados, ou catálogo de dados, e chamam-se de metadados.
- Assinale a alternativa que apresenta as vantagens do Sistema de Gerenciamento de Banco de Dados.
    - Controle de redundância, compartilhamento de dados e restrição a acesso não autorizado.
        - Resposta correta!Além dessas vantagens, o SGBD ainda tem: redução do tempo de desenvolvimento de aplicações e disponibilidade de informações atualizadas.
- Sobre a arquitetura de um SGBD, marque a alternativa correta.
    - Nesse sistema, os computadores pessoais fazem seus processamentos sozinhos, utilizando o sistema stand-alone.
        - Resposta correta!A principal vantagem dessa arquitetura é a simplicidade.

7.3 - MÓDULO 03

7.3.1 - Conhecer a linguágem SQL a linguagem do SGBD

SQL estrutura as ferramentas de geração de banco de dados.

SQL surgiu nos anos 70 por meio da empresa IBM. Nesse ano que surgiu o primeiro sistema de banco de dados.

Em 1980, com a união dos esforços do Instituto Nacional Americado de Padrões-ANSI e a Organização internacional para padronização-ISO, foi publicada a primeira padronização da linguagem SQL. Vem passando por transformações ao longo do tempo, em especial SQL-92 e SQL-99.

Atualmente, utilizada na maioria dos bancos de dados relacionais. Ferramenta essa com grande grau de poder na definição e manipulação dos dados.

> SQL **NÃO É LINGUÁGEM PROCEDURAL**(LINGUAGEM DE PROGRAMAÇÃO), E SIM DE **CONSULTA**. Essa linguagem apresenta subconjuntos de linguagem(SUBLINGUAGENS) para DEFINIÇÃO, MANIPULAÇÃO, CONSULTAS, CONTROLE E TRANSAÇÃO DE DADOS.

- **LINGUAGEM DE DEFINIÇÃO DE DADOS-DDL** Possui três comandos:

    1. `Create`, que cria a estrutura. Que pode ser uma tabela.
    2. `Alter`, altera uma estrutura. Ex: inserir uma coluna ou excluir uma coluna.
    3. `Drop`, Exclui uma estrutura caso exista no banco uma tabela que não sirva mais.

- **LINGUAGEM DE MANIPULAÇÃO DE DADOS-DML** Manipula as estruturas criadas pela DDL, e possui três comandos:

    1. `Insert`, inserir dados na tabela.
    2. `Update`, atualiza os dados da tabela.
    3. `Delete`, exclui registros das tabelas.

- **LINGUAGEM DE CONSULTA DE DADOS-DQL** Permite a recuperação e leitura dos dados inseridos nas tabelas dos bancos. Possui um comando:

    1. `Select`, que permite alteração dos dados, ordenação dos retornados e pode elaborar consultas simples e avançadas.

- **LINGUAGEM DE CONTROLE DE DADOS-DCL** Seus comandos são responsáveis pela segurança do banco de dados. Realiza Controle de acesso, onde é atribuindo ou retirado permissões de usuários. Possui dois comandos:

    1. `Grant`, habilita acesso a dados e operações.
    2. `Revoke`, que retira acesso.

- **LINGUAGEM DE TRANSAÇÃO DE DADOS-DTL** Entendendo que transação é um conjunto de operações SQL, temos o comando:

    1. `Start transaction`, para iniciar a transação.
    2. `Commit`, concretizar a transação.
    3. `Rollback`, para anular.

7.3.1.1 - Anotação Exercícios

1. Sobre a linguagem DTL, marque a alternativa correta.
    1. É a linguagem de transação de dados.
        1. Resposta correta!A DTL realiza transações através do comando START TRANSACTION. O comando COMMIT serve para concretizá-la e o ROLLBACK para anulá-la.
2. Qual é o principal comando utilizado pela DQL?
    1. O comando SELECT, que serve para permitir a alteração dos dados e ordenar dados retornados.
        1. Resposta correta!O comando SELECT pode elaborar consultas simples e avançadas.
3. Sobre a linguagem DML, marque a alternativa correta.
    1. É uma linguagem de manipulação de dados.
        1. Resposta correta!Essa linguagem manipula os dados através dos comandos INSERT, UPDATE e DELETE.
4. Quais são as principais sublinguagens que compõem a SQL?
    1. DDL, DML, DCL, DQL e DTL.
        1. Resposta correta!Essas são as principais sublinguagens que compõem a linguagem SQL.

7.3.2 - TIPOS DE DADOS EM SQL

Os tipos de dados em SQL são parecidos com os tipos de dados em programação. **A diferença é que em SQL existe uma preocupação com o tamanho do valor que será armazenado**.

7.3.2.1 - Valores máximos

Podemos imaginar cada coluna de uma tabela como um conjunto de caixas. Se formos guardar um par de sapatos, o melhor é usar uma caixa de sapato. Se formos guardar garrafas de água de dois litros, pode ser necessário usar um engradado. Já objetos muito grandes, como carros, precisam ser armazenados em contêineres.

Faz sentido usar recipientes maiores para objetos maiores. Porém, embora seja possível guardar um par de sapatos em um contêiner, fazer isso tem um alto custo e é um desperdício de espaço.

> O **espaço necessário para o armazenamento** é um dos fatores principais para **definir o preço na hora de contratar um serviço de banco de dados**.

Usando essa analogia, os dados que nosso sistema armazenará podem ser pequenos, como um username de até 16 caracteres; um pouco maiores, como um texto de uma postagem com 300 caracteres; ou podem ser um livro inteiro em formato PDF, com até 350 mil caracteres.

Com isso em mente, um único tipo de dado em programação, por exemplo, a `string`, *pode ter vários tipos de dados semelhantes nos bancos de dados*, como `char`, `varchar`, `text`, entre outros.

7.3.2.2 - Tipos de dados em programação e em SQL

Observe a seguir alguns tipos de dados os que mais nos interessam nesse momento:

Equivalente em programação	SQL	Descrição
String	CHAR	Tamanho fixo de 8.000 caracteres
	VARCHAR	Tamanho variável de até 8.000 caracteres
	TEXT	Tamanho variável de até 2 GB de dados

Equivalente em programação	SQL	Descrição
Number/ Integer	TINYINT	0 a 255
	SMALLINT	-32.768 a 32.767
	INT	-2.147.483.648 a 2.147.483.647
Float	FLOAT	Precisão de número com flutuante de -1,79E+308 + 1.79E+308
	REAL	Precisão de número flutuante de -3,40E + 38 a 3,40E + 38

Equivalente em programação	SQL	Descrição
Tempo (sem tipo de dado específico, é comum usar *strings*)	TIME	Tempo em formato HH:MM:SS
	DATE	Datas em formato AAAA-MM-DD
	DATETIME	Data e tempo em formato AAAA-MM-DD HH:MM:SS
	TIMESTAMP	Armazena um único número que é atualizado sempre que uma linha é criad ou modificada.

Por ora, vamos focar apenas nos tipos de dados básicos necessários para construir nossa primeira tabela em SQL.

Dessa forma, vamos definir um tipo de dado SQL para cada um dos tipos de dados primitivos que usamos em programação: string, number, float e boolean.

String, usaremos `varchar` que nos permite definir um número máximo de caracteres.

Number, usaremos `int`.

Float, usaremos `float`.

Boolean, nem todas as linguagens de consulta têm um tipo de dado específico correspondente. Por isso é comum utilizar `bit`, onde `0 é falso` e `1 verdadeiro`

> SQL Server é um SGBD que não possui um tipo de dado específico para booleanos. Por outro lado, PostgreSQL tem um tipo de dado específico boolean. É ele que iremos utilizar.

7.3.2.3 - Conclusão

Dessa forma, conseguimos perceber que, em SQL, podemos utilizar alguns tipos de dados diferentes dos que utilizamos nas linguagens de programação.

É **importante** se atentar a isso, pois **os tipos de dados podem se diferenciar de acordo com cada Sistema de Gerenciamento de Banco de Dados**.

> Material Complementar

[PANKAJ. SQL Data Types. Digital Ocean, 15 dez. 2022.]https://www.digitalocean.com/community/tutorials/sql-data-types

> Referência Bibliográfica

[CREATESE. 2 Ferramentas de SQL online | CreateSe. YouTube, 11 maio 2021]https://www.youtube.com/watch?v=mVN2dZqeOnw&ab_channel=CreateSe

[TIPOS de dados no SQL. RLSystem, [s.d.].]https://www.rlsystem.com.br/tipos-dados-sql-server

7.3.3 - CRIAR TABELA SQL

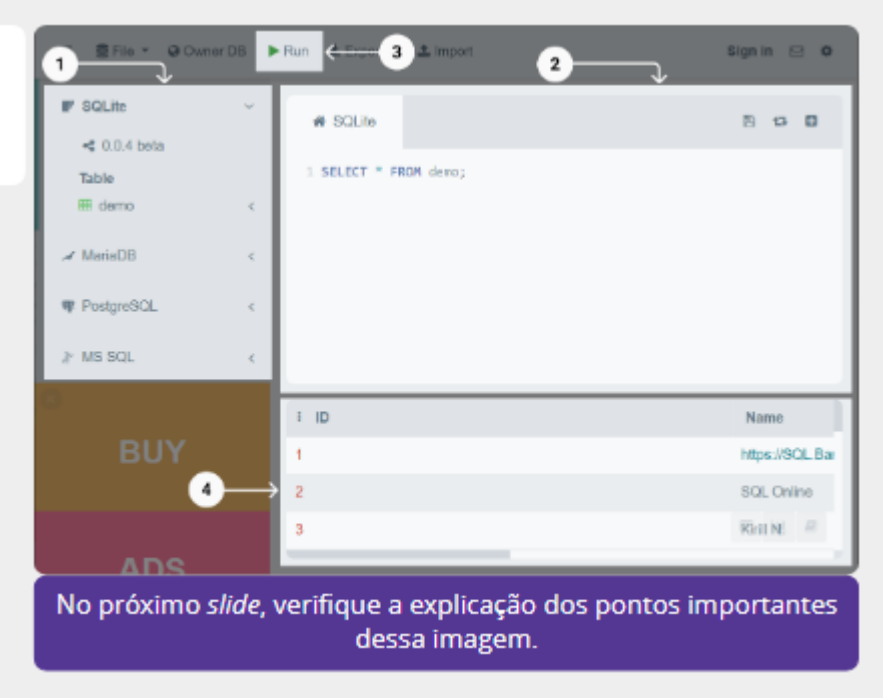
Utilização de ferramentas online para aprendizado em SGBDs.

[Programiz]https://www.programiz.com/sql/online-compiler/

[SQLite]https://sqliteonline.com/

> Apesar de ter algumas limitações em relação aos SGBDs reais, essas ferramentas disponibilizadas online são claras, práticas e ótimas para praticar.

Aberto o SQLite



1. Barra lateral com a lista dos SGBDs disponíveis.
2. Área onde escrevemos as consultas SQL.
3. Botão para executar consultas.
4. Área onde são exibidos os resultados das nossas consultas.

Após realizar nossa primeira consulta, teremos uma barra mostrando o histórico das consultas.

Para configurar o sistema, selecionamos a opção **PostgreSQL** na barra lateral e, em seguida, clicamos no botão `Click to connect` para iniciar a conexão com o servidor.

Na área para escrever consultas, deixamos a estrutura pronta para criar uma tabela. Assim, usamos os comandos create e table, o nome da tabela que queremos criar e um par de parênteses:

Entre os parênteses, definimos os atributos, ou colunas, que estarão na nossa tabela.

Para fazer isso, escrevemos o nome da coluna e, depois, o nome do tipo do dado. No nosso exemplo, começamos definindo uma coluna texto, com o tipo de dado VARCHAR(8).

O número oito entre parênteses representa a quantidade máxima de caracteres que esse campo de texto poderá ter.

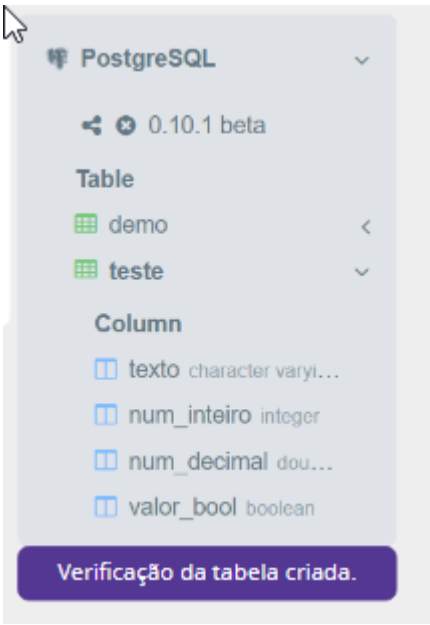
Para definir a próxima coluna, devemos colocar uma vírgula no final da declaração da coluna anterior. Porém, vale destacar que a última definição não deve terminar em vírgula.

No nosso caso, colocamos a vírgula após VARCHAR(8) e, em seguida, definimos cada coluna com seu respectivo tipo de dado. Observe:

```sql
CREATE TABLE teste (
  texto VARCHAR(8),
  num_inteiro INT,
  num_decimal FLOAT,
  valor_bool BOOLEAN
)
```

Para verificar se a tabela foi criada com sucesso, você deve observar a barra lateral, logo abaixo da tabela demo, apresentada na imagem.

Se clicarmos na seta à direita do nome da tabela, veremos os nomes das colunas com seus respectivos tipos de dados.



DADOS OBRIGATÓRIOS

Para entender o que são os dados obrigatórios, vamos pensar em uma situação.

Imagine o banco de dados de um pet shop, onde todo animal de estimação precisa ser associado a um tutor. Portanto, as informações desse tutor, como nome, telefone e e-mail, precisam ser obrigatórias.

Às vezes, os clientes também querem registrar informações de um segundo tutor. Por isso, para diferenciar os valores obrigatórios dos valores opcionais na tabela, devemos levar em consideração a propriedade `null`.

Por padrão, quando criamos uma tabela SQL, *pressupomos que todas as colunas são opcionais*. Assim, se quisermos definir que uma *coluna tenha valor obrigatório*, devemos definir a propriedade `not null`. Dessa forma, não podemos inserir um registro na nossa tabela se o valor da coluna estiver vazio.

Vamos definir uma nova tabela, com o nome pessoa para aplicar esse e os próximos conceitos.

Definimos uma coluna chamada nome, com o tipo de dado VARCHAR(30) e a opção not null. Com isso, estabelecemos que nenhum registro poderá ser adicionado à tabela pessoa sem incluir um valor para a coluna nome.

Na maioria dos SGBDs, temos duas opções para definir a primary key, ou a chave primária, da nossa tabela.

Se tivermos uma chave primária simples, podemos defini-la usando o comando primary key na própria definição da coluna.

Se tivermos uma chave primária composta, precisamos declará-la após a definição das colunas. Para isso, usamos o comando primary key com um par de parênteses e os nomes das colunas que compõem a chave primária composta. Esses nomes devem estar separados por vírgulas. Observe a imagem.

No nosso caso, usaremos uma chave primária simples e definiremos uma nova coluna chamada pessoa_id. Essa terá o tipo de dado int e o comando primary key.

Autoincremento

É muito comum usarmos números como a chave primária dos nossos registros. Como as chaves primárias não podem se repetir em uma tabela, verificar todos os dados utilizados para achar um valor disponível pode ser um processo demorado e ineficiente.

Por isso, também é muito comum definir essas colunas com a opção de autoincremento. Assim, não precisamos nos preocupar em estabelecer o valor da coluna, pois o SGBD define os números automaticamente em ordem crescente, atribuindo sempre o próximo valor para cada novo registro.

Além de definir o valor, ao usar o autoincremento, o SGBD define o tipo de dado int e a opção not null na coluna.

A propriedade de autoincremento varia de sintaxe em cada SGBD. Às vezes, podendo usar comandos como auto_increment ou o identity().

No nosso caso, por estarmos com o PostgreSQL, usaremos o comando serial na definição da coluna pessoa_id. Como essa funcionalidade só pode ser usada com dados numéricos, podemos substituir o tipo de dado int pelo comando serial:

```
1 CREATE TABLE pessoa (
2   pessoa_id SERIAL PRIMARY KEY,
3   nome VARCHAR(8) Not NULL
4 )
```

Finalmente, devemos apertar o botão Run para executar nosso código e criar a tabela pessoa.

Para verificar o funcionamento do autoincremento, basta inserir um valor na coluna nome e realizar um select na tabela. Dessa forma, veremos o registro criado com o nome e o valor que foi atribuído automaticamente.

Inserindo e buscando dados na tabela

Para inserir o nome de uma pessoa na tabela, execute os seguintes comandos no ambiente SQL:

```
INSERT INTO pessoa (nome) VALUES ('João');
```

Já para selecionar a tabela pessoa e verificar seus dados:

```
SELECT * FROM pessoa;
```

Assim, no nosso exemplo, após inserir o dado na tabela, a busca encontrará o nome João seguido do id atribuído automaticamente.

Conclusão

Muito bem! Agora você já consegue verificar os primeiros passos para a utilização de um SGBD, onde podemos criar tabelas, inserir, buscar e manipular dados. Não deixe de continuar praticando para aprimorar suas habilidades na construção de tabelas.

> Material Complementar

[PANKAJ. SQL Data Types. Digital Ocean, 15 dez. 2022.]https://www.digitalocean.com/community/tutorials/sql-data-types

> Referência Bibliográfica

[CREATESE. 2 Ferramentas de SQL online | CreateSe. YouTube, 11 maio 2021.]https://www.youtube.com/watch?v=mVN2dZqeOnw&ab_channel=CreateSe

[TIPOS de dados no SQL. RLSystem,]https://www.rlsystem.com.br/tipos-dados-sql-server

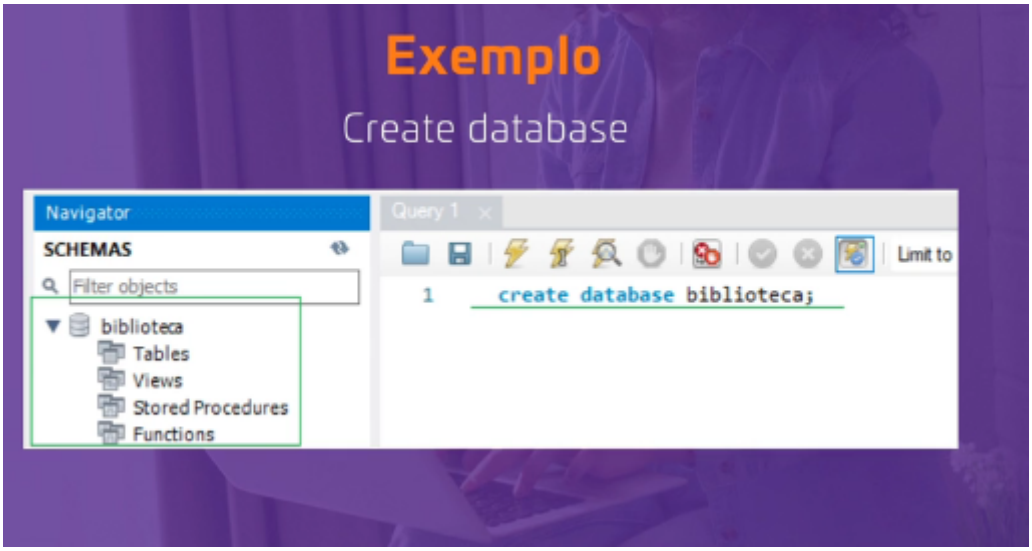
7.3.4 - IMPLEMENTAR UM BANCO DE DADOS EM SQL

Etapas necessárias para Criar, exibir e excluir.

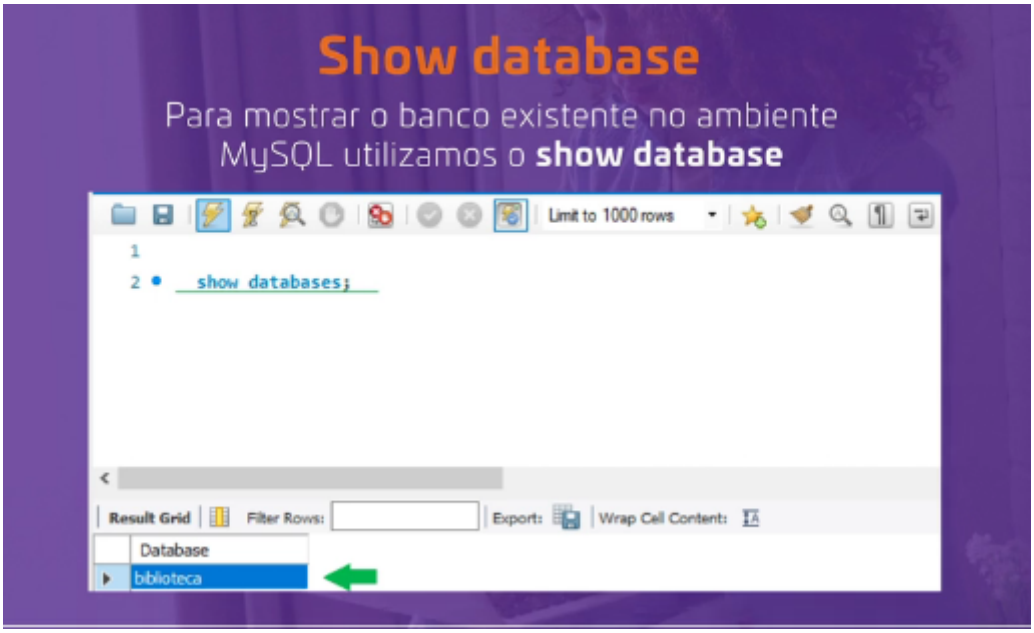
Para implementar é preciso cumprir etapas essenciais:

1. análise de requisitos
    1. realizado entrevistas e a coleta de dados. Apenas dados relacionados as regras de negócio.
2. início do projeto conceitual
    1. É realizada a projeção por meio de conceitos simples fazendo uso de diagramas.
3. Projeto Lógico.

    1. Criado os modelos internos do BD, setando as tabelas, regras, relacinoamento e dados das colunas, como tamanho e tipo. Tem **como resultado o esuqema detalhado do banco de dados**.
4. Projeto Físico.
    1. Localizado na parte mais baixa em relação ao cliente, definindo a estruturas de armazenamentos e apresentada a forma como os dados estão armazenados. Além de gerar os scripts que criam os objetos como tabelas.
5. Ferramentas para criar o banco de dados
    1. MySQL + MySQL Workbench
        1. MySQL Workbench, permite criar gráficos de Relação-entidade para bancos MySQL. Linguagem será ANSI SQL.
6. Criar tabela
    1. create database "nome do banco"

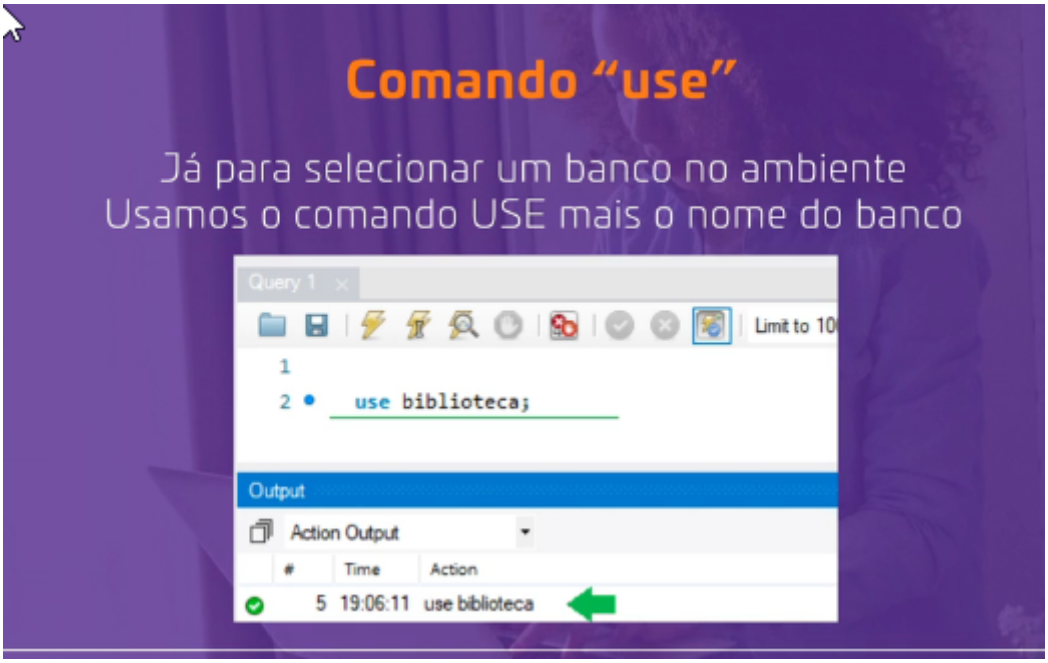


7. Mostrar tabela criada
    1. show databases;

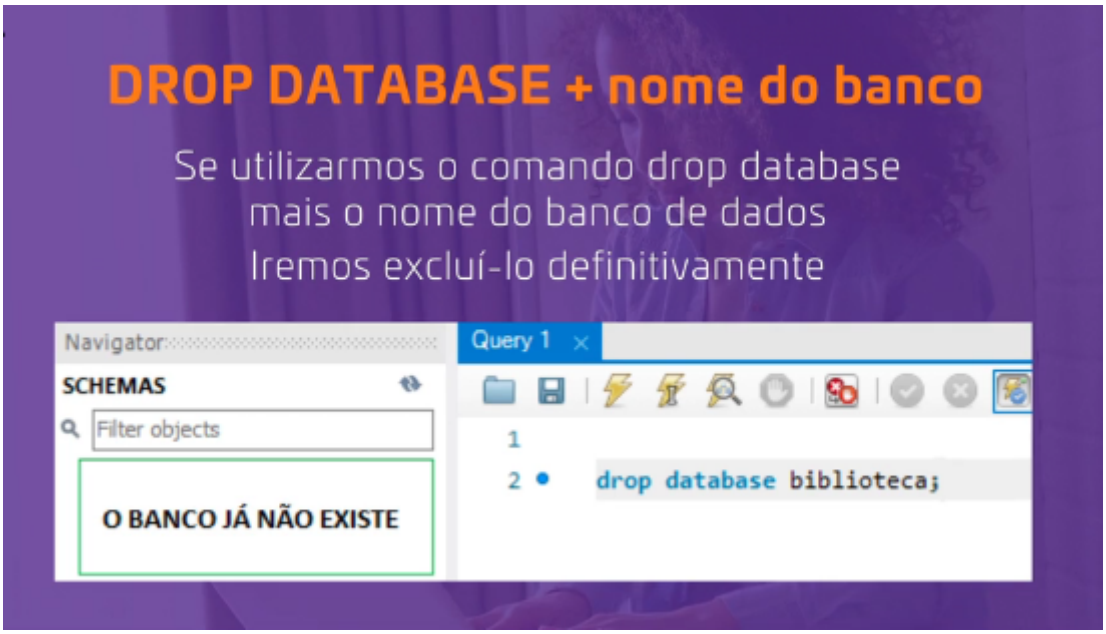


8. Selecionar um banco no ambiente

1. use "nome do banco"



9. Excluir tabela
    1. drop database "nome da tabela"



7.3.5 - SQL - LINGUAGEM ESTRUTURADA DE CONSULTAS

SQL é uma `linguagem estruturada de consultas` **e a mais utilizada** pelos SGBDs relacionais. Ela pode realizar consultas simples ou complexas e permite aos usuários a criação das bases de dados e suas estruturas relacionais correspondentes. Lembrando que uma base de dados pode ser administrada com a criação, eliminação e alteração dos dados.

[SQL]https://www.w3schools.com/sql/

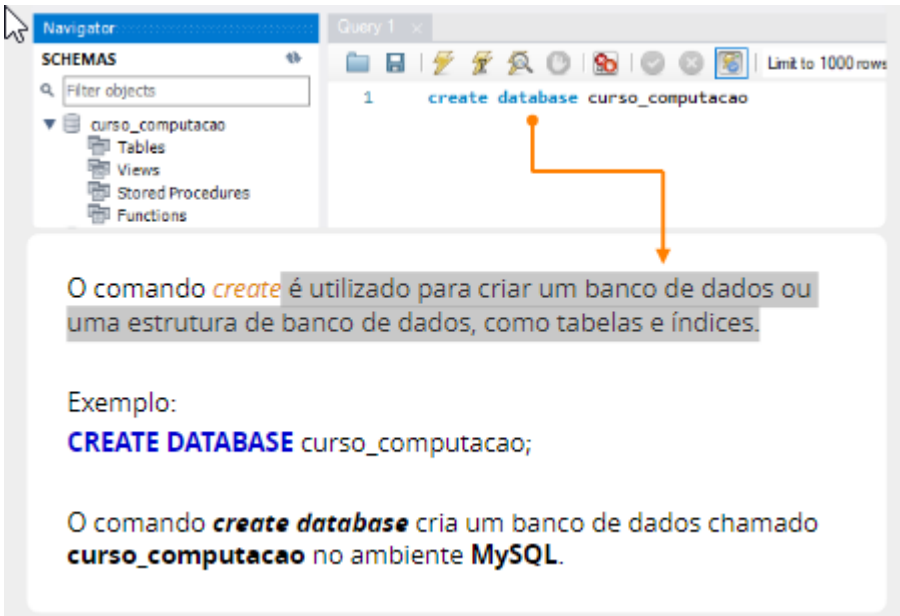
A SQL foi projetada pela IBM nos anos 1970. No final dessa década, o primeiro banco de dados baseado nessa linguagem foi disponibilizado comercialmente.

Em 1980, os institutos ANSI e ISO publicaram a sua primeira versão padronizada. Além disso, há evoluções dessa linguagem, como a versão SQL-92 e a SQL-99.

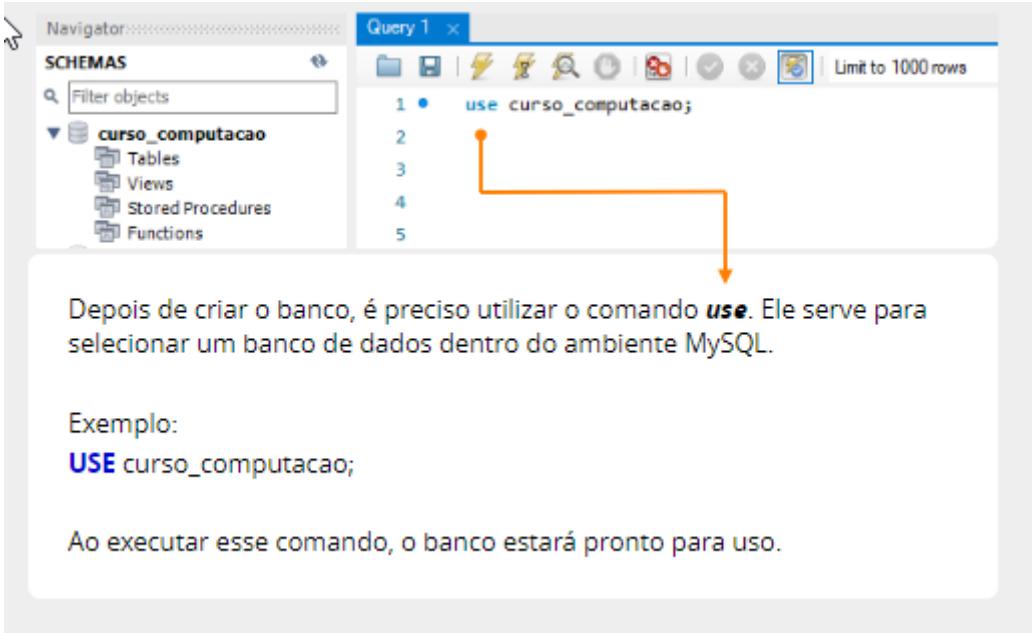
A **SQL é um conjunto de linguagens**, onde cada uma possui comandos específicos para executar diferentes funções em um banco de dados. As cinco linguagens fundamentais, os subconjuntos da SQL, que serão abordadas são: **DDL**, **DML**, **DQL**, **DCL** e **DTL**.

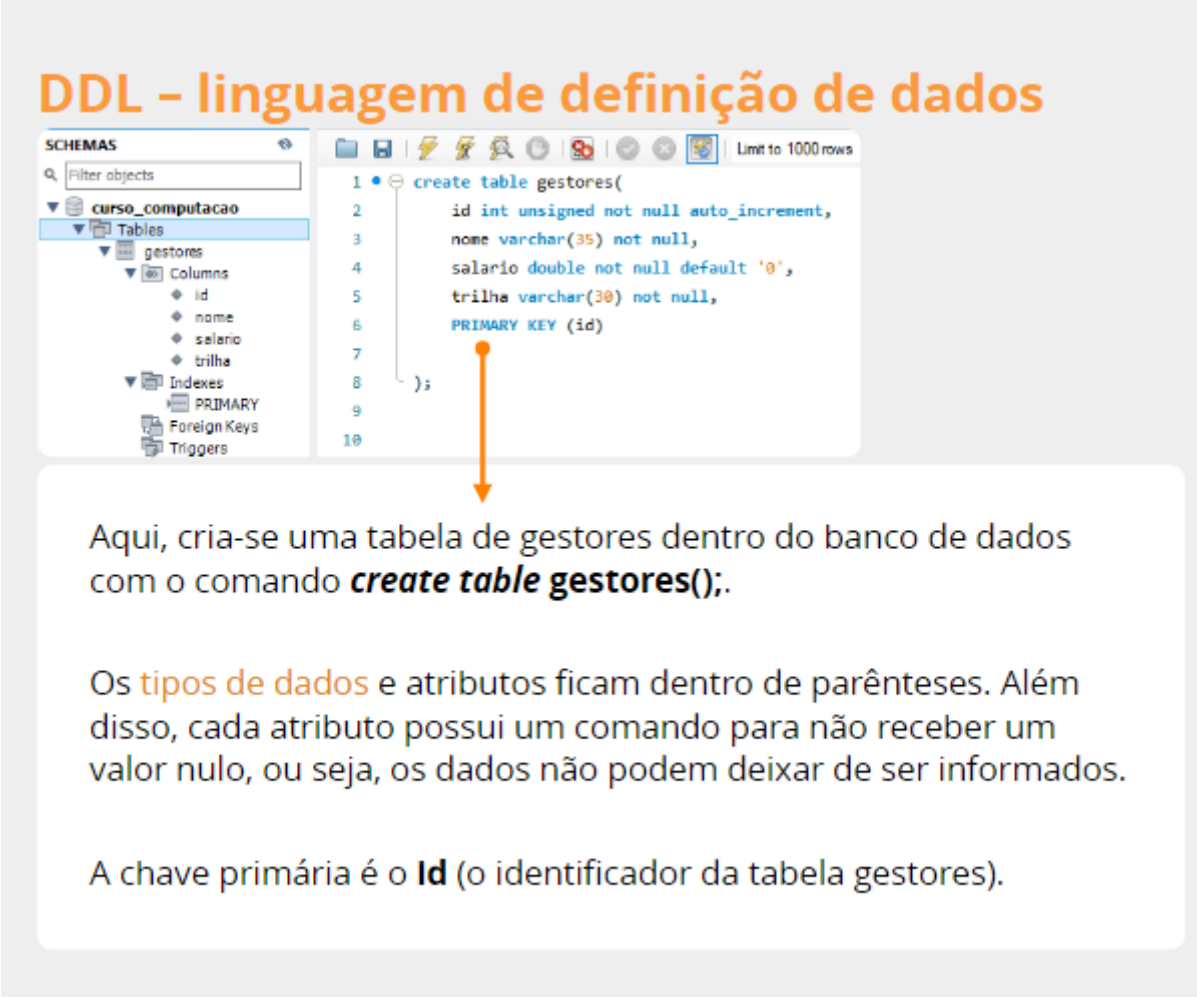
DDL é a responsável pela **definição dos dados**. Seus principais comandos são: `create`, `alter` e `drop`;

- **create**, é utilizado para criar um banco de dados ou uma estrutura de banco de dados, como tabelas e índices.



Depois de criar o banco, é preciso utilizar o comando use. Ele serve para selecionar um banco de dados dentro do ambiente MySQL.

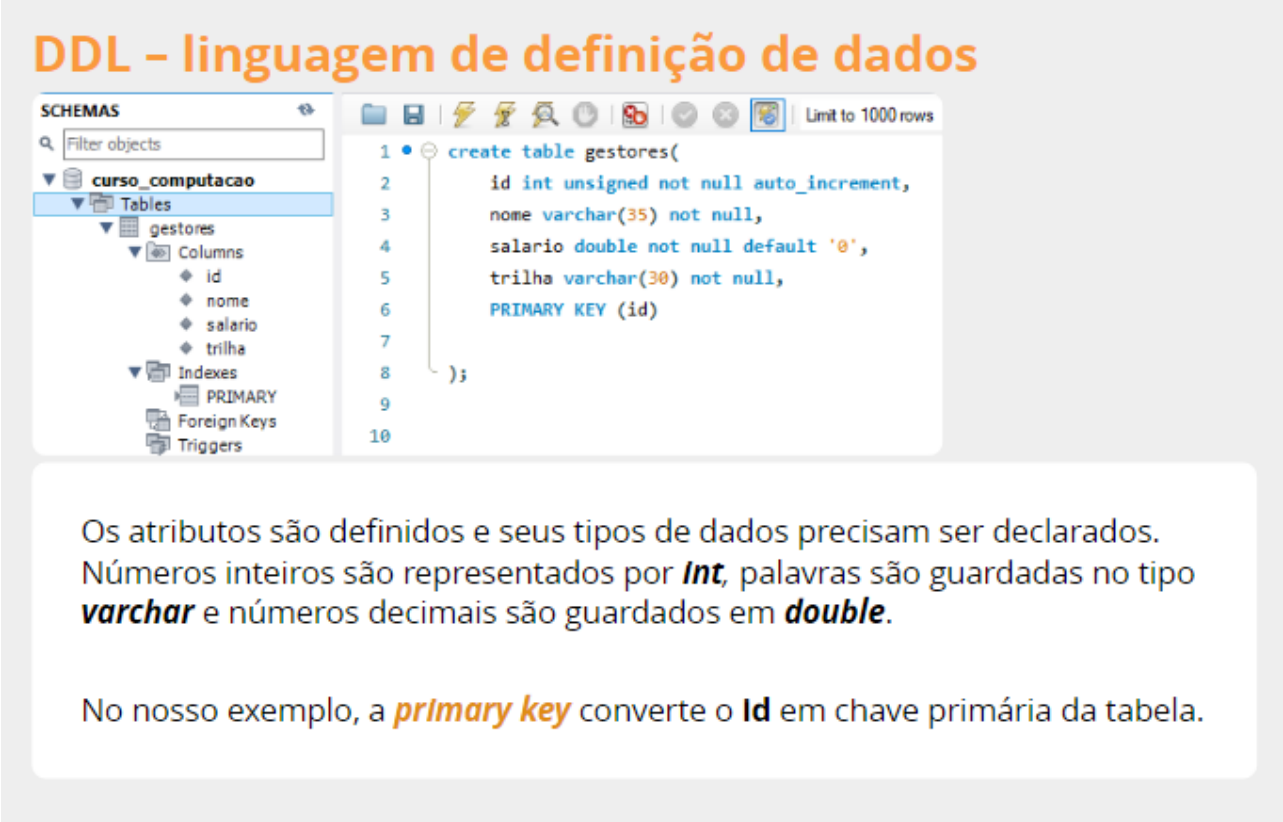




Aqui, cria-se uma tabela de gestores dentro do banco de dados com o comando create table gestores();.

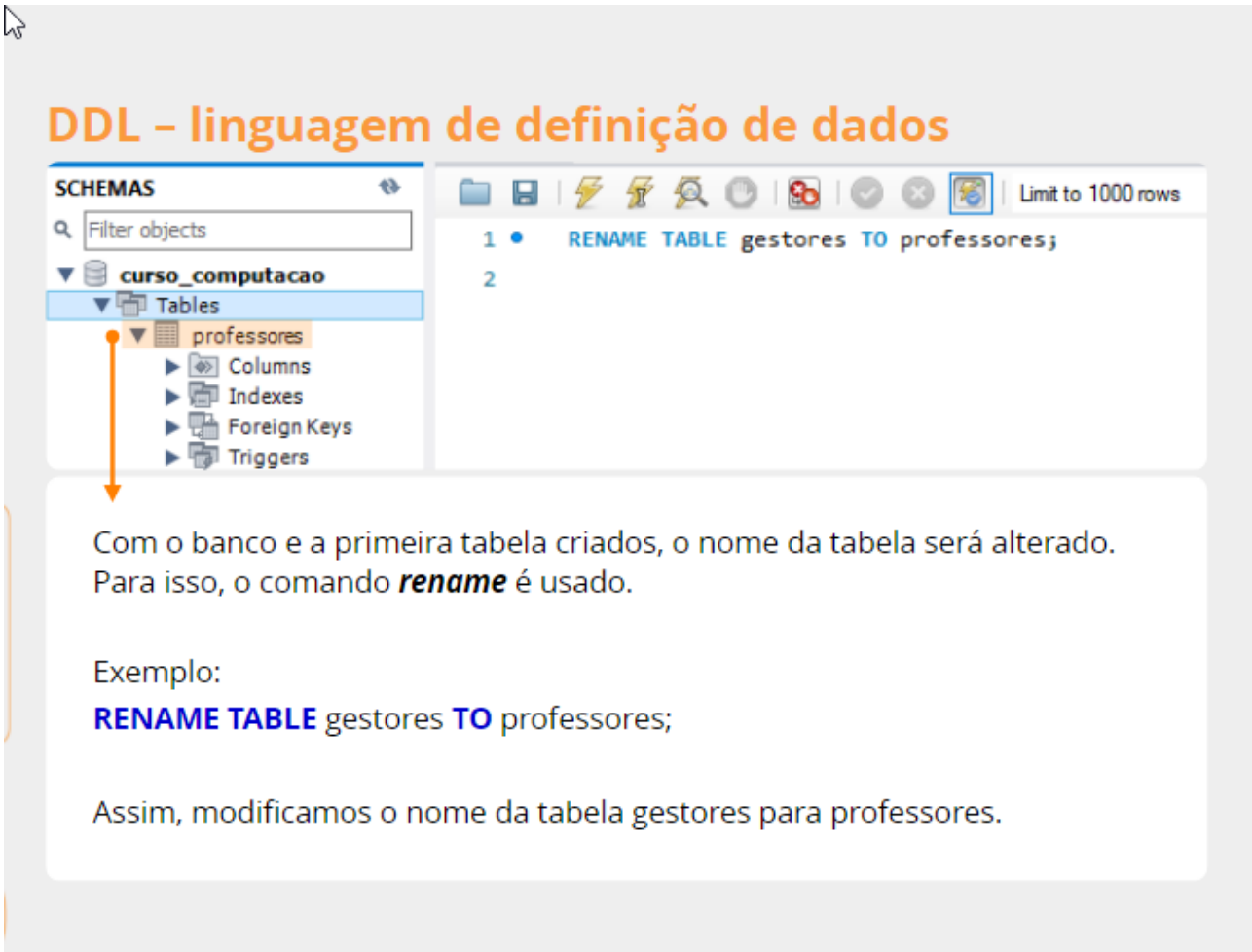
Os tipos de dados e atributos ficam dentro de parênteses. Além disso, cada atributo possui um comando para não receber um valor nulo, ou seja, os dados não podem deixar de ser informados.

A chave primária é o id (o identificador da tabela gestores).



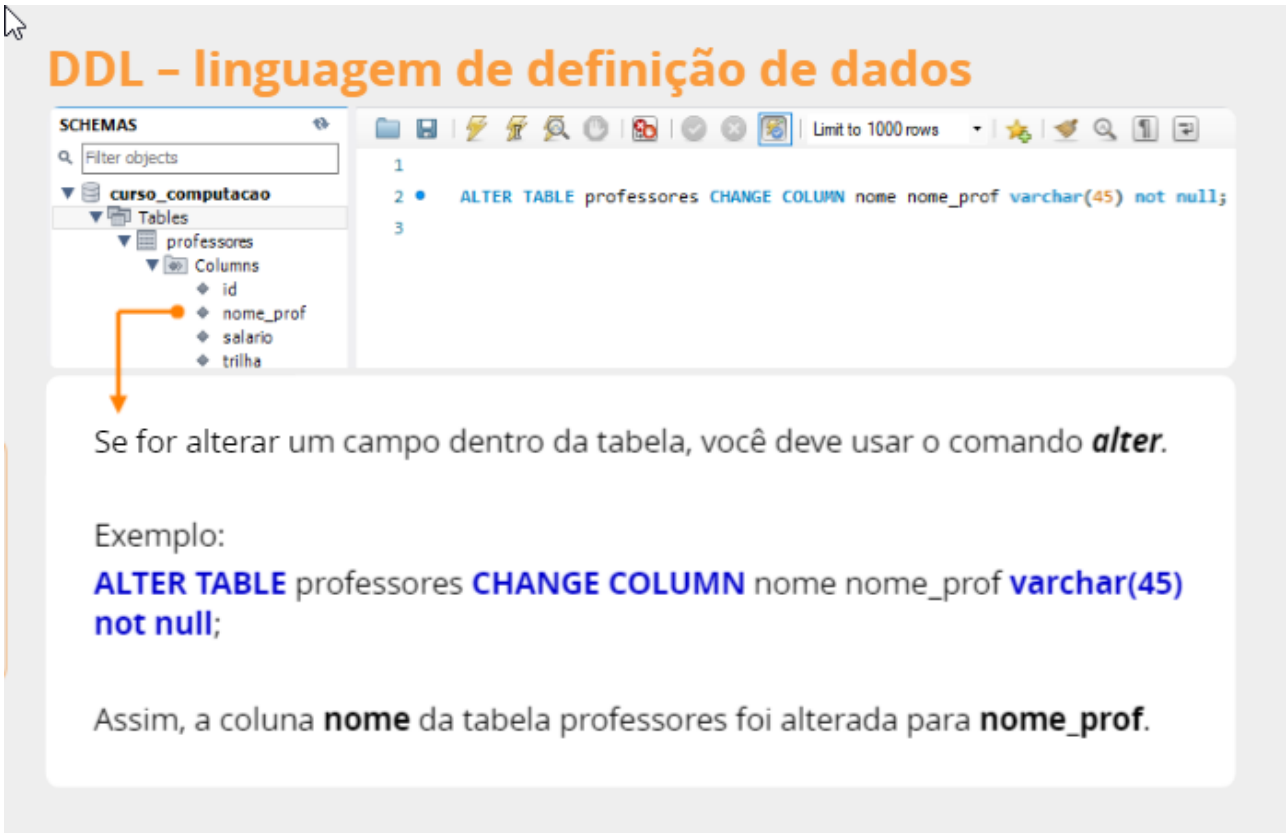
Os atributos são definidos e seus tipos de dados precisam ser declarados. Números inteiros são representados por int, palavras são guardadas no tipo varchar e números decimais são guardados em double.

No nosso exemplo, a primary key converte o id em chave primária da tabela.



Com o banco e a primeira tabela criados, o nome da tabela será alterado. Para isso, o comando rename é usado.

Assim, modificamos o nome da tabela gestores para professores.



Se for alterar um campo dentro da tabela, você deve usar o comando alter.

DDL – linguagem de definição de dados

Porém, se for preciso excluir um campo da tabela do banco de dados, utilizamos o comando ***drop***.

Exemplo:

ALTER TABLE nome_da_tabela DROP COLUMN nome_da_coluna.

Com esse comando, também é possível excluir definitivamente uma tabela do banco.

Exemplo:

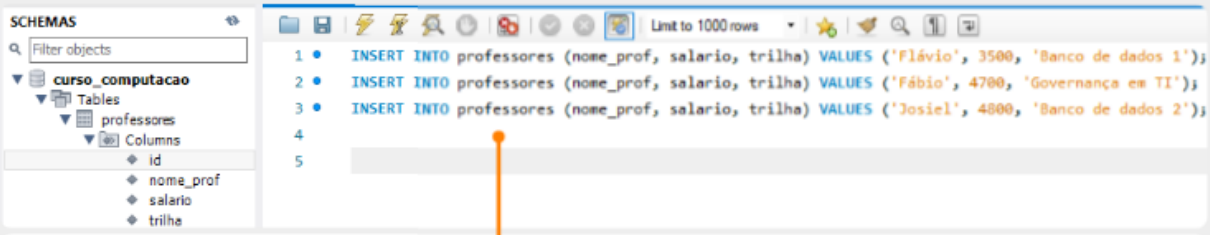
DROP TABLE nome_da_tabela;

DDL – linguagem de definição de dados

O comando ***Insert*** é utilizado para inserir dados nessa tabela. Nos próximos exemplos, serão inseridos: o código de identificação do professor, o nome dele, o seu salário e a aula (trilha) que ministra. Observe:

INSERT INTO professores (id, nome_prof, salario, trilha) **VALUES** (1, 'Kelvin', 4000, 'Banco de dados distribuídos');
INSERT INTO professores (id, nome_prof, salario, trilha) **VALUES** (2, 'Johny', 3400, 'Java1');

DDL – linguagem de definição de dados



Mais exemplos:

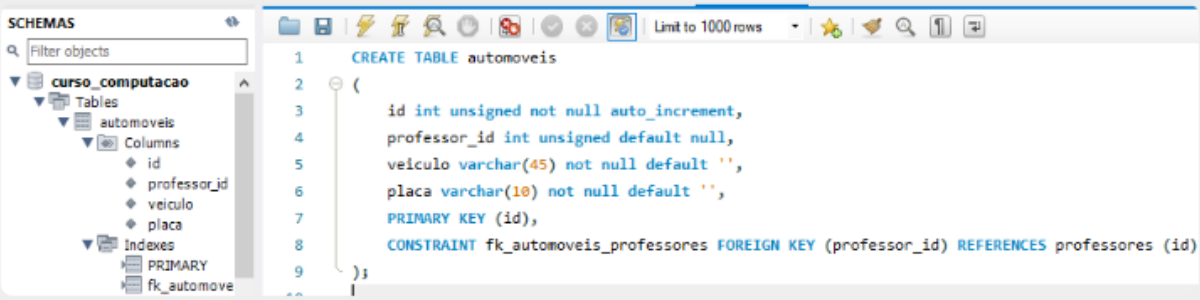
INSERT INTO professores (nome_prof, salario, trilha) **VALUES** ('Flávio', 3500, 'Banco de dados 1');

INSERT INTO professores (nome_prof, salario, trilha) **VALUES** ('Fábio', 4700, 'Governança em TI');

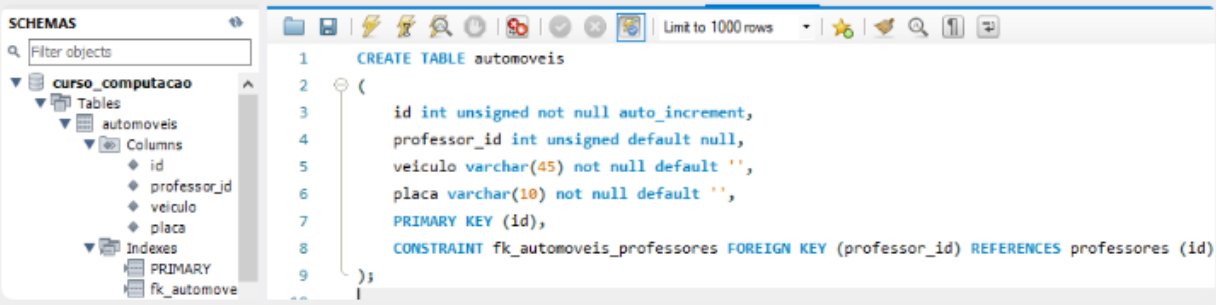
INSERT INTO professores (nome_prof, salario, trilha) **VALUES** ('Josiel', 4800, 'Banco de dados 2');

Criando nova tabela no banco

Para incrementar o banco do nosso exemplo, será inserida uma tabela **automóveis**, que deixará registrado os dados dos carros dos professores:



Criando nova tabela no banco



- A chave primária da tabela será o identificador;
- Teremos um código **id** (identificador) para cada professor;
- O modelo ou marca do automóvel deve ser definido com até 45 caracteres;
- A placa do automóvel deve ser definido com até dez caracteres;
- A tabela possuirá uma chave estrangeira, que é o **id da tabela professores** associada ao **professor_id** dessa tabela, para fazer referência a qual professor o carro pertence.

DML é a responsável pela **manipulação dos dados**. Seus principais comandos são: `insert`, `update` e `delete`;

Comando *delete*

Enquanto o comando ***drop***, que vimos anteriormente, exclui estruturas de um banco de dados, o ***delete*** elimina apenas os dados e registros das tabelas.

Para excluir os dados de um professor com **Id** = 4, observe o exemplo:
DELETE FROM professores WHERE **Id** = 4;

id	nome	salario	trilha
1	Kelvin	4000	Banco de dados distribuídos
2	Johny	3400	Java 1
3	Flávio	3500	Banco de dados 1
5	Josiel	4800	Banco de dados 2

Assim, os dados do professor com **Id** = 4 não aparecem mais na tabela.

Comando *update*

Já para atualizar uma informação, é preciso usar o comando ***update***. Por exemplo, para o salário do professor de **Id** = 1 receber um aumento de 10%:

```
UPDATE professores SET salario = salario * 1.1 WHERE id = 1;
```

Nesse caso, o salário do professor Kelvin aumentará.

id	nome	salario	trilha
1	Kelvin	4000	Banco de dados distribuídos
2	Johny	3400	Java 1
3	Flávio	3500	Banco de dados 1
4	Fábio	4700	Governança em TI
5	Josiel	4800	Banco de dados 2

Comando *update*

O salário do professor Kelvin foi atualizado e ele passou a ganhar R$4.400, quando antes recebia R$4.000.

id	nome	salario	trilha
1	Kelvin	4400	Banco de dados distribuídos

Inserindo dados na tabela automóveis

Agora, iremos inserir dados na tabela de automóveis.

```
INSERT INTO automoveis (professor_id, veiculo, placa) VALUES (1, 'Carro', 'KLP-1234');
INSERT INTO automoveis (professor_id, veiculo, placa) VALUES (1, 'Carro', 'KLB-4567');
SELECT * FROM automoveis ;
```

id	professor_id	automovel	placa
1	1	Carro	KLP-1234
2	1	Carro	KLB-4567

Observe que foi inserido dois carros para o professor com **id** = 1.

Atualizando dados da tabela automóveis

Para atualizar essa informação contida na tabela, utilizamos o comando ***update***.

```
UPDATE automoveis SET professor_id = 5 WHERE id = 2;
SELECT * FROM automoveis ;
```

Assim, o segundo veículo do professor **id** = 1 foi mudado para o professor do **id** = 5.

id	professor_id	automovel	placa
1	1	Carro	KLP-1234
2	5	Carro	KLB-4567

DQL é a linguagem de consulta de dados. Seu principal comando é o `select`;

- **select**, Para projetar na tela os dados inseridos nos campos de uma tabela, utilizamos o select. Ele é o comando mais utilizado na criação de um banco de dados e pertence à DQL, a linguagem de consultas de dados. Por isso, para exemplificar esse comando, uma tabela será apresentada com o nome de cinco professores, os seus salários, os seus códigos de identificação (ids) e as suas trilhas.

Comando *select*

Exemplo:

SELECT * FROM professores;

id	nome	salario	trilha
1	Kelvin	4000	Banco de dados distribuídos
2	Johny	3400	Java 1
3	Flávio	3500	Banco de dados 1
4	Fábio	4700	Governança em TI
5	Josiel	4800	Banco de dados 2

Todos os dados dos professores inseridos na tabela são apresentados com o mesmo comando.

Comando SELECT

É possível pesquisar informações específicas da tabela, utilizando o comando **SELECT*FROM** com o comando ***where***.

Exemplo:

SELECT * FROM professores WHERE salario > 4000;

Dessa forma, apenas os dados dos professores que têm salário acima de R$4.000 serão exibidos.

id	nome	salario	trilha
4	Fábio	4700	Governança em TI
5	Josiel	4800	Banco de dados 2

Comando SELECT

Outro exemplo:

SELECT * FROM professores WHERE nome_prof = 'Josiel';

Aqui, só serão exibidos os dados do professor Josiel.

id	nome	salario	trilha
5	Josiel	4800	Banco de dados 2

Exemplo:

SELECT * FROM professores WHERE id = 3;

Assim, serão exibidos os dados do professor que tem o **Id** = 3, o professor Flávio.

id	nome	salario	trilha
3	Flávio	3500	Banco de dados 1

DCL é a responsável pelo controle dos dados. Seus principais comandos são: `grant` e `revoke`;

DTL é a linguagem de transação de dados. Seus principais comandos são: start transaction, `commit` e `rollback`.

UNION

Comando *union*

Para unir os dados das tabelas, podemos utilizar o comando ***union***.
Exemplo:

```
SELECT * FROM professores WHERE nome_prof = 'Kelvin'
UNION
SELECT * FROM automoveis WHERE id = 1;
```

id	nome_prof	salario	trilha
1	Kelvin	4400	Banco de dados distribuídos
1	1	Carro	KLP-1234

Nessa união, temos os dados do professor Kelvin e do carro que ele possui. Assim, com esse comando, os dados que são armazenados em tabelas diferentes podem ser apresentados juntos.

Usando os subconjuntos da linguagem SQL, conseguimos criar um banco de dados, inserir tabelas, adicionar dados e fazer diversas outras manipulações. Dessa forma, podemos compreender como funciona o banco de dados no mundo real.

> leitura Complementar

[HCODE. MySQL 8 - Instalando e Configurando. YouTube, 3 dez. 2018.]https://www.youtube.com/watch?v=KYaZVqHHXpM
[SILBERSCHATZ, Abraham, KORTH, Henry F., SUDARSHAN, S. Sistema de Banco de Dados. 5 ed. Rio de Janeiro: Campus, 2006.]
[SQL Tutorial. W3Schools, [s/d]. Disponível em]https://www.w3schools.com/sql/

> Referência Bibliográfica

DATE, Christopher J. Introdução a Sistemas de Bancos de Dados. Rio de Janeiro: Campus, 2004.

ELMASRI, Ramez E., NAVATHE, Shamkant B. Sistemas de Banco de Dados. 4 ed. São Paulo: Pearson Addison-Wesley, 2008.

OPPEL, Andy. Banco de Dados Desmistificado. Rio de Janeiro: Alta Books, 2004.

7.3.6 - Anotações Exercícios

1. Aluno (Matricula..., Endereco..., Telefone..., Email..., Nome..., PRIMARY KEY(Matricula)); Quanto à inserção de dados nessa tabela, assinale a alternativa correta.
    1. INSERT INTO ALUNO (MATRICULA, NOME, ENDERECO, CONTATO, EMAIL) VALUES (1254, 'Joaquim', 'Rua 13 de Maio 720', '99932-5466', 'joaquim@gmail.com.br');

        1. Resposta correta!A INSERT INTO é a instrução usada para inserir novos registros em uma tabela.
2. Qual comando é utilizado para excluir dados inseridos em uma tabela de um banco de dados?
    1. O comando DELETE.
        1. Resposta correta!O comando DELETE é usado para excluir registros existentes em uma tabela. Exemplo: DELETE FROM Aluno WHERE matricula = 1254;
3. Aluno (Matricula..., Endereco..., Telefone..., Email..., Nome..., PRIMARY KEY(Matricula)); Qual comando é necessário para exibir todas as informações da tabela aluno?
    1. SELECT * FROM aluno;
        1. Resposta correta!SELECT é uma instrução usada para selecionar dados de um banco de dados. Os retornados são armazenados em uma tabela de resultados chamada de conjunto de resultados.
4. UPDATE ALUNO SET ENDERECO = 'Av. Brasil 778' WHERE CODIGO = 1; O que significa essa sintaxe?
    1. O endereço do aluno que possui o código = 1, na tabela aluno, será alterado.
        1. Resposta correta!UPDATE é a instrução usada para modificar os registros existentes em uma tabela.
5. CREATE TABLE livros(); é uma sintaxe usada para criar o quê?
    1. Uma tabela chamada livros.
        1. Resposta correta!Alem disso, os objetos, tipos e atributos ficam dentro dos parênteses.

7.3.7 - Estágio FINAL

1. Sobre a operação SELECT na álgebra relacional, assinale a alternativa correta.
    1. A operação SELECT seleciona tuplas das tabelas para atender uma requisição para extrair informação.
        1. ?
2. Assinale a alternativa que possui exemplos de Sistemas de Gerenciamentos de Banco de Dados relacionais (SGBDs)?
    1. ORACLE e MySQL.
        1. ?
3. Sobre os modelos de dados, assinale a alternativa correta.
    1. Segundo Silberschatz (2006), um modelo de dados é uma coleção de ferramentas conceituais para descrever dados, suas relações, as semânticas e as restrições de consistência.
        1. ?
4. Sobre as fases de implementação de um projeto, assinale a alternativa correta.
    1. O projeto conceitual se baseia nas informações coletadas pela análise de requisitos e, a partir delas, um esquema conceitual é gerado, utilizando um modelo de dados conceitual de alto nível, o MER (Modelo Entidade-Relacionamento).
        1. ?
5. Sobre o processo de normalização de dados, assinale a alternativa correta.
    1. A normalização de dados foi criada em 1970, sendo reconhecida posteriormente como formas normais. Essas formas descrevem o que deve ser feito para organizar os dados no banco de dados.
        1. ?
6. Sobre a abstração de dados, assinale a alternativa correta.
    1. A abstração de dados garante uma visão abstrata do banco de dados para quem vai utilizá-lo. Isso acontece porque não é importante para o usuário conhecer como esses dados estão

armazenados, mas sim a disponibilidade deles.

1. ?

7. Sobre os bancos de dados, assinale a alternativa correta.
    1. Um banco de dados é um conjunto de dados inter-relacionados.
        1. ?
8. Qual é a função da instrução CREATE DATABASE?
    1. Criar um banco de dados.
        1. ?
9. Sobre a SQL, assinale a alternativa correta.
    1. A SQL é uma linguagem estruturada de consultas.
10. Quais desses comandos fazem parte da Linguagem de Definição de Dados (DDL)?
    1. CREATE, ALTER E DROP.
        1. ?

10 ACERTOS