# **Banco de Dados**

Um banco de dados é uma coleção organizada de dados estruturados, normalmente armazenados eletronicamente em um sistema de computador.

Um banco de dados é geralmente controlado por um SGBD, Sistema Gerenciador de Banco de Dados. Então, os dados e o SGBD, juntamente com os aplicativos associados a eles, são chamados de sistema de banco de dados, geralmente abreviados para apenas banco de dados.

As empresas, hoje conectadas com todo o mundo, estão conectadas e manipulam base de dados universal, ou seja, banco de dados que obedeçam aos padrões estabelecidos pela SQL – Structure Query Language, pois o mundo é relacional.

Os dados nos tipos mais comuns de bancos de dados em operação atualmente são modelados em linhas e colunas em uma série de tabelas para tornar o processamento e a consulta de dados eficientes, a partir de base de dados universal.

Logo, se o mundo é relacional, está ligado a base de dados universal, ou seja, manipulação de banco de dados SQL.,

Podemos definir um banco de dados como um conjunto de dados que se relacionam.

Propriedades de um banco de dados:

. Um banco de dados é uma coleção lógica de dados com um significado inerente;

. Uma disposição desordenada de dados não pode ser um banco de dados;

. Um banco de dados é projetado, construído e populado com dados para um propósito específico.

. Um banco de dados possui um conjunto pré-definido de usuários e aplicações.

. Um banco de dados representa algum aspecto do mundo real, o qual é chamado de “minimundo” e qualquer alteração efetuada neste minimundo é automaticamente refletida no banco de dados.

Um SGBD é uma coleção de programas que permite a seus usuários criar e manipular bancos de dados. O conjunto formado por um BD e pelos programas que o manipulam é chamado de SGBD.

Metadados: São informações que contém a estrutura de cada arquivo, o tipo e o formato de armazenamento de cada tipo de dado.

SGBD: Um arquivo (tabela) é definido uma única vez e atende a várias aplicações, ou seja, existe múltipla visão dos dados. Armazena-se junto com os dados todas as informações referentes à forma como estes foram estruturados e onde eles estão armazenados fisicamente.

Abstração de dados: caracteriza-se por uma independência entre programas e dados e entre programas e operações de manipulação de dados. os acessos são escritos no banco de dados e os programas enviam comandos solicitando o acesso aos dados.

É permitido acesso simultâneo de vários usuários ao mesmo dado. Essa simultaneidade é tratada através do gerenciamento da concorrência. No entanto, somente o 1° usuário que acessar terá acesso a fazer as operações de um CRUD. Os demais só poderão ler (read).

**Propriedades de um SGBD**

Independência programa-dados: O banco de dados pode ter sua estrutura alterada sem precisar alterar os programas que o acessam.

Transação: é um processo que inclui um ou mais acessos no banco de dados, como leitura e gravação de registros.

Isolamento: garante que cada transação possa ser efetuada de forma isolada de outras transações.

Atomicidade: garante que todas as operações da transação sejam realizadas ou nenhuma delas seja.

Controle de Redundância: Para manter a consistência do banco de dados, deve-se armazenar o dado uma única vez e em apenas um lugar no banco de dados. Isto permite manter a consistência, economizar espaço de armazenamento.

Compartilhamento de Dados: Permite usuários diferentes a utilização simultânea de um mesmo dado.

Restrição de Acesso: Um SGBD deve prover controles de segurança e autorização, que são utilizados para criar contas e seus respectivos direitos de acesso quando múltiplos usuários compartilham um banco de dados.

Restrição de Acesso: Os controles de segurança abrangem conceitos tais como: procedimentos de validação e controle, garantia de integridade e controle de acesso, que visam resguardar o banco de dados de uma possível perda ou destruição de dados seja por falha de programa ou por falha de equipamento.

Restrição de Integridade: A maioria dos SGBDs proveem certas restrições de integridade que devem ser aplicadas aos dados. O SGBD deve ter mecanismos para possibilitar a definição das restrições e assegurar o respeito a estas.

Mecanismo de Backup e Recuperação: Um SGBD deve prover facilidades para recuperação de falhas do hardware ou software.

Múltiplas Interfaces: Um ambiente de banco de dados é acessado por variados tipos de usuários com variadas necessidades de informação e com diferentes níveis de conhecimento técnico. Para atender esta diversidade usuários, o SGBD deva fornecer diferentes tipos de interfaces.

**Benefícios de um SGDB**

* Potencial para o estabelecimento e o cumprimento de padrões.

• Garantir o uso de padrões na organização, códigos, nomes, datas.

* Flexibilidade de mudanças.

• Alteração da estrutura dos dados de forma simples.

* Redução no tempo de desenvolvimento de novas aplicações.

• Encapsulamento das regras de armazenamento e integrações.

* Disponibilidade de informação atualizada.

• Logo que um dado é atualizado, ele está disponível para os usuários.

Para um projeto de banco de dados, existe em três modelos/categorias distintas:

**- Modelo Conceitual:**

. Linguagem usada para descrever esquemas conceituais independe do SGBD escolhido.;

. Não tem dependência com a classe do SGBD a ser escolhido;

. Produz um esquema conceitual a partir de “requisitos” de um mundo real.

**- Modelo Lógico:**

. Linguagem usada para especificar esquemas lógicos. Pertencem a três classes: **Relacional**, Redes e Hierárquico;

. Tem dependência com a classe, mas não com o SGBD a ser escolhido.

**- Modelo Físico:**

. É a descrição da Implementação da base de dados em memória secundária. Descreve estruturas de armazenamento e métodos de acesso. Tem forte ligação com o SGBD específico. No nosso caso, um SGBD relacional, voltado para base universal, SQL.;

. Total dependência do SGBD específico.

A divisão da modelagem de dados (modelo conceitual, modelo lógico e modelo físico), facilita a correção de erros e a evolução do modelo a partir de novos requisitos de informação.

Banco de Dados Relacional (Universal): Um banco de dados relacional consiste em uma coleção de tabelas, que podem ser relacionadas através de seus atributos, ou seja, uma linha de uma tabela pode estar sendo relacionada com uma outra linha em uma outra tabela.

Modelo Entidade-Relacionamento:

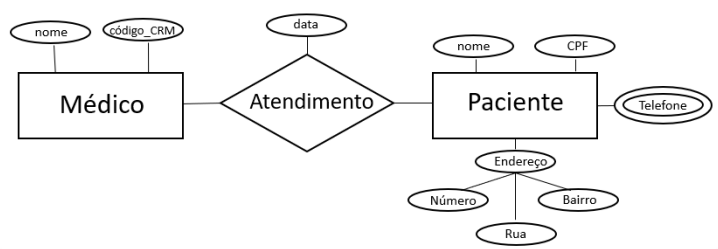
Entidade:

Relacionamento:

Atributo:

Atributo Multivalorado:

Atributo Composto:



Cardinalidade: A cardinalidade ou mapeamento de restrição, especifica o número mínimo e o máximo de instâncias que uma entidade pode participar. Para facilitar o entendimento, observe o diagrama abaixo.



A cardinalidade da entidade está do lado oposto ao seu retângulo, havendo a cardinalidade mínima e a máxima. Para lê-las, adotamos a seguinte estrutura:

ENTIDADE + RELACIONAMENTO + CARDINALIDADE + ENTIDADE

Desse modo, as cardinalidades de cada uma das duas entidades são:

•Um professor pode ministrar 0 (mínimo) ou ‘n’ (máximo) disciplinas.

•Uma disciplina pode ser ministrada por apenas 1 (mínimo) ou 1 (máximo) professor.

Tipos de Cardinalidades Possíveis

1, 1 – Um e apenas um

0, 1 – Nenhuma ou uma

0, N – nenhuma ou várias (muitas)

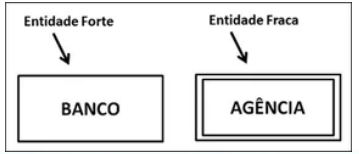
1, N – um ou várias (muitas)

N, N – muitas para muitas

Observação: Quando existe um relacionamento de muitas para muitas, esse relacionamento é denominado entidade-relacionamento e, será representado por uma nova tabela, ou seja, vai ser uma tabela no modelo relacional.

Entidade Forte x Entidade Fraca

As entidades fracas possuem dependência de existência e/ou identificação. São sempre ligadas a outras tabelas através de relacionamentos.



Relacionamento Forte x Relacionamento Fraco

Sabe-se que os relacionamentos fortes existem entre entidades referenciadas como entidades fortes. Já os relacionamentos fracos existem em decorrência de existir o relacionamento entre entidades e, pelo menos uma delas, ser caracterizada como entidade fraca.

Grau dos Relacionamentos:

Unário: Onde uma entidade se relaciona com ela mesma.

Binário: Onde duas entidades participam de um relacionamento.

Ternário: Onde três entidades participam de um relacionamento. Alguns autores sugerem inclusive que esses relacionamentos não sejam adotados.

N-ário: Onde quatro ou mais entidades participam de um relacionamento.

Classificação dos Atributos

Atributo identificador: Representado através de uma bola cheia na extremidade do atributo. Atributos identificadores identificam ou compõe a identificação única de uma ocorrência em uma entidade.

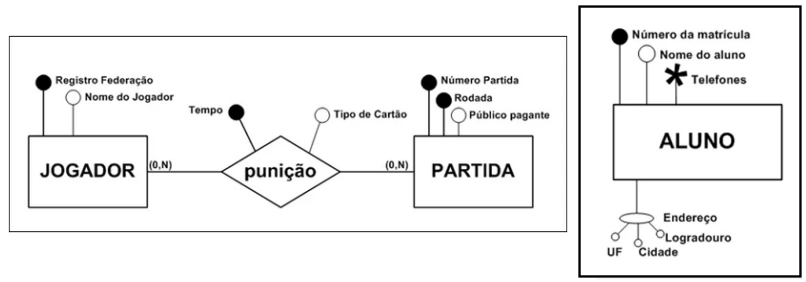
Vale ressaltar que uma entidade e/ou relacionamento pode possuir mais de um atributo identificador, desde que os mesmos em conjunto componham a identificação única.

Atributo não identificado: Representado através de uma bola vazia na extremidade do atributo. Corresponde a maioria das ocorrências de uma entidade.

atributos não identificados podem ser opcionais, ou seja, em algumas instâncias de entidade, alguns atributos poderão conter valores nulos.

Atributos multivalorados: Representado através de uma flor ou asterisco na extremidade do atributo. Representam mais de uma ocorrência de valor.

Atributos compostos: Representados através de um oval com vários nós na extremidade do atributo. Representa mais de um tipo de informação em um atributo.



Arquitetura de Banco de Dados:

Uma arquitetura de banco de dados descreve como os dados são gerenciados, desde a coleta até a transformação, a distribuição e o consumo. Ela define o plano para os dados e a maneira como eles irão fluir pelos sistemas de armazenamento de dados.

Embora tenhamos uma sólida arquitetura de banco de dados, a relação com o conceito de independência de dados é fundamental para qualquer DBA – Data Base Administrator.

Modelos de dados: conjunto de conceitos que podem ser usados para descrever a estrutura de um banco de dados. Permitem a abstração dos dados.

Estrutura de um banco de dados: tipos de dados, relacionamentos e restrições pertinentes aos dados (+ conjunto de operações para especificar como recuperar e modificar a base de dados).

Esquema: descrição (textual ou gráfica) da estrutura de um banco de dados de acordo com um determinado modelo de dados.

* Definido durante o projeto do banco de dados.
* Armazenado no catálogo do banco de dados.
* Espera-se que não seja alterado com frequência.

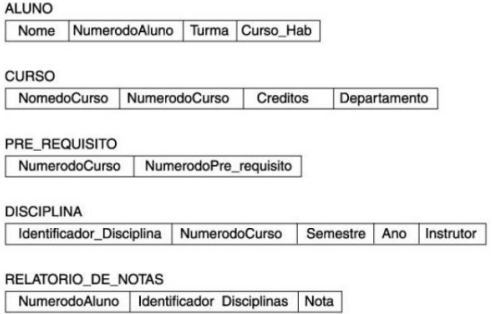
Construtor do esquema: cada objeto do esquema.

Diagrama esquemático: ilustração de alguns aspectos do esquema.

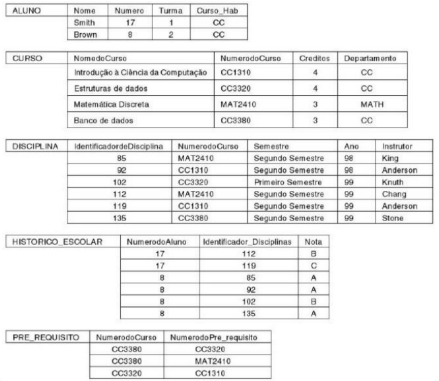
Instância (ou Estado do Banco de Dados): conjunto de dados armazenados em um banco de dados em um determinado instante de tempo.

* Inicialmente temos o estado vazio (sem nenhum dado).
* A cada atualização de dados alteramos o estado do banco de dados.
* O SGBD é parcialmente responsável por assegurar que cada estado do banco de dados seja válido.

Diagrama esquemático de um Banco de Dados



Instância de um Banco de Dados



Categorias de Modelos de Dados

Classificação de acordo com os tipos de conceitos utilizados:

* **Alto nível** ou modelos de dados conceituais.
* Modelos de dados lógicos ou implementação.
* **Baixo nível** ou modelos de dados físicos.

Características importantes da abordagem com uso de BD

* Independência de dados e programas;
* Suporte a múltiplas visões de usuários;
* Uso de catálogo.

Arquitetura proposta para realização e visualização dessas características

**Arquitetura de Três-Esquemas (ANSI/SPARC)**

Objetivo: separar as aplicações do usuário do banco de dados físico.

Os esquemas são definidos por três níveis.

* Nível externo: **Esquema externo** ou **visão de usuário**: descreve a parte do BD que um grupo de usuários tem interesse e oculta o restante do banco desse grupo.
* Nível conceitual: **Esquema conceitua**l: descreve a estrutura de todo o BD. Oculta detalhes de armazenamento físico e se concentra na descrição de entidades, tipos de dados, relacionamentos etc.
* Nível Interno: **Esquema interno**: descreve a estrutura de armazenamento físico do banco de dados. Utiliza um modelo de dados físico e descreve os detalhes completos de armazenamento de dados e caminhos de acesso ao banco de dados.

**A maioria dos SGBDs não separa os três níveis completamente.**

**Na maioria dos SGBDs os esquemas externos são especificados com o mesmo modelo de dados que descreve a informação no nível conceitual.**

**Independência de dados Lógica e Física**

Independência de Dados Lógica é a capacidade de alterar o esquema conceitual sem mudar o esquema externo ou os programas.

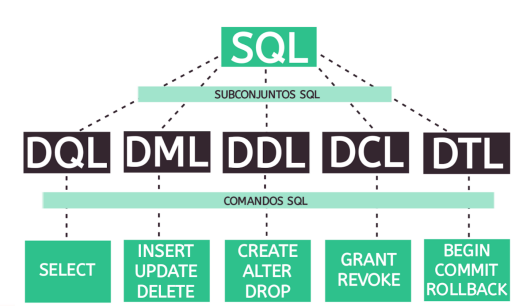
Exemplo: modificar o esquema conceitual para expandir o banco de dados (adicionando um ou mais itens de dados).

Independência Física de Dados é a capacidade de mudar o esquema interno sem ter que alterar o esquema conceitual.

Exemplo: criação de estruturas de acesso adicionais para melhorar o desempenho da recuperação ou atualização de dados.

Linguagem de Banco de Dados:

* A linguagem oferecida pelo SGBD para criação e manipulação de base de dados universal é a SQL – Structure Query Language.



O SGBD deve oferecer linguagens e interfaces apropriadas para cada categoria de usuários:

* Linguagem de Definição de Dados (Data Definition Language - DDL) é usada pelo DBA e projetistas para definir os esquemas de banco de dados.
* Linguagem de Definição de Armazenamento (Storage Definition Language - SDL) é utilizada para especificar o esquema interno em SGBDs onde uma clara separação é mantida entre os níveis conceitual e interno.
* Linguagem de Definição de Visões (View Definition Language - VDL)
  + Em uma verdadeira arquitetura três-esquemas: terceira linguagem (VDL) para especificar as visões dos usuários e seus mapeamentos para o esquema conceitual.
  + Na maioria dos SGBDs a DDL é usada para definir os esquemas conceitual e externo.
* Linguagem de Manipulação de Dados (Data Manipulation Language - DML) é usada para recuperação, inserção, remoção e modificação dos dados.

Tipos de DMLs:

* Alto nível (ou não procedural ou declarativas): os usuários especificam quais dados recuperar sem especificar como obtê-los.
* Baixo nível (ou procedural): requerem que o usuário especifique que dados são necessários e como obtê-los.

Utilitários que auxiliam o DBA no gerenciamento do SGBD

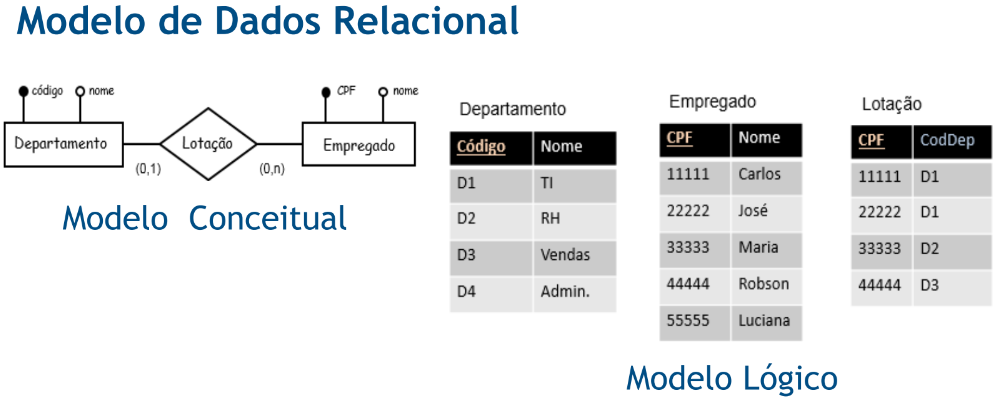
\*DBA = administrador de banco de dados.

Carga (loading): Arquivos de dados - Banco de dados. Transferência de dados de um SGBD para outro.

Backup: Para descarregar todo o BD em uma tabela. A cópia backup pode ser usada para restaurar o BD em casos de falhas.

Reorganização de arquivos: Reorganizar um arquivo de BD buscando melhorar seu desempenho.

Monitoramento de desempenho: Monitora o uso do BD e fornece estatísticas para o DBA.



Na conversão do modelo conceitual para o modelo lógico, temos o seguinte:

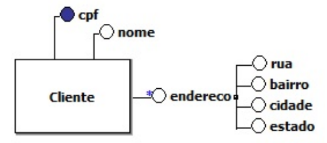
* Cada entidade é traduzida para uma tabela.
* Cada atributo define uma coluna da tabela.
* Atributos identificadores correspondem a chave primária da tabela.

Mapeamento do Modelo E.R. para Modelo Relacional

* Da literatura para a prática, existe um processo conhecido como mapeamento que nos auxilia na passagem do modelo conceitual para o modelo relacional de dados.
* Os conceitos servem para mapear, ou seja, mudar as representações do modelo de Entidade-Relacionamento para representações equivalentes no modelo Relacional.

Regras Gerais para Mapear as Entidades

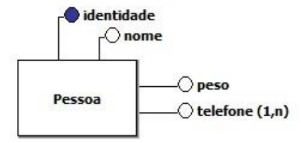
1. Toda entidade vira uma relação.
2. Atributo identificador se torna chave primária na relação.
3. Atributos simples se tornam colunas (campos).
4. Atributos compostos tornam-se atributos simples, mapeados em colunas, uma coluna para cada atributo.
5. Atributos derivados não são mapeados.
6. Atributos multivalorados podem ser mapeados de duas formas:
   1. Como n colunas, onde n é o número máximo de valores do atributo.
   2. Criando-se uma nova relação

1° Exemplo (Atributo identificador e atributo composto):

Seguindo as regras, o mapeamento do exemplo fica assim:

* Cliente (**cpf**, nome, rua, bairro, cidade, estado)

O atributo identificador **cpf**, virou a chave primária na relação; o atributo composto “endereço” foi mapeado de maneira que os seus componentes se tornaram campos na relação, mas o atributo mais abstrato endereço não foi mapeado.

2° Exemplo (Atributo multivalorado):

1° opção:

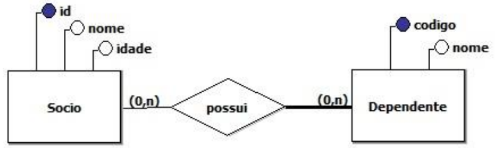
* Pessoa (**identidade**, nome, peso)
* Telefone (**identidade**, **número**)

Nessa primeira alternativa, o atributo multivalorado telefone foi mapeado em uma nova relação, que recebe a chave primária de pessoa, **identidade** e o **número**, que é o número de telefone. A chave primária dessa nova relação **Telefone**, é uma chave composta, formada pela junção de **identidade** e **número**.

2° opção:

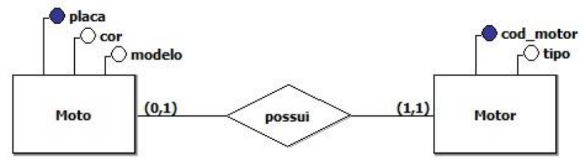
* Pessoa (**identidade**, nome, peso, tel1, tel2, tel3)

Nessa segunda alternativa foram criados três campos na relação pessoa para receber os valores correspondentes a três telefones. Essa alternativa é adequada quando se estabelece um número fixo da quantidade de números de atributos. No entanto, quando não se sabe esse número, a 1° opção se torna mais adequada.

Mapeamento de Entidades Fracas

No exemplo acima, linha mais grossa que liga Dependente, representa que Dependente é uma entidade fraca, ou seja, não existe se Sócio não existir. O seu mapeamento para o modelo relacional de dados fica assim:

* Socio (**id**, nome, idade)
* Dependente (**id\_socio**, **codigo**, nome)
* A chave primária da relação Dependente é composta da chave primária de Socio, **id\_socio**, mais a chave primaria de Dependente, que nesse caso é o **codigo**;
* Portanto, a chave primaria de entidades fracas será sempre **chave composta**;
* O relacionamento possui, por apresentar uma relação de muitos-para-muitos, detalhes mais a frente, vira uma entidade-relacionamento: Possui (**id\_socio, cod\_dep**);
* Então a entidade-relacionamento **Possui** apresenta uma chave composta. Caso o relacionamento possui tivesse atributos ligados a ele, os mesmos seriam atributos da E-R **Possui**.

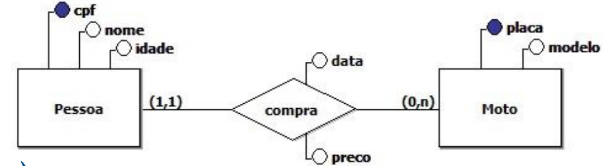
Relacionamentos binários 1:1

Moto (**placa**, cor, modelo)

Motor (**cod\_motor**, tipo, placa\_moto)

Em um relacionamento 1:1, de acordo com os passos acima, escolhe-se uma das relações para receber a chave estrangeira, essa que é a chave primária da relação que ocorre o relacionamento.

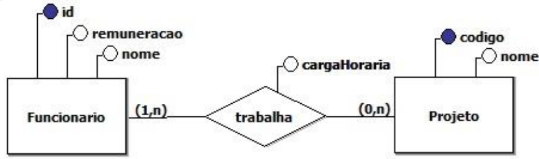
* Observação: Não é regra, mas geralmente, nesses casos, escolhe-se a relação que possui participação total no relacionamento. Nesse exemplo, é motor, pois é (1,1), ou seja, irá sempre existir, enquanto moto (0,1) pode ou não existir.

Relacionamentos binários 1:n

Pessoa (**cpf**, nome, idade)

Moto (**placa**, modelo, cpf\_pessoa, data, preco)

Em um relacionamento 1:n, escolhe-se a relação “n” (no caso Moto) e inclui como chave estrangeira a chave primária da outra relação. Inclui **também** os atributos do relacionamento.

Relacionamentos binários n:n

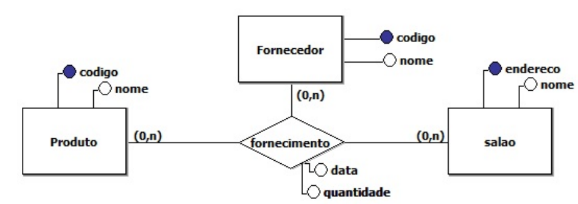
Funcionario (**id**, remuneracao, nome)

Projeto (**codigo**, nome)

Trabalha (**id\_funcionario**, **código\_projeto**, cargaHoraria)

Em um relacionamento n:n, cria-se uma nova relação e inclui como chave estrangeira, as chaves primarias das relações em que ocorre o relacionamento.

* A chave primaria dessa nova relação (no caso Trabalha) será uma chave composta.
* Por fim, carrega também os atributos do relacionamento, cargaHoraria, neste caso.

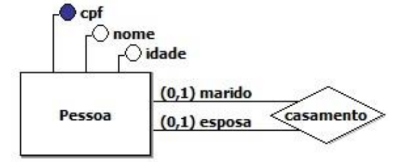
 Relacionamentos n-ários:

Fornecedor (**codigo**, nome)

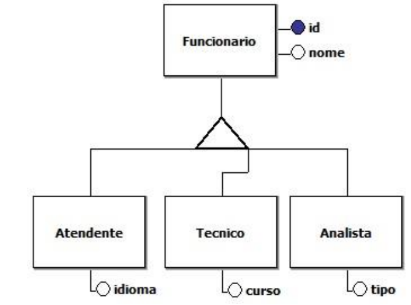
Produto (**codigo**, nome)

Salao (**endereco**, nome)

Fornecimento (**codigo\_fornecedor**, **codigo\_produto**, **endereço\_salao**, data, quantidade)

Auto Relacionamento (Unário):

Pessoa (**cpf**, nome, idade, id\_conjuge)

Generalização / Especialização

Atendente (**id**, nome, idioma)

Tecnico (**id**, nome, curso)

Analista (**id**, nome, tipo)

Todo o banco de dados de base universal, apresenta uma estrutura lógica relacional.

* Neste contexto, além de gerar o modelo conceitual a partir do minimundo estudado previamente, e, além de gerar o modelo lógico relacional a partir do modelo conceitual, é necessário aplicar regras de normalização dessas bases de dados lógicas relacionais a fim de melhor organizar e otimizar os bancos de dados para facilitar o processo de manutenibilidade da base de dados pelo DBA.

Normalização

* O processo de normalização de dados representa uma série de passos que se seguem no projeto de um banco de dados que permitem um armazenamento consistente e um eficiente acesso aos dados de um banco de dados relacional.
* Esses passos reduzem a redundância de dados e, consequentemente, as chances de ocorrerem inconsistências. A normalização serve para analisar tabelas e organizá-las de forma que sua estrutura seja simples, relacional e estável, para que o gerenciamento possa ser também simples.

**Objetivos da normalização:**

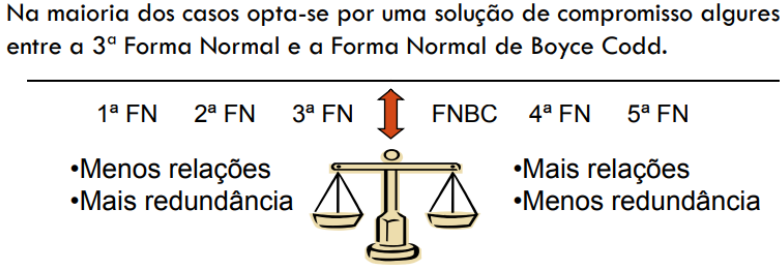
* Evitar perda e repetição da informação e atingir uma forma de representação adequada para o que se deseja armazenar;
* Oferecer mecanismos para analisar o projeto de Banco de Dados e a identificação de erros;
* Oferecer métodos para corrigir o problema.

**Perigos Potenciais nos Projetos de Bancos de Dados Relacionais:**

* Repetição da Informação
  + Informações repetidas consomem espaço de armazenamento;
  + Dificuldade de atualização.
* Incapacidade de representar a informação
  + Incidência de valores nulos.
* Perda da Informação
  + Projetos mal elaborados sugerem a decomposição de esquemas relacionais com muitos atributos

**A normalização visa controlar os seguintes problemas:**

* Grupos repetitivos (atributos multivalorados) de dados;
* Variação temporal de certos atributos, dependências funcionais totais ou parciais em relação a uma chave concatenada;
* Redundâncias de dados desnecessárias;
* Perdas acidentais de informação;
* Dificuldade na representação de fatos da realidade observada;
* Dependências transitivas entre atributos.

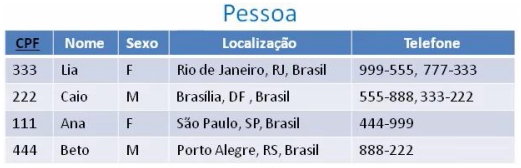


**Primeira Forma Normal**

* É a parte da definição formal de uma relação.
* Foi definida para não permitir atributos multivalorados, atributos compostos e suas combinações (atributos complexos).

“Uma relação está na 1FN se e somente se todos os seus atributos contêm apenas valores atômicos (simples e indivisíveis)”.

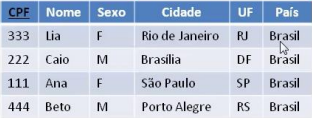
Exemplo Prático:



Na relação Pessoa, temos um atributo composto (Localização) e um atributo multivalorado (Telefone) e, portanto, esta relação NÃO está na **1FN**.

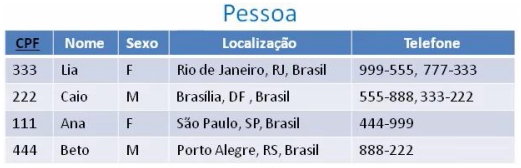
**Primeiro passo:** devemos eliminar o atributo composto da relação

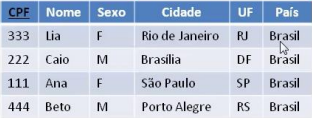
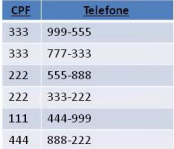
* Crie uma nova relação onde cada semântica do atributo composto passe a ser representado por uma coluna



A nova relação ainda Não está na **1FN.**

Em seguida, para eliminar o atributo multivalorado devemos criar uma nova relação cuja Primary-Key seja composta pela PK da relação inicial, **CPF**, + o atributo identificador **Telefone**, neste caso.



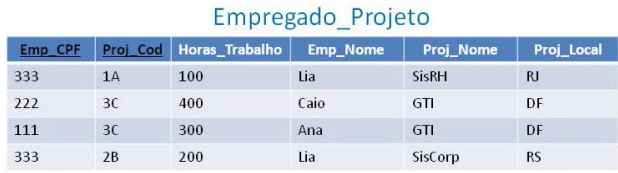
Somente agora, as relações apresentadas estão efetivamente na **1FN.**

**Segunda Forma Normal**

“Uma relação está na 2FN se e somente se estiver na 1FN e NÃO contém dependências parciais”.

* **Dependência parcial**: ocorre quando uma coluna depende apenas de uma parte da chave primária composta.
* Então, para que uma relação esteja na **2FN**, NÃO deve possuir atributo não-chave funcionalmente determinado por parte da chave primária.

Exemplo Prático:



Na relação Empregado\_Projeto, temos um mix de atributos que apresentam dependências parciais a chave composta da relação, **Emp\_CPF** **+ Proj\_Cod**, neste caso e, portanto, esta relação NÃO está na **2FN**.

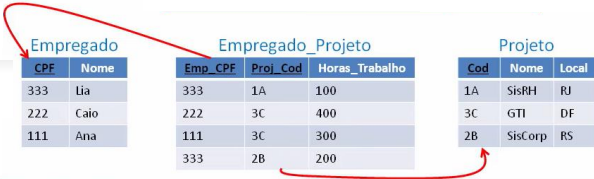
* Inicialmente, devemos identificar todas as reais dependências parciais existentes.



Cria-se a relação Empregado (**CPF**, Nome), Projeto (**Cod**, Nome, Local) e outra relação

Empregado\_Projeto (**Emp\_CPF**, **Proj\_Cod**, Horas\_Trabalho) com uma chave composta.





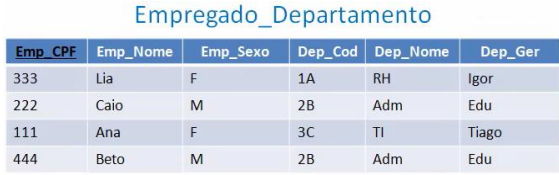
Dessa forma, a relação inicial, se transforma em três novas relações com apenas dependências funcionais ligadas a PK e, portanto, agora já temos aplicada a **2FN**.

**Terceira Forma Normal**

“Uma relação está na 3FN se e somente se estiver na 2FN e nenhum atributo não-primo (isto é, que NÃO seja membro de uma chave) for transitivamente dependente da chave primária”.

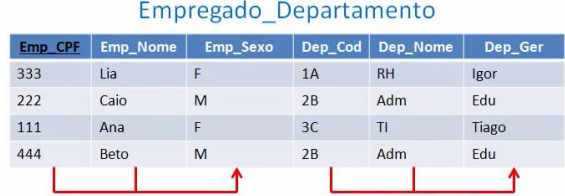
* Dependência transitiva: ocorre quando uma coluna, além de depender da chave primária de uma relação, depende de outra coluna ou conjunto de colunas da relação.
* Então, para que uma relação esteja na **3FN**, NÃO deve ter um atributo não-chave funcionalmente determinado por outro atributo não-chave.

Exemplo Prático:

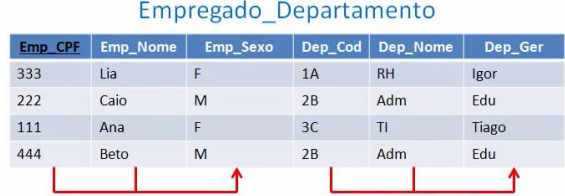


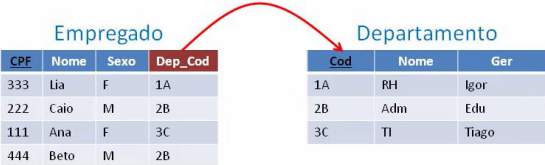
Na relação Empregado\_Departamento, temos um conjunto de atributos (Dep\_Nome e Dep\_Ger) que apresentam dependências transitivas ao atributo não-chave, Dep\_Cod, que por sua vez, é dependente da chave **Emp\_CPF**, e, portanto, esta relação NÃO está na **3FN**.

* Identificando as dependências transitivas:



Criar a relação Empregado (**CPF**, Nome, Sexo, Dep\_Cod), e outra Departamento (**Cod**, Nome, Ger) de tal forma que seja eliminada a dependência transitiva entre atributos não -chave.





* A **3FN** é aquela que, na maioria dos casos, termina o processo de normalização. No entanto, em casos bastante específicos, ainda existem algumas anomalias, resolvidas pela **FNBC (Forma Normal de Boyce-Codd)**.

**Forma Normal de Boyce-Codd.**

Por definição, uma relação está na FNBC se...

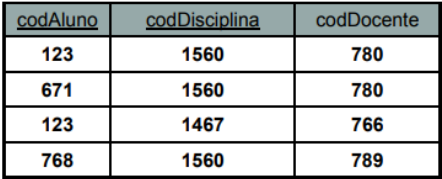
- Está na 3FN.

- Todos os determinantes são chaves candidatas.

Exemplo Prático:

Leciona (**CodAluno**, **CodDisciplina**, CodDocente)

A relação apresentada serve para registrar os alunos das turmas práticas das disciplinas de certa Universidade. Sabe-se que cada disciplina pode ser lecionada por diversos docentes. No entanto, cada docente só pode lecionar uma disciplina.



A relação satisfaz as premissas relativas a **1FN**, **2FN** e **3FN**.

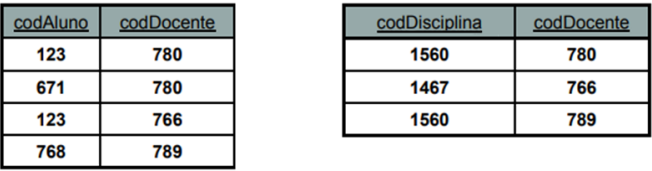
Forma Normal de Boyce-Codd

* Os atributos contêm valores atômicos.
* Não existem grupos de valores repetidos.
* Os atributos não chave dependem exclusivamente da chave.
* O atributo “CodDocente” **NÃO É CHAVE CANDIDATA**, no entanto, é um determinante.
* Uma vez que cada docente só pode lecionar uma disciplina, temos que

CodDocente**->CodDisciplina**.

A passagem da relação anterior para a FNBC exigiria a sua **decomposição** em duas relações.

* Uma delas registra os docentes de cada aluno;
* A outra registra a disciplina lecionada por cada docente;
* Todos os atributos das relações são chaves candidatas.



Normalmente uma relação na **FNBC** também já se encontra na **4FN** e **5FN**, surgindo estas para resolver casos muito raros.

**Quarta Forma Normal**

Diz-se que uma relação está na 4FN se...

* Está na FNBC.
* Não existirem dependências multivaloradas.
  + Uma vez que existam dependências multivaloradas, para colocar a relação na 4FN é necessário efetuar a sua decomposição.

**Quinta Forma Normal**

* Corresponde ao grau de normalização mais elevado.
* Na prática, uma relação na **5FN** NÃO PODE SER DECOMPOSTA DE NENHUMA FORMA SEM QUE HAJA PERDA DE INFORMAÇÃO.
* Por definição, uma relação está na 5FN se:
  + Está na 4FN;
  + Não existem dependências de junção relativas a subconjuntos próprios do conjunto de todas as possíveis projeções.

Na prática a normalização NÃO DEVE ser levada às últimas consequências, pois a proliferação de relações terá consequências ao nível do desempenho da base de dados.

A necessidade de manipular múltiplas relações para consultar, registrar, alterar informações é o principal condicionamento à velocidade de resposta de um SGBD.