Banco de Dados I

Última atualização: 25/05/2021

Sumário

Caro Aluno,

Esta apostila tem como objetivo permitir que você possa acompanhar a aula, fazendo algumas anotações e servindo como orientação no momento em que você for estudar os temas abordados. Apesar disso, não dispensa a leitura dos livros indicados na bibliografia, principalmente do livro texto da disciplina, uma vez que eles tratam os assuntos com mais detalhes, permitindo, assim, o enriquecimento no estudo de Bancos de Dados.

Bom trabalho!

Alguns DER , exemplos e conceitos foram retirados do livro Projeto de Banco de Dados do prof. Carlos Alberto Heuser.

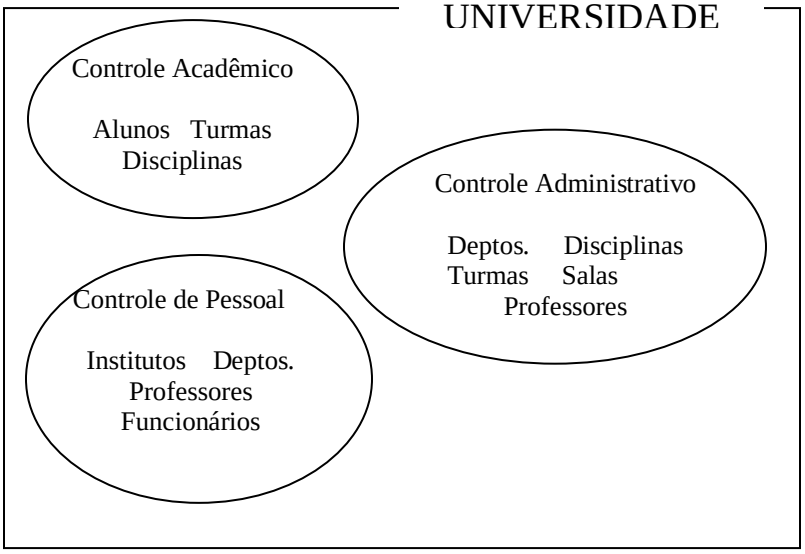
1. Introdução a Sistemas de Bancos de Dados

Para entender o estudo de banco de dados é necessário a definição de três termos principais:

- *banco de dados*: corresponde a um conjunto de informações relacionadas que possuem algum significado. São informações referentes a um empreendimento particular;
- *sistema gerenciador de banco de dados (SGBD)*: consiste em uma coleção de dados interrelacionados e em um conjunto de programas para acessá-los. O principal objetivo de um SGBD é prover um ambiente que seja conveniente e eficiente para recuperar informações de banco de dados;
- *sistema de banco de dados (SBD)*: é constituído pelo banco de dados propriamente dito e o software necessário para gerenciá-lo (SGBD) e manipulá-lo (consultas e aplicações do usuário).

1.1 Objetivos dos sistemas de bancos de dados

Para entender a importância dos SBD observe o exemplo abaixo baseado num sistema de arquivos convencional, ou seja, que não utiliza um banco de dados.



Situação:

- Cada aplicação da organização com o seu conjunto de dados.
- Descrição dos dados fica dentro da aplicação.
- Não existe compartilhamento de dados entre as aplicações.

Problemas:

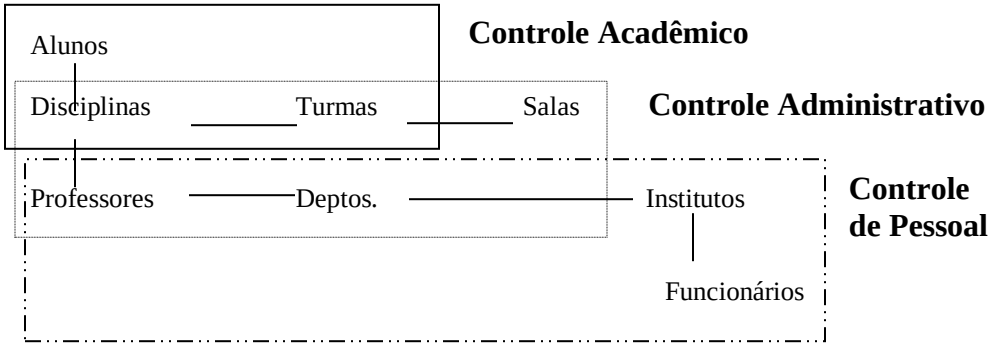
- Redundância de dados: um sistema de arquivos pode gerar um alto nível de redundância e duplicação de dados, porque o mesmo dado é armazenado em diferentes arquivos;
- Difícil manutenção dos dados (inconsistência dos dados): a manutenção fica difícil porque é necessário fazer a manutenção em mais de um lugar. Quando a mudança de um mesmo dado é feita em um arquivo mas não nos outros, o resultado são dados inconsistentes;
- Falta de uma padronização na definição dos dados: posso ter o mesmo dado com o formato e tipo diferentes;
- Não há preocupação com a segurança dos dados (segurança de acesso e segurança contra falhas);
- Limitações no compartilhamento dos dados: cada usuário tem os seus arquivos e programas de aplicação. Fica difícil o compartilhamento de dados contribuindo para a redundância e ineficiência no geral. Devido a este ambiente descentralizado fica difícil estabelecer um padrão e um controle sobre quais dados são processados.

Necessidade de um banco de dados:

- melhor organização e gerência dos dados;
- controle centralizado dos dados.

∗ Os dados, em um banco de dados, são armazenados de forma independente dos programas que os utilizam, servindo assim a múltiplas aplicações de uma organização.

Exemplo com BD

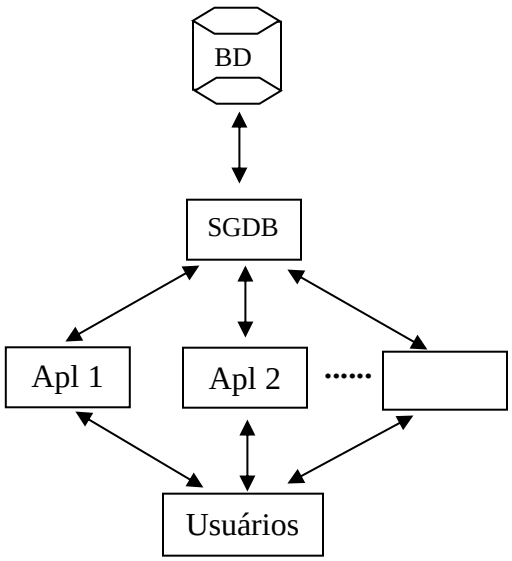


Vantagens:

- Dados são armazenados em um único local físico
- Dados são compartilhados pelas aplicações

- Independência dos dados
- Aplicações não se preocupam com a gerência dos dados

Um Banco de Dados dentro do contexto de um ambiente computacional



1.2 Funções de um SGBD

a) Definição de dados e métodos de acesso

- DDL (Data Definition Language): especificação do esquema do BD
- DD (Dicionário de Dados): armazenamento dos metadados (catálogo do sistema)
- DML (Data Manipulation Language): permite a manipulação de dados
- Processamento eficaz de consulta

b) Restrições de Integridade: controle sobre a integridade dos dados armazenados.
   - Estados possíveis de serem assumidos pelos dados. Ex: o código de um aluno deve ser um valor entre 000 e 999.
   - Manutenção de relacionamentos válidos entre os dados. Ex: todo professor deve pertencer a pelo menos um departamento.
   - Triggers

c) Segurança dos Dados
   - Controle de acesso (autorização)
   - Segurança contra falhas: Transação, Sistema e Meio de armazenamento

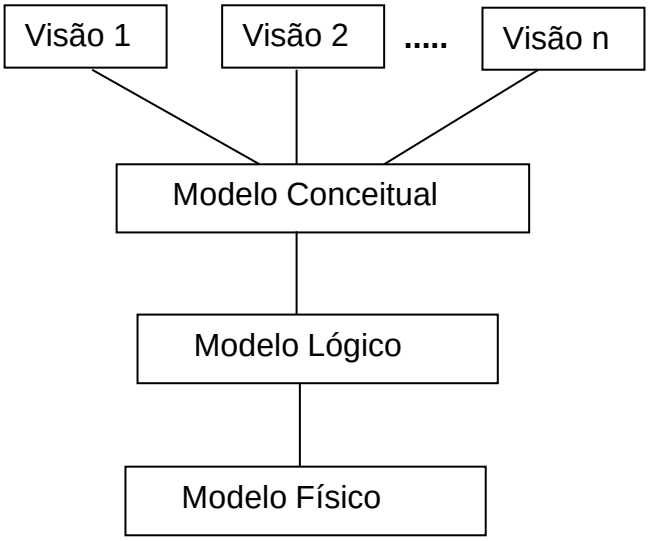
d) Controle de Concorrência: sanar conflitos de acesso a dados

e) Independência dos dados

- Independência Física: modificações no esquema de gerenciamento dos dados (esquema físico) sem afetar a implementação das aplicações e a definição conceitual dos dados.
- Independência Lógica: independência da estrutura conceitual dos dados.
  - ❖ Visões diferenciadas da estrutura conceitual (Ok!)
  - ❖ Modificações na estrutura conceitual (problema!)

1.3 Níveis de Visões

Forma de visão dos dados dentro do contexto aplicação-SGBD

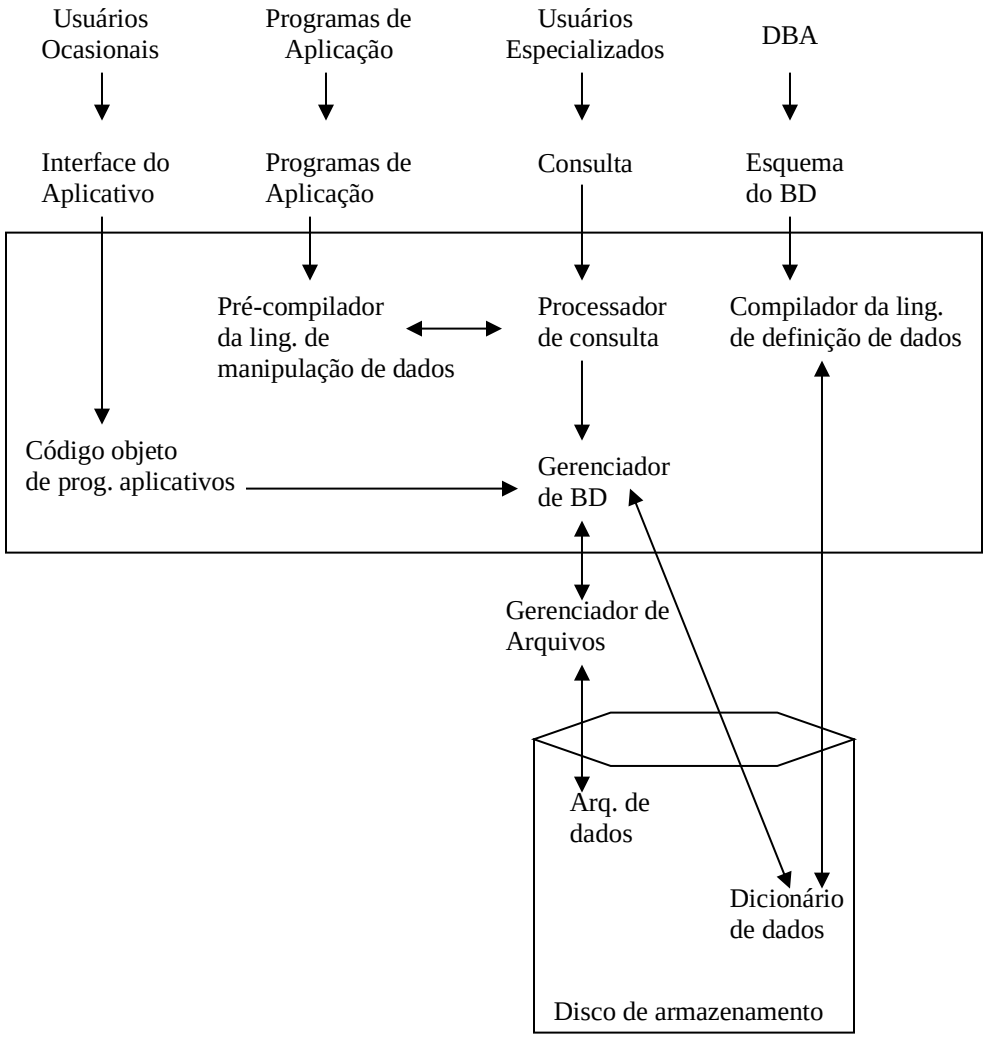


1.4 Usuários do SGBD

a) Administrador de banco de dados (DBA): controle de diversas funcionalidades do SGBD
   - definição do esquema conceitual
   - definição da estrutura de armazenamento e métodos de acesso
   - modificação do esquema conceitual, estrutura de armazenamento e métodos de acesso
   - concessões de autorização de acesso
   - especificação de restrições de integridade
   - controle das operações de recuperação após falhas

b) Usuários especializados: interagem diretamente com o SGBD
c) Programadores de aplicação: definem aplicações para acesso aos dados
d) Usuários ocasionais: utilizam aplicações que acessam o BD.

Estrutura Geral do SGBD



2. Modelos de Dados

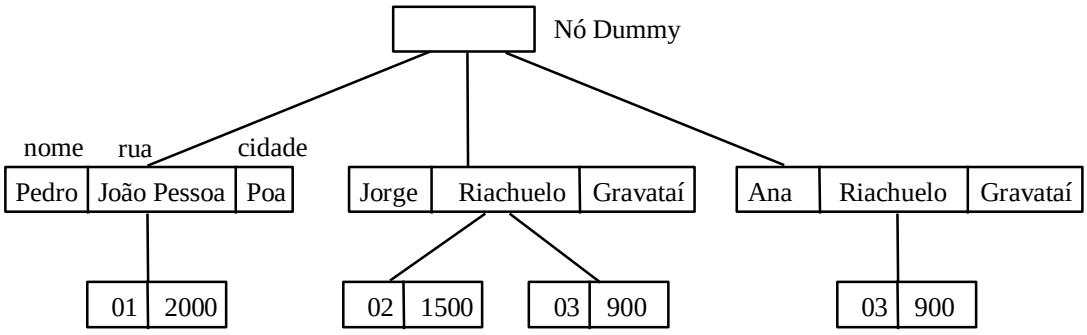
⇒ **Hierárquico**
⇒ **Rede**
⇒ **Relacional**
⇒ **Abordagem pós-relacional**

2.1 Modelo Hierárquico

Importância: IMS (information Management System) da IBM, largamente utilizado durante a década de 70 e início da década de 80;

Características
- Dados organizados em uma coleção de registros interconectados através de ligações.
  Registros: Conjuntos de campos (informações ≠ s)
  Ligação : Associação entre dois registros .

- Esquema definido em forma de árvore
EX .: Sistema Bancário : 2 registros
⇒ cliente
⇒ conta



Representação Diagramática

- Diagrama de Estrutura em Àrvore (DEA)

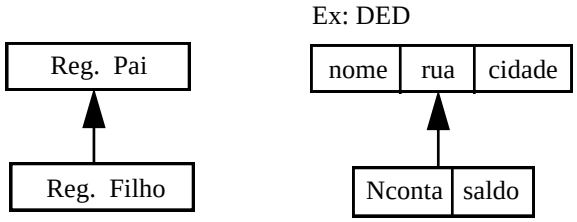
a) Características

⇒ Caixas: Tipos de Registros
⇒ Linhas: Ligações

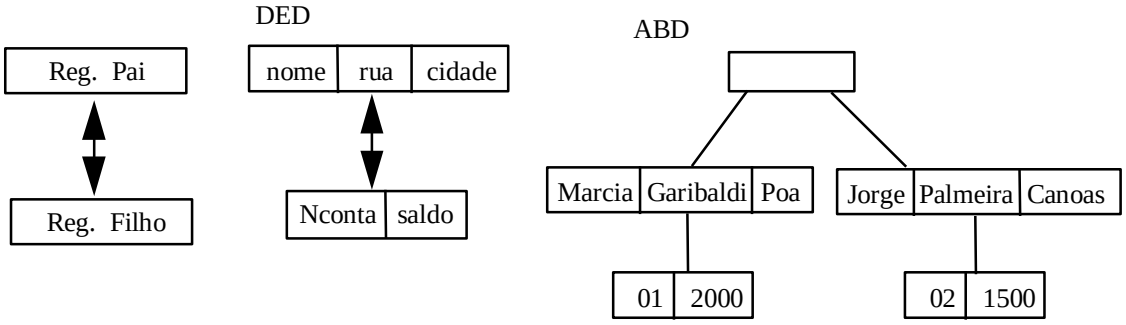
DED - O diagrama de estrutura de dados é um esquema para o BD. Consiste de dois componentes: caixas (que correspondem a tipos de registros) e linhas (que correspondem a ligações entre este registros).

b) Representação dos Relacionamentos

1-Relacionamentos um-para-muitos:



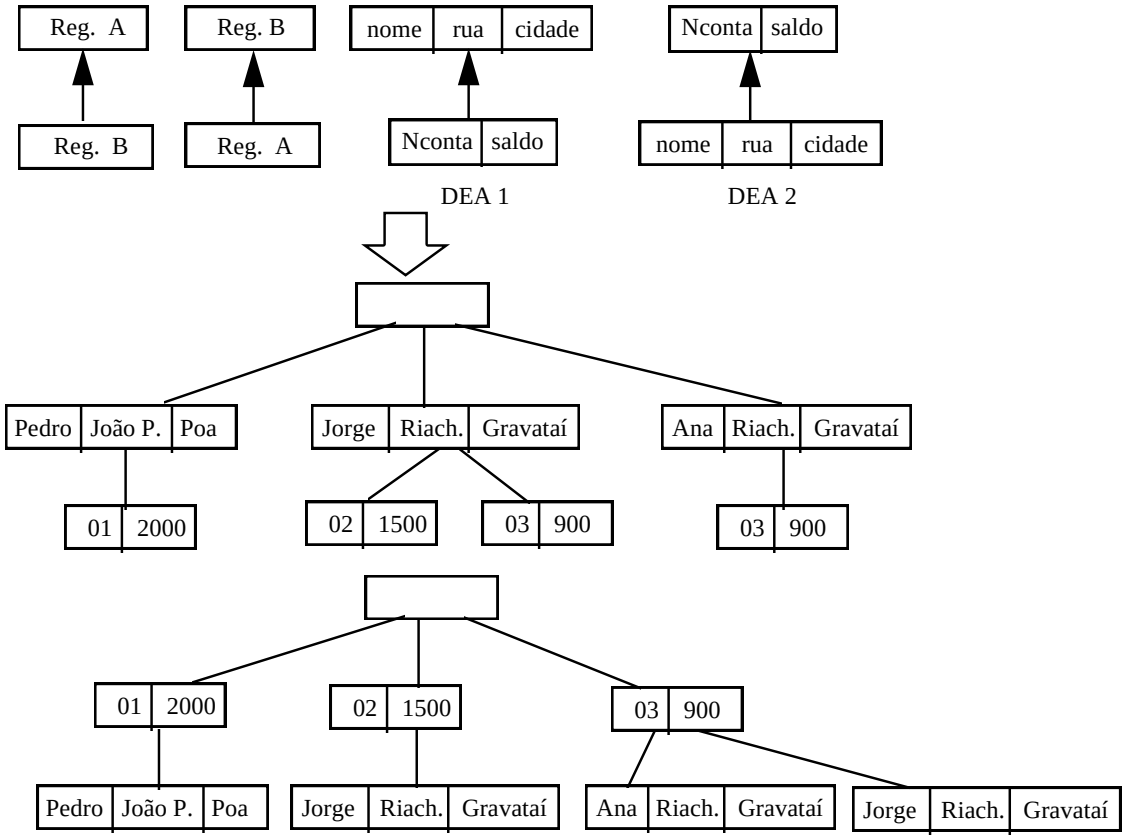
2-Relacionamento um-para-um:



3-Relacionamentos muitos-para-muitos:

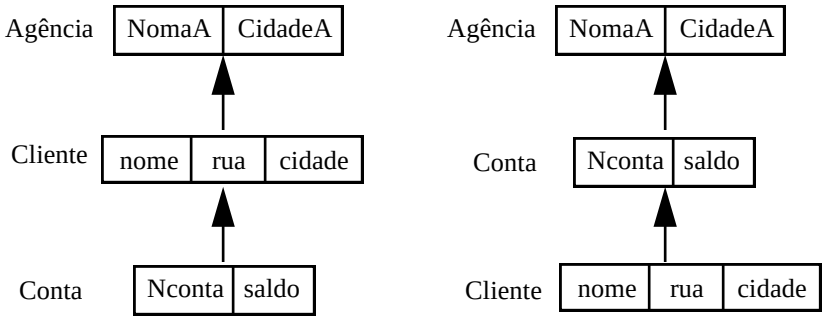
O modelo não suporta este relacionamento diretamente.

Solução: Aplicar o princípio um-para-muitos p/cada um dos dois tipos de registros.



- 2 DEAs: Redundância Maior
- 1 DEAs: Depende do tipo de Consulta

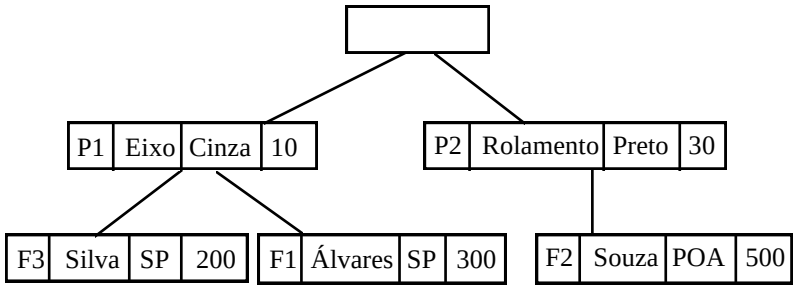
Exemplo2: Um cliente pode ter diversas contas em uma agência de banco. Uma conta pode pertencer a diversos clientes.



Problemas

⇒ Redundância de informação
ex: relacionamento muitos-para-muitos (cliente-conta)

⇒ Dificil atualização de registros filhos
ex: percorrer toda a árvore para atualizar todos os saldos de uma determinada conta
⇒ Problema na inserção
ex: para inserir os dados de um forncedor que ainda não forneceu nenhuma peça, seria necessário criar um registro fantasma



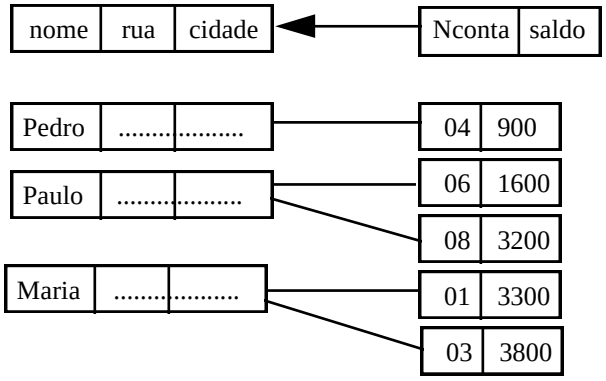
⇒ Problemas na exclusão
ex: a exclusão da única peça fornecida por um fornecedor inplicaria na exclusão de seus dados
⇒ Problemas de Assimetria nas Consultas
Tempo de acesso a registros dos níveis mais baixos na hierarquia

2.2 Modelo de Rede

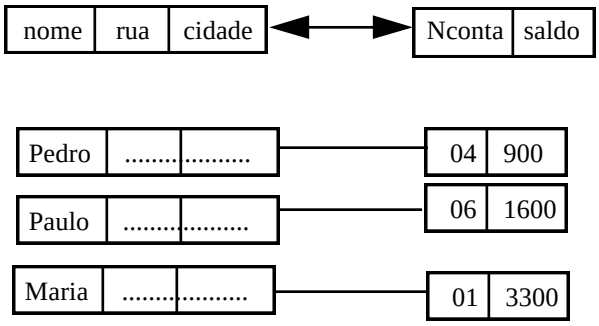
Características
⇒ Dados organizados em registros conectados através de ligações
⇒ Ligações : Associações entre exatamente 2 tipos de registros.
⇒ Permite todos os tipos de relacionamentos: 1:1 , 1:N , M:N
⇒ 1971 : Especificação padrão de banco de dados: Modelo DBTG (Data Base Task Group) CODASYL (Conference on Data Systems and Languages)
⇒ surgem diversos BD comerciais baseados no modelo de Rede.

Relacionamentos

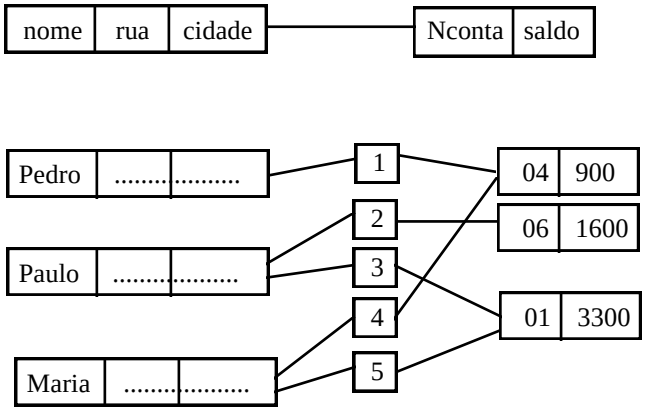
⇒ relacionamento 1:N



⇒ relacioanamento 1:1



⇒ relacionamento M:N



Vantagens em relação ao modelo Hierárquico

⇒ Representação de todos os tipos de relacionamentos
⇒ Sem redundância de informações
⇒ Maior simetria nas consultas

Desvantagens do Modelo de Rede

⇒ Bancos de dados com muitos tipos de entidades podem resultar em esquemas muito confusos;
⇒ Utilização de ligação física entre os registros;

3. Modelo Entidade-Relacionamento

Alguns diagramas , exemplos e conceitos foram retirados do livro Projeto de Banco de Dados do prof. Carlos Alberto Heuser.

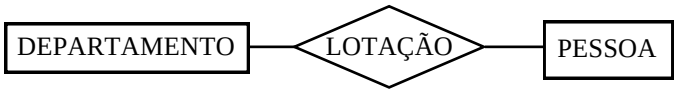
3.1 Elementos Básicos

A) ENTIDADE

⇒ Conjunto de objetos da realidade modelada sobre os quais deseja-se manter informação.

B) RELACIONAMENTO

⇒ Conjunto de associações entre entidades



<u>Cardinalidade de Relacionamentos</u>

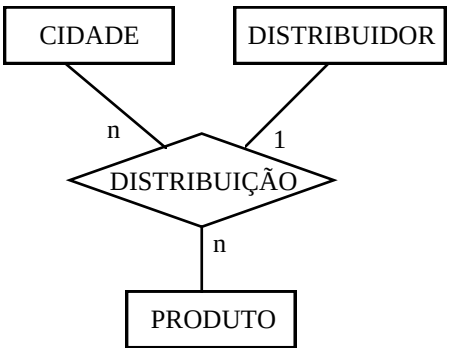
⇒ Cardinalidade (mínima, máxima) de entidade em relacionamento = número (mínimo, máximo) de ocorrências de entidade associadas a uma ocorrência da entidade em questão através do relacionamento.

⇒ Cardinalidade máxima: 1 ou muitos (n)
⇒ Ex. com cardinalidade máxima:

1 - expressa que a uma ocorrência de empregado pode estar associada uma ocorrência de departamento;
n - expressa que a uma ocorrência de departamento podem estar associadas muitas ocorrências de empregados;

Relacionamentos Ternários

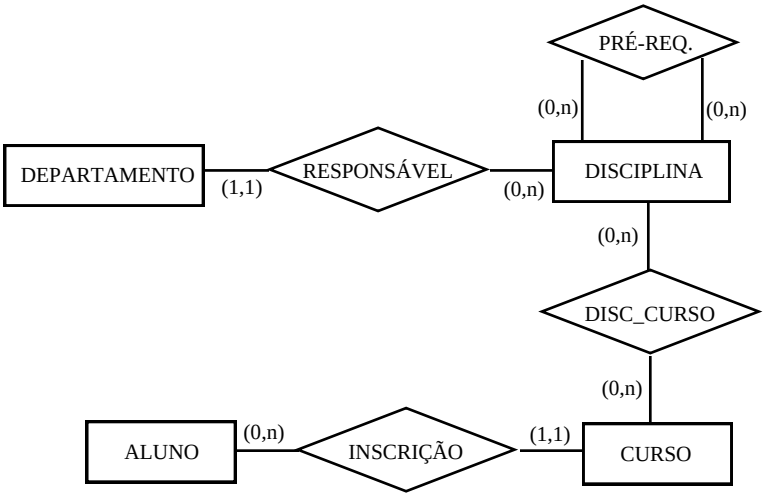
⇒ Em um relacionamento ternário a cardinalidade se refere a pares de entidades.



Cardinalidade Mínima

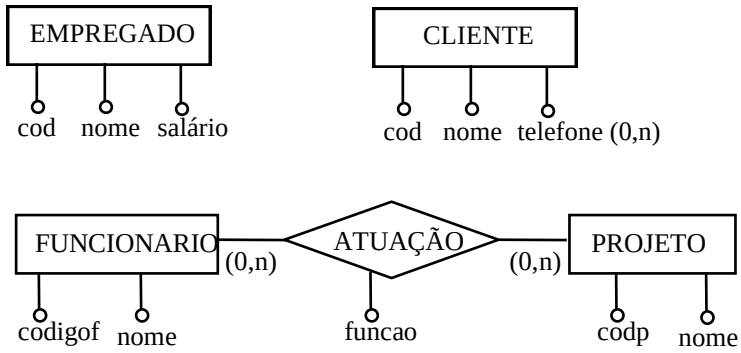
⇒ Número mínimo de ocorrências de entidades que são associadas a uma ocorrência de entidade através de um relacionamento.
⇒ Cardinalidades: 0 (associação opcional) ou 1 (associação obrigatória).

⇒ DER para o controle acadêmico de uma universidade

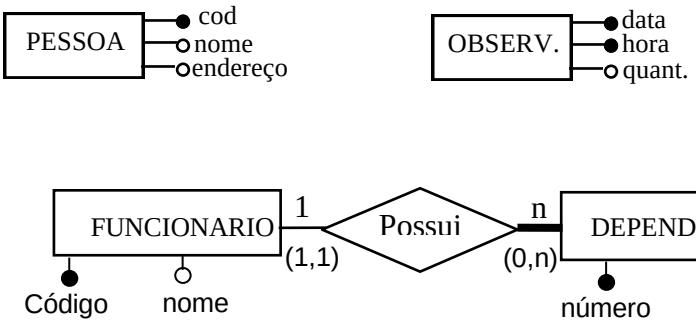


C) ATRIBUTO

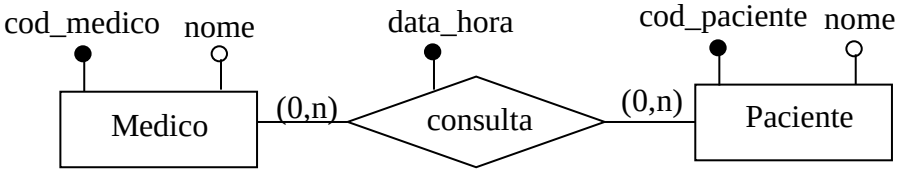
⇒ Dado que é associado a cada ocorrência de uma entidade ou de um relacionamento.



Identificador de Entidade

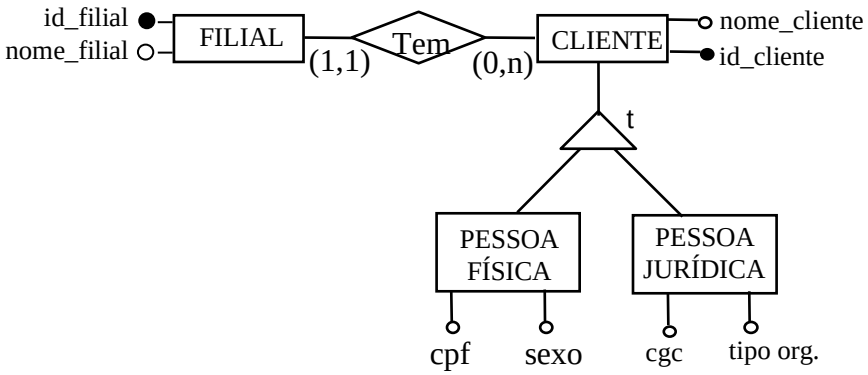


Identificador de Relacionamentos



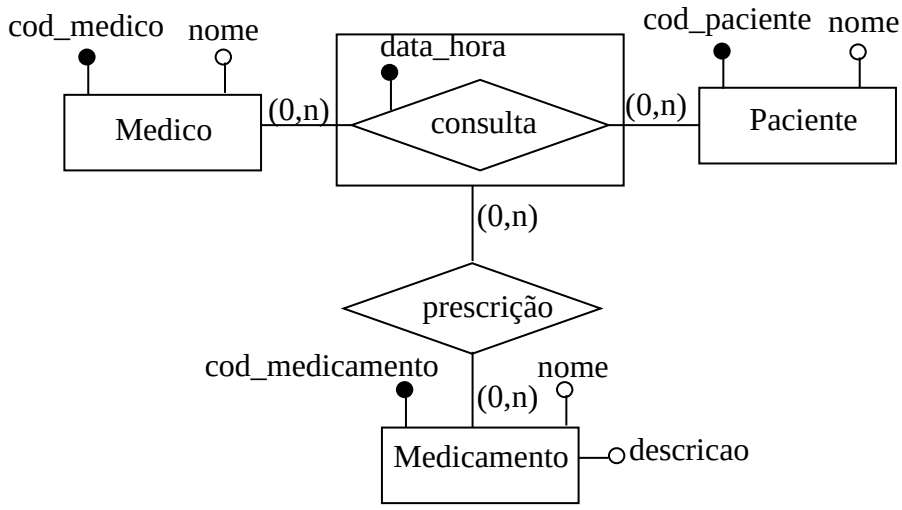
3.2 Outros Conceitos

GENERALIZAÇÃO/ESPECIALIZAÇÃO

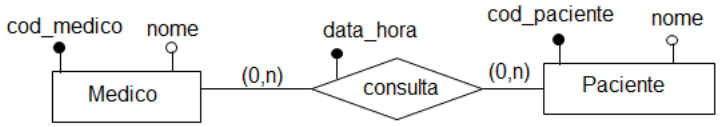


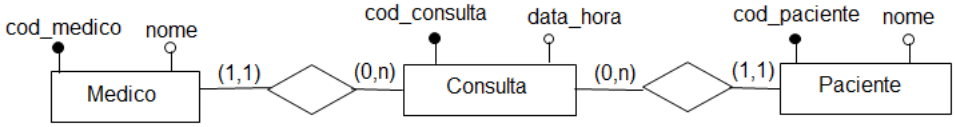
ENTIDADE ASSOCIATIVA

⇒ incluir os medicamentos que são prescritos por uma médico a um paciente em uma consulta;



EQUIVALÊNCIA ENTRE MODELOS

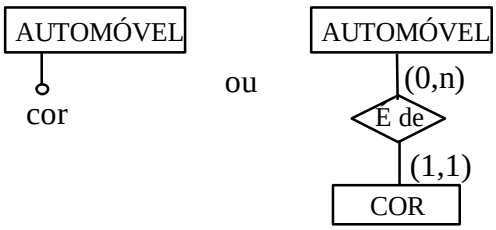




3.3 Que Cuidados Tomar ao Construir um Modelo E-R

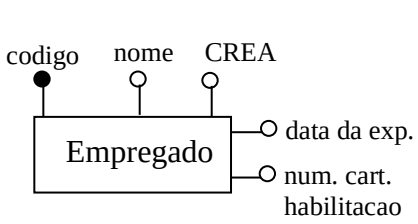
a) Atributo versus entidade relacionada

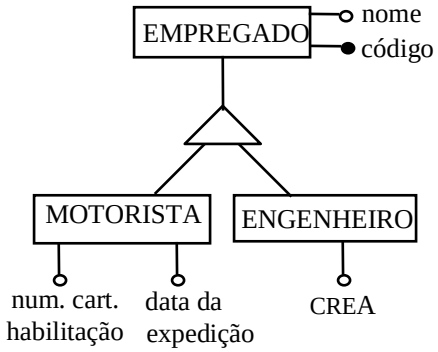
⇒ Caso o objeto cuja modelagem está em discussão esteja vinculado a outros objetos, o objeto deve ser modelado como entidade. Caso contrário, o obj. deve ser modelado como atributo.



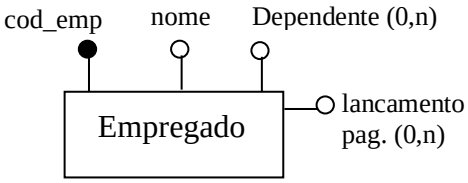
b) Atributos opcionais e multi-valorados

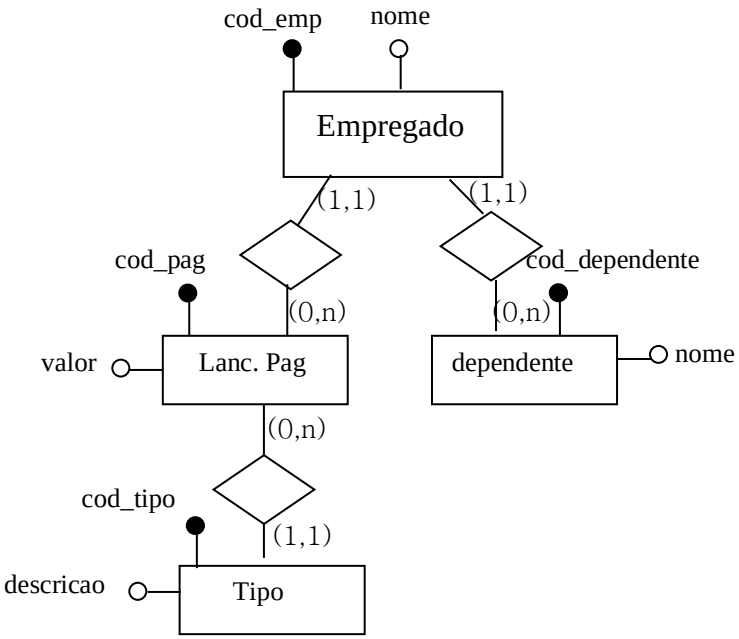
⇒ Atributos opcionais indicam subconjuntos de entidades que são modeladas mais corretamente através de especializações.





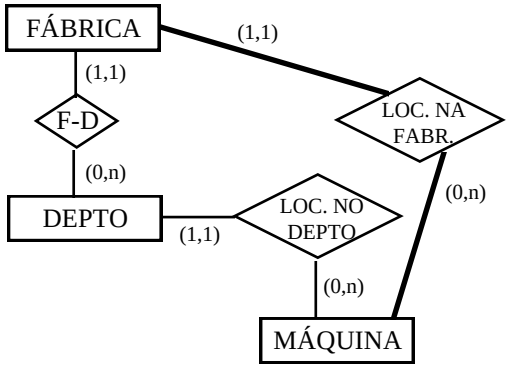
⇒ Atributos multi-valorados são indesejáveis.



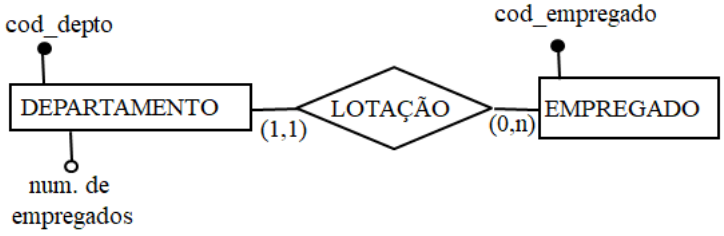


c) O modelo deve ser livre de redundância

⇒ Relacionamentos redundantes - são relacionamentos que são resultado da combinação de outros relacionamentos entre as mesmas entidades.



⇒ Atributos redundantes - são atributos deriváveis a partir da execução de procedimentos de busca de dados e/ou cálculos sobre a base de dados (num. de empregados).



4. O Modelo Relacional

O modelo relacional foi criado nos 70 e deu origem aos bancos de dados relacionais. No modelo relacional os dados são organizados em relações, uma relação é composta por tuplas e cada tupla tem um conjunto de atributos. Nos bancos de dados relacionais utilizados a tabelas (relações), linhas (tuplas), colunas ou campos (atributos).

As ligações entre as linhas de diferentes tabelas é feita através da ligação dos valores dos atributos.

Por exemplo:

Tabela departamento

id_departamento	nome_departamento
100	Vendas
101	RH
102	Informática

Tabela funcionario

id_funcionario	nome_funcionario	id_departamento
1	Ana	100
2	Heitor	101
3	Ricardo	100

∗ Estas tabelas podem estar relacionadas? Sim, a partir do campo id_departamento

4.1 Chaves

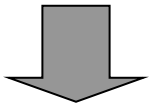
Um conceito importante para o modelo relacional é o conceito de chave. Nesse contexto tem-se três tipos de chaves:

- *chave primária*: é qualquer coluna ou combinação de colunas que identifica uma única tupla em uma tabela;
- *chave estrangeira*: é uma coluna ou combinação de colunas, cujos valores aparecem necessariamente na chave primária (ou única) de uma tabela;
- *chave alternativa ou candidata*: em alguns casos, mais de uma coluna ou combinações de colunas podem servir para distinguir uma linha das demais. Uma das colunas é escolhida como chave primária e as demais são chamadas chaves alternativas.

Exemplos:

Dependente

id_funcionario	nro_dependente	nome	categoria	data_nascimento
1	01	João	filho	12/12/1991
1	02	Maria	esposa	01/01/1950
2	01	Ana	esposa	05/11/1955
5	01	Paula	esposa	04/07/1962
5	02	José	filho	03/02/1985

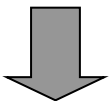


chave primária

∗ Em algumas situações (como na tabela dependente), apenas um dos valores dos campos que compõem a chave não é suficiente para distinguir uma linha das demais. É necessário a utilização de outro campo, neste caso a tabela possui uma chave primária composta. No exemplo id_funcionario, nro_dependente é a chave primária da tabela Dependente.

departamento

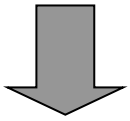
id_departamento	nome_departamento
100	Vendas
101	RH
102	Informática



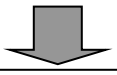
Chave primária

funcionario

id_funcionario	nome_funcionario	id_departamento
1	Ana	100
2	Heitor	101
3	Ricardo	100



chave primária



chave estrangeira que referencia a tabela departamento

funcionario

id_funcionario	nome_funcionario	id_departamento	cpf
1	Ana	100	698.567.123-90
2	Heitor	101	543.786.333-86
3	Ricardo	100	895.325.562-65

chave alternativa

Restrições de Integridade

A principal função de um SGBD é garantir a integridade dos dados, ou seja, garantir que os dados estão de acordo com as regras de integridade (regra de consistência dos dados) definidas no banco de dados. Os SGBDs oferecem mecanismos para definição das restrições de integridade.

Tipos de restrições de integridade:

- Integridade de domínio
- Integridade de vazio
- Integridade de chave de chave primária
- Integridade referencial (chave estrangeira)

4.2 Mapeamento do Modelo ER para Modelo Relacional

O Modelo ER (Entidade-Relacionamento) é um modelo de dados semântico que descreve a realidade independente dos aspectos de implementação.

O Modelo Relacional descreve a realidade a nível de SGBD relacional (tabela, atributos e relacionamentos implementados através de chaves estrangeiras).

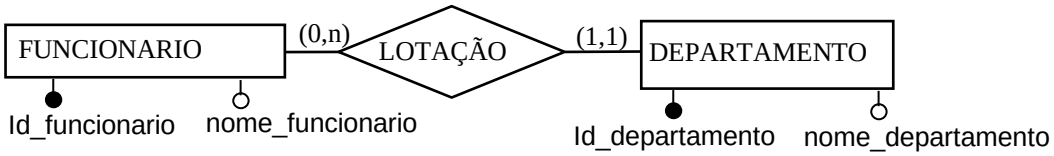
Sendo assim, é necessário regras de transformação do modelo ER para o modelo relacional.

Regras básicas:

1. Toda entidade vira uma tabela
2. Atributo identificador vira chave primária
3. Atributo simples vira colunas (campos)

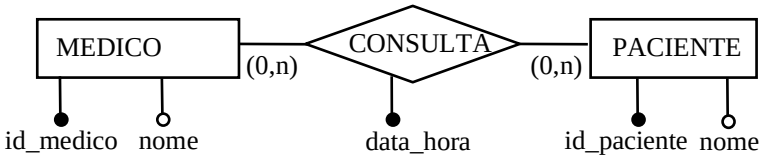
4.2.1 Implementação de Relacionamento 1:N

A alternativa preferida para implementação de relacionamento 1:n é a adição de coluna.



Departamento (id_departamento, nome_departamento)
Funcionario(id_funcionario, nome_funcionario, id_departamento)
id_departamento referencia departamento

4.2.3 Implementação de Relacionamento N:N



O relacionamento deve ser implementado através de uma tabela;

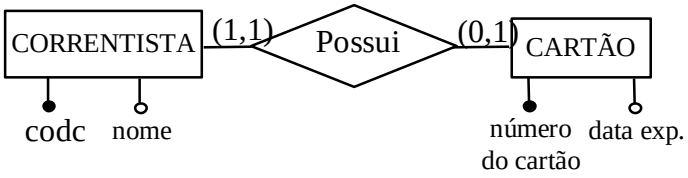
Medico(id_medico, nome);

Paciente(id_paciente, nome);

Consulta(id_medico, id_paciente, data_hora);
id_medico referencia medico
id_paciente referncia paciente

4.2.3 Implementação de relacionamento 1:1

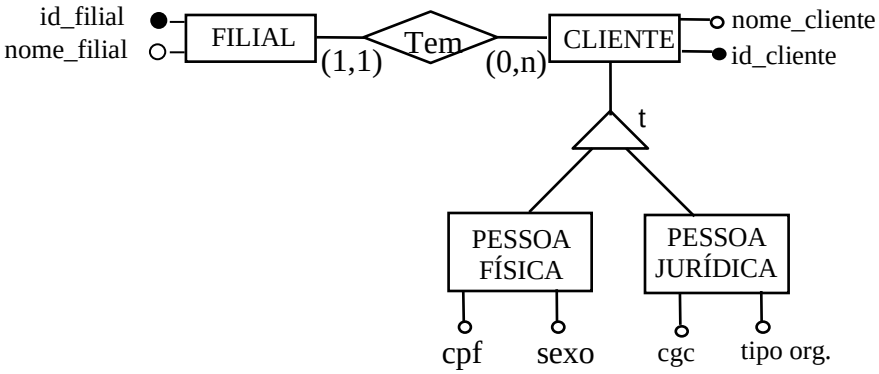
Os relacionamentos 1:1 são na maioria das vezes implementados através de adição de coluna, geralmente se escolhe uma das tabelas (relações) para receber a chave estrangeira (na maioria dos casos é a relação que é obrigatória no relacionamento).



Correntista(<u>codc</u>, nome)
Cartao(<u>num_car</u>, data, codc)
Codc referencia correntista

obs: a implementação utilizando-se uma única tabela geraria muitos campos vazios, caso houvesse muitos correntistas sem cartão de crédito;

4.2.4 Implementação de Generalização/Especialização



⇒ Alternativa 1

Filial (<u>id_filial</u>, nome, tipo);

Cliente (<u>id_cliente</u>, nome_cliente, id_filial);

id_filial referencia filial

Pessoa_Fisica (<u>id_cliente</u>, cpf,sexo);

id_cliente referencia cliente

Pessoa_Juridica (<u>id_cliente</u>, cgc,tipo_org);

id_cliente referencia cliente

obs: id_cliente é chave estrangeira nas tabelas Pessoa_Fisica e Pessoa_Juridica.

⇒ Alternativa 2:

Filial (id_filial, nome, tipo);

Cliente (id_cliente, nome_cliente, id_filial, cpf,sexo, cgc,tipo_org);

id_filial referencia filial

⇒ Alternativa 3

Filial (id_filial, nome, tipo);

Pessoa_Fisica (id_cliente, nome_cliente, id_filial , cpf,sexo); só pessoas físicas

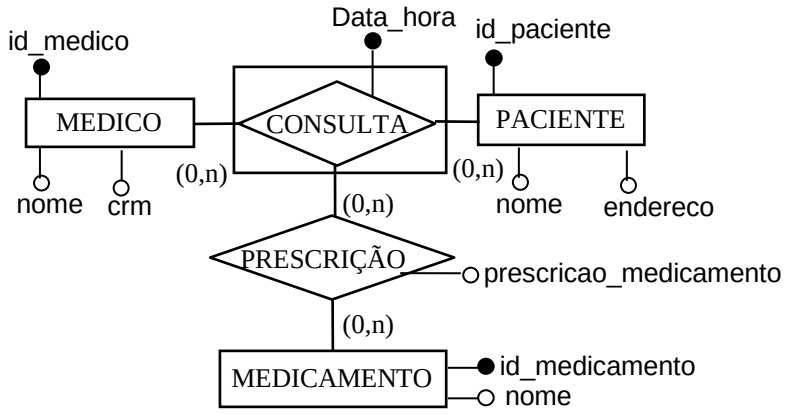
id_filial referencia filial

Pessoa_Juridica (id_cliente, nome_cliente, id_filial , cgc,tipo_org); só pessoa jurídica

id_filial referencia filial

obs: Só pode ser utilizada para generalização for total

4.2.5 Implementação de Entidade Associativa



Medico(id_medico, nome, crm);

Paciente(id_paciente, nome, endereco);

Consulta (id_medico, id_paciente, data_hora);

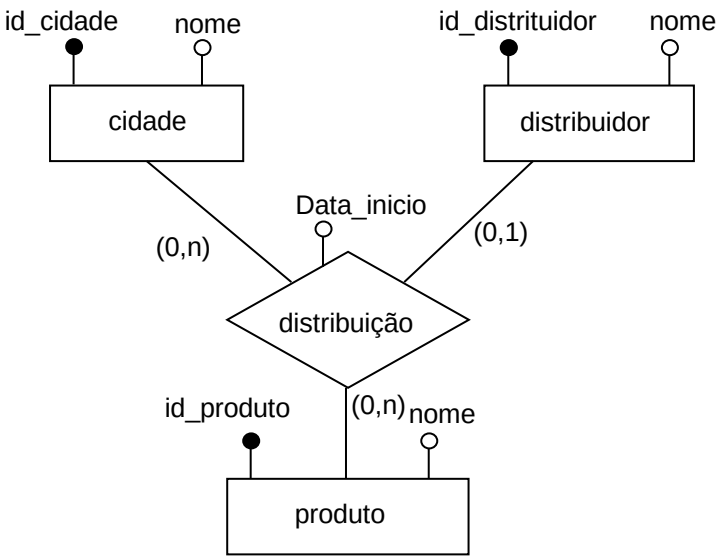
id_medico referencia medico

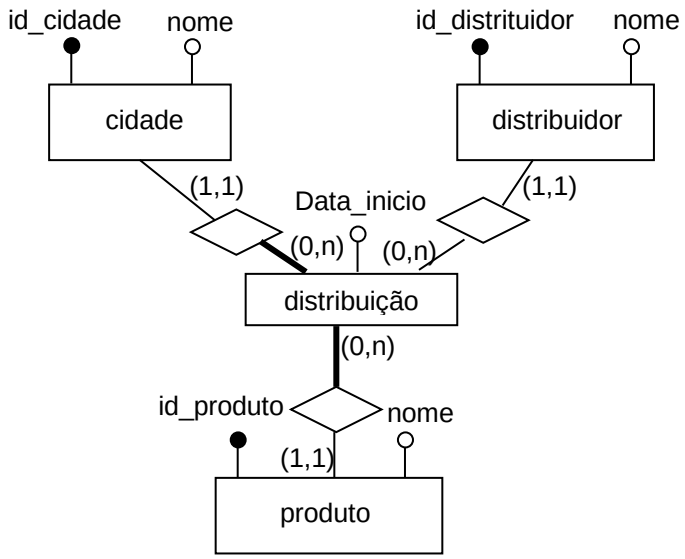
id_paciente referencia paciente

Medicamento(id_medicamento, nome);

Prescricao(id_medico, id_paciente, data_hora, id_medicamento, prescricao_medicamento);
id_medico,id_paciente,data_hora referencia consulta
id_medicamento referencia medicamento

4.2.6 Relacionamento de grau maior que dois





cidade (<u>id_cidade</u>, nome)
produto (<u>id_produto</u>,nome)
distribuidor (<u>id_distribuidor</u>,nome)
distribuicao (<u>id_cidade, id_produto</u>, id_distribuidor, data_inicio)
id_cidade referencia cidade
id_produto referencia produto
id_distribuidor referencia distribuidor

4.3 Engenharia Reversa e Normalização

Esquema de arquivo convencional ou documento -> engenharia reversa de arquivos -> modelo relacional -> engenharia reversa de BD relacional -> modelo ER.

Algumas empresas possuem sistemas legados que devem ser entendidos para melhorias ou manutenção. Sobre estes sistemas pode ser realizada a engenharia reversa e a normalização. Além dos sistemas legados, os documentos existentes na empresa também possuem informações importantes.

Normalização é o processo através do qual uma tabela relacional não normalizada é transformada em um conjunto de tabelas normalizadas, que representam da forma mais adequada a realidade modelada.

CodProj	Tipo	Descr	Emp					
			CodEmp	Nome	Cat	Sal	DataIni	TemAl
LSC001	Novo Desenv.	Sistema de Estoque	2146	João	A1	4	1/11/91	24
			3145	Silvio	A2	4	2/10/91	24
			6126	José	B1	9	3/10/92	18
			1214	Carlos	A2	4	4/10/92	18
			8191	Mário	A1	4	1/11/92	12
PAG02	Manutenção	Sistema de RH	8191	Mário	A1	4	1/05/93	12
			4112	João	A2	4	4/01/91	24
			6126	José	B1	9	1/11/92	12

Fonte: Livro Projeto de Banco de Dados - Heuser

A tabela acima poderia também ser escrita como:

Proj(codProj, tipo, descr, (codEmp, nome, cat, sal, dataIni, tempoAl))

A tabela acima é uma tabela não-normalizada, cuja chave primária é o código do projeto (sublinhado). A utilização de uma tabela única para representar estes projetos, em um banco de dados, traria alguns problemas: um empregado só pode ser incluído se estiver vinculado a um projeto, se o empregado tiver seu salário alterado, será preciso percorrer toda a tabela para realizar múltiplas alterações. Em vista disso, a utilização da normalização permite organizar as informações das tabelas de uma forma simples e relacional, evitando perda e repetição da informação e oferecendo, ainda, uma representação adequada para o que se deseja armazenar.

Para normalizar uma tabela, são utilizadas quatro regras básicas, chamadas de primeira, segunda, terceira e quarta formas normais, respectivamente, 1FN, 2FN, 3FN e 4FN.

1FN: Uma tabela está na primeira forma normal se não contém tabelas aninhadas (grupos repetidos ou atributos multivalorados).

Então, observando a tabela não normalizada identificamos 1 grupo repetido. (codEmp, nome, cat, sal, dataIni, tempoAl).

Assim, são criadas tantas tabelas quantos forem os grupos de elementos repetidos:

Passagem para a 1FN:

1. Criar uma tabela na 1FN com os dados da tabela não normalizada sem as tabelas aninhadas. A chave da tabela é a mesma da tabela não normalizada.

   Proj(codProj, tipo, descr)

2. Para cada tabela aninhada criar uma tabela com as seguintes colunas: a chave primária de cada uma das tabelas na qual a tabela em questão está aninhada e as colunas da tabela aninhada.

   ProjEmp(codProj, codEmp, nome, cat, sal, dataIni, tempoAl)

3. Definir na 1FN as chaves primárias das tabelas correspondentes as tabelas aninhadas.
   a) tomar a chave primária da tabela embutida original
   b) para a chave primária da tabela embutida, fazer a seguinte pergunta:
   Um valor da chave primária aparece associado a exatamente um ou a muitos valores da chave primária da tabela externa?

   - Um - a chave primária da tabela externa não faz parte da chave primária da tabela na 1FN.
   - Muitos - a chave primária da tabela externa faz parte da chave primária da tabela na 1FN.

   ProjEmp(codProj, codEmp, nome, cat, sal, dataIni, tempoAl)

1FN:

Proj(codProj, tipo, descr)
ProjEmp(codProj, codEmp, nome, cat, sal, dataIni, tempoAl)

A 2FN e a 3FN dependem de um outro conceito: dependência funcional.

A dependência funcional é definida da seguinte forma: Em uma tabela relacional diz-se que uma coluna B é dependente funcional de A (ou que a coluna A determina a coluna B) quando em todas as linhas da tabela, para cada valor de A que aparece na tabela , aparece o mesmo valor de B.

A dependência funcional pode ocorrer de duas formas:

- ***parcial***: a dependência funcional parcial ocorre quando uma coluna não chave depende apenas de parte de chave primária composta;

- ***transitiva***: quando um atributo ou conjunto de atributos A depende de outro atributo B que não pertence à chave primária, mas é dependente funcional desta.

2FN: Uma tabela encontra-se na 2FN, quando além de estar na 1FN, não contém dependências funcionais parciais.

Passagem para 2FN:

1. Copiar para a 2FN cada tabela que tenha chave primária simples ou que não tenha atributos não chave;
2. Para tabelas com chave composta e atributos não chave:
   2.1) Criar na 2FN uma tabela com as chaves primárias da tabela na 1FN
   2.2) Para cada atributo não chave fazer a seguinte pergunta: O atributo depende de toda chave?
      Sim - Copiar o atributo para a 2FN.
      Não - Criar, caso não exista, uma tabela na 2FN que contenha como chave primária a parte da chave à qual o atributo pertence. Copiar o atributo dependente para a tabela criada.

2FN:

A tabela Proj permanece como na 1FN (não tem chave composta);
A tabela ProjEmp possui três campos que dependem apenas de parte da chave, neste caso, nome, cat e sal dependem de codEmp, então gera-se uma outra tabela chamada Emp.

Proj(<u>codProj</u>, tipo, descr)
Emp(<u>codEmp,</u> nome, cat, sal)
ProjEmp(<u>codProj, codEmp,</u> dataIni, tempoAl)

Para a 3FN, deve-se analisar a dependência transitiva da chave:

Passagem para 3FN:

1. Copiar para a 3FN cada tabela que não contenha zero ou no máximo 1 atributo não chave.

2. Para tabelas com mais de um atributo não chave:

2.1) Criar uma tabela na 3FN com a chave primária em questão;

2.2) Para cada atributo não chave fazer a seguinte pergunta: O atributo depende de algum outro atributo não chave (dependência transitiva ou indireta)?
   Não - Copiar o atributo para a tabela na 3FN.
   Sim - Executar três passos:
      a. Criar, caso ainda não exista, uma tabela na 3FN que contenha como chave primária o atributo do qual há dependência indireta.
      b. Copiar o atributo dependente para a tabela criada.

c. O atributo do qual há dependência deve permanecer também na tabela criada no passo a.

3FN:

Proj(codProj, tipo, descr)
Emp(codEmp, nome, cat)
Categora(cat,sal)
ProjEmp(codProj, codEmp, dataIni, tempoAl)

4FN: Uma tabela está na 4FN quando além de estar na 3FN não contém dependências funcionais multivaloradas.

Outro exemplo:

Codproj	Codfunc	Codequip
100	1	1
100	2	1
100	3	1
100	1	2
100	2	2
100	3	2
101	4	1
101	4	2

Quando dois relacionamentos independentes A:B e A:C são misturados em uma mesma relação R(A,B,C) uma dependência funcional multivalorada pode acontecer.

O Codproj 100 determina múltiplas vezes um conjunto de valores para Codfunc.

Codproj ->> CodFunc

Codproj ->> CodEquip

Codproj	Codfunc
100	1
100	2
100	3

101	4

Codproj	CodEquip
100	1
100	2
101	1
101	2

5. Álgebra Relacional

A álgebra relacional é uma linguagem de consulta procedural. Possui as seguintes características:

- Descreve qualquer operação de consulta sobre relações;
- Linguagem orientada à manipulação de relações e não de registros;
- O resultado de uma consulta sobre uma ou mais relações gera uma relação;
- É base para o desenvolvimento de DMLs de mais alto nível

A álgebra relacional possui um conjunto de operações básicas. Essas serão explicadas utilizando as seguintes relações exemplos:

Agencia(codag, nomeag, cidadeag)

codag	nomeag	cidadeag
100	X	POA
450	Y	Gravataí
102	Z	POA

Cliente(codc, nomec, idadec)

codc	nomec	idadec
200	Pedro	30
201	Ana	28
202	Paulo	46

Conta(codag, numconta, codc, saldo)

codag	numconta	codc	saldo
102	389	202	2500
450	678	202	3700
100	987	201	6000

Emprestimo(codag, numemp, codc, quantia)

codag	numemp	codc	saldo
450	560	202	2500
450	561	200	3700

- Operações Fundamentais

A) SELEÇÃO

- Seleciona tuplas que satisfazem uma dada condição (predicado);
- Produz um subconjunto horizontal de uma relação;
- Notação:

σ <predicado> (<relação>)

- <predicado> permite o uso dos seguintes operadores de comparação: =, ≠, <, <=, >, >=

ex: Obter informações sobre todas as agencias de POA.

σ cidadeag = 'POA' (Agencia)

resultado:

100 X POA
102 Z POA

B) PROJEÇÃO

- Seleciona atributos de interesse
- Produz um subconjunto vertical de uma relação
- Notação:

π <Lista_atributos> (<relação>)

ex: Obter os códigos das agências de POA.

π codag (σ cidadeag = 'POA' (Agencia))

resultado:

100 102

C) PRODUTO CATESIANO

- Combinação de todas as tuplas de duas relações
- Utilizado quando necessita-se obter dados presentes em duas ou mais relações.
- Notação:

<relação1> X <relação2>

ex1: Obter o nome de todos os clientes que tem conta na agencia de código 450

* Expressão não otimizada

π nomec (σ codag = 450 ^ Cliente.codc = Conta.codc (Cliente X Conta))

etapas da executação:

❖ **relação resultante do produto cartesiano**

Cliente				Conta		
codc	nomec	idadec	codag	numconta	codc	saldo
200	Pedro	30	102	389	202	2500
200	Pedro	30	450	678	202	3700

200	Pedro	30	100	987	201	6000
201	Ana	28	102	389	202	2500
201	Ana	28	450	678	202	3700
201	Ana	28	100	987	201	6000
202	Paulo	46	102	389	202	2500
202	Paulo	46	450	678	202	3700
202	Paulo	46	100	987	201	6000

- **resultado da seleção aplicada a relação anterior**

Cliente Conta

codc	Nomec	idadec	codag	numconta	codc	saldo
202	Paulo	46	450	678	202	3700

- **resultado da consulta**

Paulo

Expressão Otimizada da consulta anterior

$$\pi_{\text{nomec}} (\sigma_{\text{Cliente.codc = Conta.codc}} (\pi_{\text{codc,nomec}} (\text{Cliente}) \times \pi_{\text{codc}} (\sigma_{\text{codag = 450}} (\text{Conta}))))$$

etapas da execução:

- **relação resultante da seleção e projeção sobre a relação Conta.**

codc
202

- **relação resultante da projeção sobre a relação Cliente**

codc	nomec
200	Pedro
201	Ana
202	Paulo

- **relação resultante do produto cartesiano**

Cliente Conta

codc	nomec	codc
200	Pedro	202
201	Ana	202
202	Paulo	202

❖ **resultado da consulta**

Paulo

D) UNIÃO

- Une as tuplas de duas relações que sejam compatíveis
- Notação:

<relação 1 > ∪ <relação2>

obs: operadores matemáticos (união, diferença e interseção) aplicam-se a duas relações ditas compatíveis, ou seja:

- relações com o mesmo grau (número de atributos)
- relações cujos domínios dos atributos são iguais, na mesma ordem de definição de colunas.

ex: Obter o código de todos os clientes da agencia 450

$\pi_{codc}(\sigma_{codag = 450}(Conta)) \cup \pi_{codc}(\sigma_{codag = 450}(Emprestimo))$

resultado: 200 202

E) DIFERENÇA

- Retorna as tuplas de uma relação1 cujos valores não estão presentes em uma relação2.
- Notação:

<relação1> - <relação2>

ex: obter o código dos clientes que não fizeram empréstimos

$\pi_{codc}(Cliente) - \pi_{codc}(Emprestimos)$

resultado: 201

F) INTERSECÇÃO

- Retorna as tuplas cujos valores de seus atributos sejam comuns às relações 1 e 2.
- Notação: <relação1> ∩ <relação2>

ex: obter o código de todos os clientes que possuem uma conta e um empréstimo

$\pi_{codc}(Conta) \cap \pi_{codc}(Emprestimos)$

resultado: 202

G) JUNÇÃO NATURAL (JOIN)

- Combinação dos operadores produto cartesiano e seleção (retorna as tuplas de um produto cartesiano que satisfazem uma dada condição).
- Assume uma junção por um ou mais atributos comuns sem repetir este atributo na relação resultante.
- Notação: <relação1> [x] <relação2>

ex: Obter o nome de todos os clientes que tem conta na agencia de código 450

$\pi_{nomec}(\pi_{codc,nomec}(\text{Cliente})\ [x]\ \pi_{codc}(\sigma_{codag = 450}(\text{Conta})))$

❖ **relação resultante do Join**

codc	nomec
202	Paulo

❖ **resultado da consulta:**

Paulo

H) DIVISÃO

- Operadores de divisão:
  Dividendo (relação R1 com grau m + n)
  Divisor (relação R2 com grau n)
  Quociente (relação resultante com grau m)
- Utilizada quando se deseja extrair de uma relação R1 uma determinada parte que possui as características (valores de atributos) da relação R2.
- Os atributos de grau n devem possuir o mesmo domínio.
- Notação: <relação1> ÷ <relação2>

ex: Obter o nome de todos os cliente que tem conta em todas as agencia de POA

$\pi_{nomec, codag}(\text{Cliente}\ [x]\ \text{Conta}) \div \pi_{codag}(\sigma_{nomeag = \text{'POA'}}(\text{Agencia}))$

❖ **relação resultante do dividendo**

nomec	codag
Paulo	102
Paulo	450
Ana	100

❖ **relação resultante do divisor**

codag
100
102

- **resultado da consulta : vazio (nenhuma tupla no resultado)**

BIBLIOGRAFIA

- DATE, C. J. Introdução a Sistemas de Bancos de Dados. Rio de Janeiro: Campus, 2000. 7ª edição.
- ULLMAN, J. A First Course in Database Systems. Upper Saddle River: Prentice Hall, 2002. 2ª edição.
- ELMASRI, R. & NAVATHE, S.B. Fundamentals of database systems. Redwood City: The Benjamin/Cummings, 2003.4ª edição.
- HEUSER, Carlos Alberto. Projeto de Bancos de Dados. Ed. Sagra-Luzzatto, 2001.5ª edição.
- KORTH, Henry F. e SILBERSCHATZ, Abraham. Sistema de Bancos de Dados. São Paulo: Makron Books, 1999. 3ª edição revisada.CHEN, Peter. Gerenciando banco de dados: a abordagem entidade-relacionamento para projeto lógico. McGraw-Hill. 1990.
- KORTH, H.F.; SILBERSCHATZ, A.; SUDARSHAN, S. Database Systems Concepts. (3rd Edition) McGraw Hill, Inc., 1997.
- NAVATHE, S.B. et al. Conceptual Database Design. Redwood City: The Benjamin/Cummings, 1992.
- BOWMAN, J.; EMERSON, S.; DARNOVSKY, M. The Practical SQL Handbook (3rd Edition) Addison-Wesley, 1996.
- KORTH, Henry F. e SILBERSCHATZ, Abraham. Sistema de Bancos de Dados. São Paulo: Makron Books, 1995. 2ª edição revisada.