

Apostila de Sistemas de Banco de Dados (2009)

Prof. Ms. Eduardo Rosalém Marcelino
ermarc@itelefonica.com.br

Bibliografias utilizadas:

ELMASRI, Ramez; NAVATHE, S.B., **Sistemas de Bancos de Dados**, 4º ed. São Paulo: Pearson, 2005.
DATE, C.J. **Introdução a Sistemas de Banco de Dados**. 7 ed. São Paulo: Campus, 2000.
MACHADO, Felipe N.R. **Projeto de Banco de Dados – Uma visão prática**. 9 ed. São Paulo: Érica, 2000.
MSDN, <http://msdn.microsoft.com/pt-br/default.aspx>
MUNARI, A.C.B., <http://www.pucrs.campus2.br/~jiani/bd/OpRelacional.pdf>

Índice:

Introdução a Banco de Dados	5
Informação X Dado	5
O que é um Banco de Dados	5
Por que Banco de Dados	6
Sistema Gerenciador de Banco de Dados (SGBD)	6
Usuários	8
Administrador de Banco de Dados (DBA)	8
Projetista de Banco de Dados	8
Usuários Finais	8
Analistas de Sistemas e Programadores de Aplicações	9
Vantagens e desvantagens do uso de um SGBD	9
Controle de Redundância	9
Compartilhamento de Dados	10
Restrição a Acesso não Autorizado	10
Representação de Relacionamentos Complexos entre Dados	10
Tolerância a Falhas	10
Quando não Utilizar um SGBD	10
Conceitos e Arquiteturas de um SGBD	11
Modelos de Dados	11
Modelo hierárquico	11
Modelo de Rede	13
Modelo Relacional	14
Esquemas e Instâncias	14
A Arquitetura Três Esquemas	15
Independência de Dados	16
As Linguagens para Manipulação de Dados	16
Os Módulos Componentes de um SGBD	16
Arquitetura SGBD Centralizada	17
Arquiteturas Cliente/Servidor	18
Classificação dos SGBDs	20
Modelagem de Dados Utilizando o Modelo Entidade Relacionamento (ER)	20
Entidades e Atributos	20
Valores Nulos (NULL)	21
Tipos Entidade, Conjunto de Valores, Atributo Chave e Domínio	21
Chave Primária e Chave Candidata	22
Tipos e Instâncias de Relacionamento	22
Grau de um Relacionamento	23
Outras Características de um Relacionamento	23
Relacionamentos como Atributos	23
Nomes de Papéis e Relacionamentos Recursivos	23
Restrições em Tipos Relacionamentos	24
Tipos Entidades Fracas	25
Diagrama Entidade Relacionamento	26
Exemplos de relacionamentos entre entidades:	28
Exemplo prático: a Viagem	29
Normalização	30
Primeira forma normal (1FN)	31
Dependência Funcional	32
Dependência Funcional Total (completa) e Parcial	33
Segunda forma normal (2FN)	33
Chave Estrangeira	35
Dependência Funcional Transitiva	36
Terceira Forma Normal (3FN)	36
Roteiro de aplicação da Normalização	38
Entidades Na Forma Normal Final (Normalizadas)	39
Operações relacionais e Álgebra relacional	39
Projeção (π)	40
Seleção ou Restrição (σ)	41
Produto Cartesiano (conjunto1 x conjunto2)	44
Diferença entre conjuntos: A – B	47

	3
União: $A \cup B$	47
Intersecção: $A \cap B$	48
Junção: $A \mid X \mid B$	48
Resumo	48
Exercícios	49
Respostas	50
Integridade Referencial	51

Índice de Figuras

Figura 1 - Um banco de dados	5
Figura 2 - Sistema de Banco de Dados	8
Figura 3 - O profissional de banco de dados tem alguma interação com a figura acima?	9
Figura 4 - Exemplo do Modelo de Rede	13
Figura 5 - A arquitetura de três-esquemas	15
Figura 6 - Módulos componentes de um SGBD e suas interações	17
Figura 7 - A arquitetura física centralizada	18
Figura 8 - A arquitetura lógica de duas camadas cliente/servidor.	18
Figura 9 - Arquitetura física cliente/servidor de duas camadas.	19
Figura 10 - Arquitetura lógica cliente/servidor de três camadas.	19
Figura 11 - Duas entidades, empregado <i>e1</i> e empresa <i>c1</i> , e seus atributos.	20
Figura 12 - Uma hierarquia de atributos compostos.	21
Figura 13 - Dois tipos entidade, EMPREGADO e EMPRESA, e algumas entidades-membro de cada um.	21
Figura 14 - Algumas instâncias do conjunto de relacionamento TRABALHA_PARA, que representa um tipo relacionamento TRABALHA_PARA entre EMPREGADO e DEPARTAMENTO.	22
Figura 15 - Algumas instâncias de relacionamento do conjunto de relacionamento ternário FORNECE.	23
Figura 16 - Um relacionamento recursivo SUPERVISAO entre EMPREGADO, no papel de <i>supervisor</i> (1), e EMPREGADO, no papel de <i>subordinado</i> (2).	24
Figura 17 - Um relacionamento GERENCIA 1:1.	24
Figura 18 - Exemplo de relacionamento M:N	25
Figura 19 - Relacionamento com uma Entidade Fraca (Dependente)	26
Figura 20 - Resumo da notação para diagramas ER.	27
Figura 21 - Representação de relacionamentos	28
Figura 22 - Relacionamentos para “a viagem” já com a cardinalidade informada.	29
Figura 23- Um formulário de PEDIDO	30
Figura 24 - Entidades após aplicara 1FN	31
Figura 25 - Entidades após a 1FN	32
Figura 26 - Entidades ITEM-DO-PEDIDO e PRODUTO após a 2FN	34
Figura 27 - Entidades após a 3FN	37
Figura 28 - Tabelas utilizadas nos exemplos de álgebra relacional.	40
Figura 29 - Resultado da projeção	41
Figura 30 - Resultado da Seleção	42
Figura 31 - Utilizando projeção (π) em conjunto com seleção (σ)	43
Figura 32 - Produto Cartesiano entre a tabela A e a tabela B	44
Figura 33 - $\pi_{NmFunc, DtAdm, VrSalário} (\sigma_{funcionário.CdCargo = cargo.CdCargo} (funcionário \times cargo))$	46

Introdução a Banco de Dados

Informação X Dado

Qual a diferença dentre Dado e Informação?

A Informação acrescenta algo ao conhecimento da realidade a ser analisada. Por exemplo, a dosagem que um paciente precisa receber de um determinado remédio, é uma informação.

O Dado é uma representação, um registro de uma informação. Este dado pode ser registrado fisicamente através de um papel (receita médica), um disco de computador, um impulso elétrico, etc.

Podemos concluir então, que em um sistema de informações, estão contidas todas as informações necessárias ao objetivo do sistema (ex: manter a vida do paciente). Os dados originados dessas informações serão processados pelo sistema criado. Por definição, um computador não processa informações, mas sim, DADOS.

O que é um Banco de Dados



Figura 1 - Um banco de dados

Definições de banco de dados segundo alguns autores:

"Em essência, um sistema de bancos de dados é apenas um sistema computadorizado de armazenamento de registros." (DATE, 2000)

"Um banco de dados é uma coleção de dados relacionados. Os dados são fatos que podem ser gravados que possuem um significado implícito." (ELMASRI, NAVATHE, 2005)

"É uma coleção de fatos registrados que refletem o estado de certos aspectos de interesse do mundo real. A todo momento o conteúdo do banco de dados representa uma visão instantânea do estado do mundo real. Cada mudança em algum item do banco de dados reflete uma mudança ocorrida na realidade" (MACHADO, ABREU, 1996)

Porém, o significado do termo “banco de dados” é mais restrito que simplesmente a definição dada acima. Um banco de dados possui as seguintes propriedades:

- um banco de dados representa algum aspecto do mundo real, o qual é chamado de “mini-mundo” ; qualquer alteração efetuada no mini-mundo é automaticamente refletida no banco de dados.
- um banco de dados é uma coleção lógica coerente de dados com um significado inerente; uma disposição desordenada dos dados não pode ser referenciada como 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;

Por que Banco de Dados

- Densidade: não há necessidade de arquivos em papel, possivelmente volumosos.
- Velocidade: a máquina pode obter e atualizar dados com rapidez muito maior que o ser humano. Em particular, consultas instantâneas podem ser respondidas com rapidez sem qualquer necessidade de pesquisas manuais ou visuais demoradas.
- Menor trabalho monótono: grande parte do tédio de manter arquivos à mão é eliminada. As tarefas mecânicas são sempre feitas com melhor qualidade por máquinas.
- Atualidade: informações precisas e atualizadas estão disponíveis a qualquer momento sob consulta.

Todas as vantagens acima se aplicam com mais intensidade em um ambiente multiusuário, no qual o banco de dados provavelmente será muito maior e mais complexo que no caso do ambiente de usuário único, além de proporcionar à empresa o controle centralizado de seus dados.

Sistema Gerenciador de Banco de Dados (SGBD)

“É uma coleção de programas que permite aos usuários criar e manter um banco de dados. O SGBD é, portanto, um sistema de software de propósito geral que facilita os processos de definição, construção, manipulação e compartilhamento de bancos de dados entre vários usuários e aplicações.” (ELMASRI, NAVATHE, 2005).

- A definição de um banco de dados implica especificar os tipos de dados, as estruturas e as restrições para os dados a serem armazenados em um banco de dados.
- A construção de um banco de dados é o processo de armazenar os dados em alguma mídia apropriada controlada pelo SGBD.
- A manipulação inclui algumas funções, como pesquisas em bancos de dados para recuperar um dado específico, atualização para refletir as mudanças no minimundo e gerar os relatórios dos dados.
- O compartilhamento permite aos múltiplos usuários e programas acessar, de forma concorrente, o banco de dados.

Outras funções importantes de um SGBD são a proteção e a manutenção do banco de dados. A proteção inclui a proteção do sistema contra o mau funcionamento ou falhas no hardware ou software, e segurança contra acessos não autorizados ou maliciosos. Um banco de dados típico pode ter um ciclo de vida de muitos anos, então, eles devem ser capazes de manter um sistema de banco de dados que permita a evolução dos requisitos que se alteram ao longo do tempo.

Características do emprego de bancos de dados

Um número significativo de características distingue a abordagem que utiliza o banco de dados daquela tradicional que usa a programação e arquivos. No tradicional processamento de arquivos, cada usuário define e implementa os arquivos necessários para uma aplicação específica, como parte da programação da aplicação. Por exemplo, um usuário, a secretária de notas, pode manter um arquivo para os alunos e suas notas. Os programas para imprimir um histórico do aluno e colocar novas notas no arquivo são implementados como parte da aplicação. Um segundo usuário, o departamento de contabilidade, pode controlar os dados relacionados às mensalidades e pagamentos dos alunos. Apesar de ambos os usuários estarem interessados nos dados sobre os estudantes, cada um mantém suas informações em arquivos separados – e os programas que manipulam esses arquivos – porque cada um deles precisa de alguns dados não disponíveis nos arquivos

do outro. Essas redundâncias, na definição e armazenamento dos dados, resultam em um espaço de armazenamento desperdiçado e em esforços redundantes para manter os dados comuns atualizados.

Na abordagem utilizando um banco de dados, um único repositório de dados é definido uma única vez, mantido e então acessado por vários usuários. As principais características da abordagem de um banco de dados versus a abordagem de processamento DCE arquivos são as seguintes:

Natureza autodescritiva do sistema de banco de dados.

Uma característica importante da abordagem Banco de Dados é que o SGBD mantém não somente os dados mas também a forma como os mesmos são armazenados, contendo uma descrição completa do banco de dados. Estas informações são armazenadas no catálogo do SGBD, o qual contém informações como a estrutura de cada arquivo, o tipo e o formato de armazenamento de cada tipo de dado, restrições, etc. A informação armazenada no catálogo é chamada de "Meta Dados". No processamento tradicional de arquivos, o programa que irá manipular os dados deve conter este tipo de informação, ficando limitado a manipular as informações que o mesmo conhece. Utilizando a abordagem banco de dados, a aplicação pode manipular diversas bases de dados diferentes.

Isolamento entre os programas e os dados, e a abstração dos dados.

No processamento tradicional de arquivos, a estrutura dos dados está incorporada ao programa de acesso. Desta forma, qualquer alteração na estrutura de arquivos implica na alteração no código fonte de todos os programas. Já na abordagem banco de dados, a estrutura é alterada apenas no catálogo, não alterando os programas.

O SGBD deve fornecer ao usuário uma "representação conceitual" dos dados, sem fornecer muitos detalhes de como as informações são armazenadas. Um "modelo de dados" é uma abstração de dados que é utilizada para fornecer esta representação conceitual utilizando conceitos lógicos como objetos, suas propriedades e seus relacionamentos. A estrutura detalhada e a organização de cada arquivo são descritas no catálogo.

Suporte para as múltiplas visões dos dados.

Como um conjunto de informações pode ser utilizada por um conjunto diferenciado de usuários, é importante que estes usuários possam ter "visões" diferentes da base de dados. Uma "visão" é definida como um subconjunto de uma base de dados, formando deste modo, um conjunto "virtual" de informações.

O SGBD deve fornecer ao usuário uma "representação conceitual" dos dados, sem fornecer muitos detalhes de como as informações são armazenadas. Um "modelo de dados" é uma abstração de dados que é utilizada para fornecer esta representação conceitual utilizando conceitos lógicos como objetos, suas propriedades e seus relacionamentos. A estrutura detalhada e a organização de cada arquivo são descritas no catálogo.

Compartilhamento de dados e processamento de transações de multiusuários.

Um SGBD multiusuário, como o próprio nome implica, deve permitir que diversos usuários acessem o banco de dados ao mesmo tempo. Isso é essencial se os dados para as várias aplicações estão integrados e mantidos em um único banco de dados. O SGBD deve incluir um software de **controle de concorrência** para garantir que muitos usuários, ao tentar atualizar o mesmo dado, o façam de um modo controlado, para assegurar que os resultados das atualizações sejam corretos. Uma regra fundamental do software do SGBD multiusuário é garantir que as transações concorrentes operem corretamente.

O conceito de **transação** tornou-se fundamental para muitas aplicações de bancos de dados. Uma transação é um programa em execução ou processo que inclui um ou mais acessos ao banco de dados, como a leitura ou a atualização de registros. Cada transação deve executar um acesso logicamente correto ao banco de dados, se executado sem a interferência de outras transações. O SGBD deve garantir várias propriedades da transação.

A propriedade de **isolamento** garante que cada transação possa ser feita de forma isolada de outras transações. Mesmo centenas de transações podem ser executadas simultaneamente.

A propriedade de **atomicidade** garante que todas as operações em um banco de dados, em uma transação, sejam executadas ou nenhuma delas o seja.

As características precedentes são muito importantes para distinguir um SGBD de um software tradicional de processamento de arquivos.

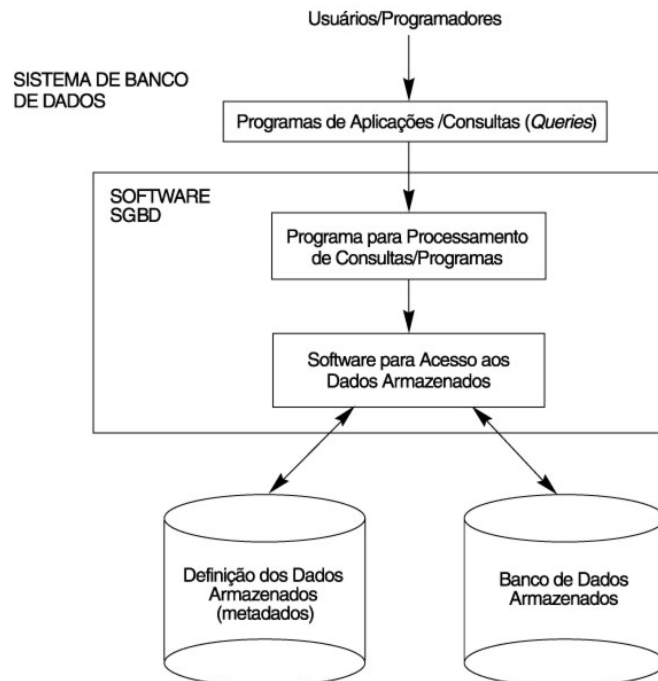


Figura 2 - Sistema de Banco de Dados

Usuários

Para um grande banco de dados, existe um grande número de pessoas envolvidas, desde o projeto, uso até manutenção.

Administrador de Banco de Dados (DBA)

Em um ambiente de banco de dados, o recurso primário é o banco de dados por si só e o recurso secundário o SGBD e os softwares relacionados. A administração destes recursos cabe ao Administrador de Banco de Dados, o qual é responsável pela autorização de acesso ao banco de dados e pela coordenação e monitoração de seu uso.

Projetista de Banco de Dados

O Projetista de Banco de Dados é responsável pela identificação dos dados que devem ser armazenados no banco de dados, escolhendo a estrutura correta para representar e armazenar dados. Muitas vezes, os projetistas de banco de dados atuam como "staff" do DBA, assumindo outras responsabilidades após a construção do banco de dados. É função do projetista também avaliar as necessidades de cada grupo de usuários para definir as visões que serão necessárias, integrando-as, fazendo com que o banco de dados seja capaz de atender a todas as necessidades dos usuários.

Usuários Finais

Há várias categorias de usuários finais que são as pessoas que acessam o banco de dados para consultas, atualizações e relatórios:

- Usuários casuais: acessam o banco de dados casualmente, mas que podem necessitar de diferentes informações a cada acesso; utilizam sofisticadas linguagens de consulta para especificar suas necessidades;
- Usuários novatos ou paramétricos: utilizam porções pré-definidas do banco de dados, utilizando consultas preestabelecidas que já foram exaustivamente testadas;

- Usuários sofisticados: são usuários que estão familiarizados com o SGBD e realizam consultas complexas.
- Usuários autônomos (stand-alone): mantêm um banco de dados pessoal por meio do uso de pacotes de programas prontos que possuem interfaces gráficas ou programas baseados em menus fáceis de usar. Um exemplo disso é o usuário de um pacote para cálculo de impostos que armazena seus dados financeiros pessoais para o pagamento de impostos.

Analistas de Sistemas e Programadores de Aplicações

Os analistas determinam os requisitos dos usuários finais e desenvolvem especificações para transações que atendam estes requisitos, e os programadores implementam estas especificações como programas, testando, depurando, documentando e dando manutenção no mesmo. É importante que, tanto analistas quanto programadores, estejam a par dos recursos oferecidos pelo SGBD.

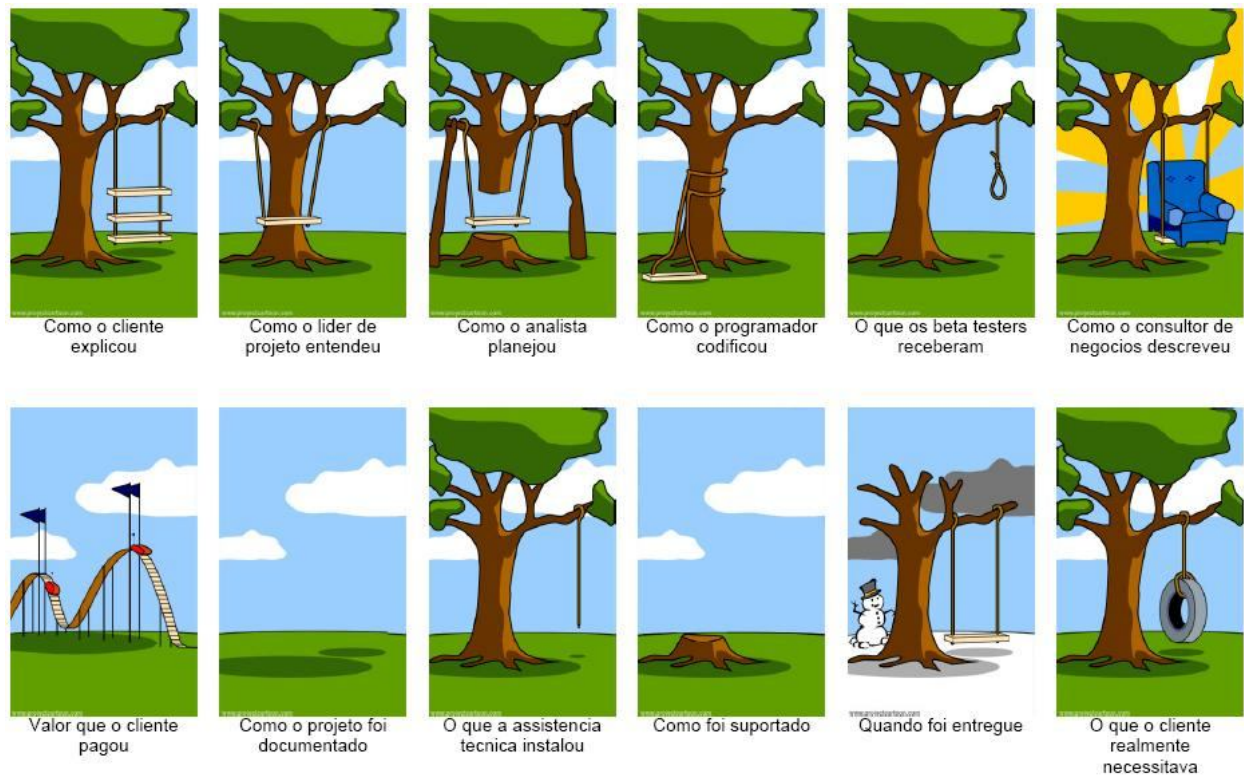


Figura 3 - O profissional de banco de dados tem alguma interação com a figura acima?

Vantagens e desvantagens do uso de um SGBD

Controle de Redundância

No processamento tradicional de arquivos, cada grupo de usuários deve manter seu próprio conjunto de arquivos e dados. Desta forma, acaba ocorrendo redundâncias que prejudicam o sistema com problemas como:

- Toda vez que for necessário atualizar um arquivo de um grupo, então todos os grupos devem ser atualizados para manter a integridade dos dados no ambiente como um todo;
- A redundância desnecessária de dados levam ao armazenamento excessivo de informações, ocupando espaço que poderia estar sendo utilizado com outras informações.

Compartilhamento de Dados

Um SGBD multi-usuário deve permitir que múltiplos usuários acessem o banco de dados ao mesmo tempo. Este fator é essencial para que múltiplas aplicações integradas possam acessar o banco.

O SGBD multi-usuário deve manter o controle de concorrência para assegurar que o resultado de atualizações sejam corretos. Um banco de dados multi-usuários deve fornecer recursos para a construção de múltiplas visões.

Restrição a Acesso não Autorizado

Um SGBD deve fornecer um subsistema de autorização e segurança, o qual é utilizado pelo DBA para criar “contas” e especificar as restrições destas contas; o controle de restrições se aplica tanto ao acesso aos dados quanto ao uso de softwares inerentes ao SGBD.

Representação de Relacionamentos Complexos entre Dados

Um banco de dados pode incluir uma variedade de dados que estão interrelacionados de várias formas. Um SGBD deve fornecer recursos para se representar uma grande variedade de relacionamentos entre os dados, bem como, recuperar e atualizar os dados de maneira prática e eficiente.

Tolerância a Falhas

Um SGBD deve fornecer recursos para recuperação de falhas tanto de software quanto de hardware.

Quando não Utilizar um SGBD

Em algumas situações, o uso de um SGBD pode representar uma carga desnecessária aos custos quando comparado à abordagem processamento tradicional de arquivos como, por exemplo:

- Alto investimento inicial na compra de software e hardware adicionais;
- Generalidade que um SGBD fornece na definição e processamento de dados;
- Sobrecarga na provisão de controle de segurança, controle de concorrência, recuperação e integração de funções.

Problemas adicionais podem surgir caso os projetistas de banco de dados ou os administradores de banco de dados não elaborem os projetos corretamente ou se as aplicações não são implementadas de forma apropriada. Se o DBA não administrar o banco de dados de forma apropriada, tanto a segurança quanto a integridade dos sistemas podem ser comprometidas. A sobrecarga causada pelo uso de um SGBD e a má administração justificam a utilização da abordagem processamento tradicional de arquivos em casos como:

- O banco de dados e as aplicações são simples, bem definidas e não se espera mudanças no projeto;
- A necessidade de processamento em tempo real de certas aplicações, que são terrivelmente prejudicadas pela sobrecarga causada pelo uso de um SGBD;
- Não haverá múltiplo acesso ao banco de dados.

Conceitos e Arquiteturas de um SGBD

Modelos de Dados

Uma das principais características da abordagem banco de dados, é que a mesma fornece alguns níveis de abstração de dados omitindo ao usuário final, detalhes de como estes dados são armazenados. Um “modelo de dados” é um conjunto de conceitos que podem ser utilizados para descrever a estrutura “lógica” e “física” de um banco de dados. Por “estrutura” podemos compreender o tipo dos dados, os relacionamentos e as restrições que podem recair sobre os dados.

Os modelos de dados podem ser basicamente de dois tipos:

- alto nível: ou modelo de dados conceitual, que fornece uma visão mais próxima do modo como os usuários visualizam os dados realmente;
- baixo nível: ou modelo de dados físico, que fornece uma visão mais detalhada do modo como os dados estão realmente armazenados no computador.

Modelo hierárquico

Uma **base de dados hierárquica** é um tipo de sistema de gerenciamento de banco de dados que conecta *registros* numa estrutura de dados em árvore através de *ligações* de tal modo que cada tipo de registro tenha apenas um possuidor. A base de dados se baseia em um Modelo de Entidades e Relacionamentos: cada registro é uma coleção de *atributos* (campos), cada um dos quais contendo somente uma informação; uma ligação é a associação entre dois registros. Por exemplo: em uma dada base de dados comercial, uma encomenda (i.e. registro) é possuída por um único cliente.

As estruturas hierárquicas foram muito usadas nos primeiros sistemas de gestão de bases de dados mainframe. No entanto, devido às suas restrições, é freqüente que não possam ser usados para relacionar estruturas que existem no mundo real. As relações hierárquicas entre diferentes tipos de dados podem tornar muito fácil a resposta a algumas questões, mas muito difícil a resposta a outras. Se a relação um-para-muitos for violada (por exemplo, um paciente pode ter mais do que um médico) então a hierarquia transforma-se numa rede.

Os registros são organizados como *árvores com raiz*. Cada árvore tem uma raiz, que é um pseudonó (cada nó é um registro, mas a raiz tem apenas a função de ser uma origem comum). Cada árvore com raiz é referida como uma árvore de base de dados; a base de dados hierárquica é uma coleção de árvores da base de dados (que formam uma floresta). Para ser mais preciso sobre o que significa uma árvore com raiz:

1. Não podem existir ciclos entre os nós (registros);
2. Ligações formadas na árvore devem ser tais que somente retratem relações um-para-um ou um-para-muitos entre um pai e um filho.

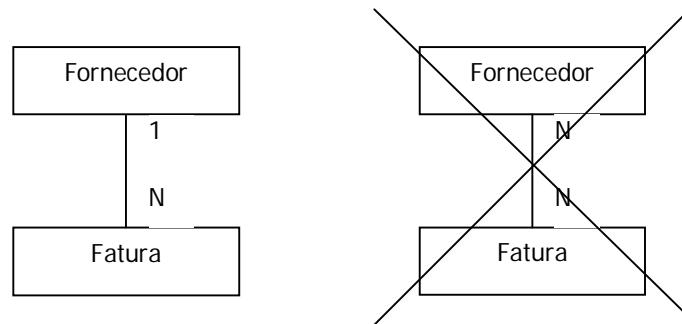
Assim, o conteúdo de um registro particular pode ter que ser replicado em vários locais diferentes. A réplica de registro possui duas grandes desvantagens: pode causar inconsistência de dados quando houver atualização, e o desperdício de espaço é inevitável.

Usa-se um diagrama de estrutura de árvore para apresentar o esquema para uma base de dados hierárquica. Consiste de dois componentes básicos: caixas (que correspondem ao tipo registro) e linhas (que correspondem às ligações). Seu propósito é especificar a estrutura lógica geral da base de dados.

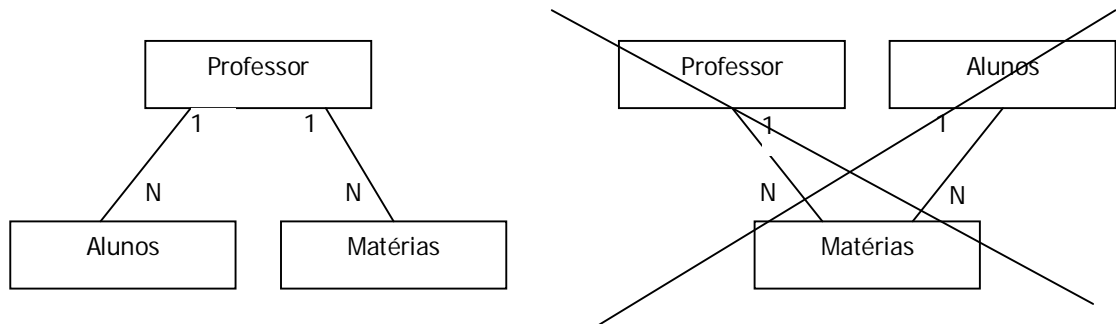
Os primeiros SGBD lançados no mercado foram os do tipo HIERÁRQUICO. Nessa categoria, o SOFTWARE de maior aceitação, no mercado brasileiro, foi o IMS da IBM.

Limitações dos modelos HIERÁRQUICOS

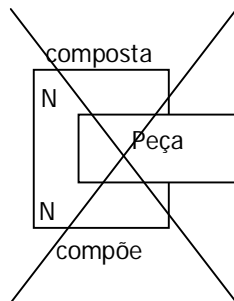
- a. Relacionamentos, no máximo, de grau "1: N". O processo de implementação dos relacionamentos do tipo N:N, ocasiona alto grau de redundância e / ou torna-se ineficaz.



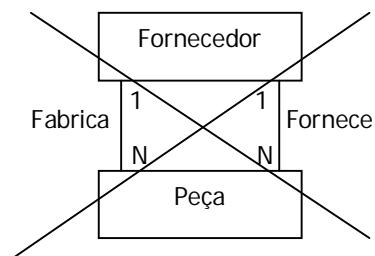
- b. O Banco de dados só pode ter um SEGUIMENTO RAIZ e cada SEGMENTO FILHO pode ligar-se a um único SEGUIMENTO PAI.



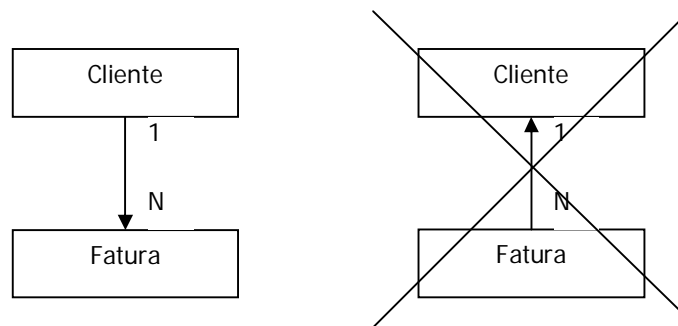
- c. Não implementam AUTO- RELACIONAMENTOS.



- d. Não permitem dupla ligação entre duas entidades:



- e. Estruturas percorridas somente no sentido de cima para baixo. No exemplo abaixo, poderia se facilmente listar "FATURA" a partir da entidade "CLIENTE", porém, o caminho inverso não seria possível



Modelo de Rede

O modelo em redes surgiu como uma extensão ao modelo hierárquico, eliminando o conceito de hierarquia e permitindo que um mesmo registro estivesse envolvido em várias associações. No modelo em rede, os registros são organizados em grafos onde aparece um único tipo de associação (set) que define uma relação 1:N entre 2 tipos de registros: proprietário e membro. Desta maneira, dados dois relacionamentos 1:N entre os registros A e D e entre os registros C e D é possível construir um relacionamento M:N entre A e D.

O gerenciador Data Base Task Group (DBTG) da CODASYL (Committee on Data Systems and Languages) estabeleceu uma norma para este modelo de banco de dados, com linguagem própria para definição e manipulação de dados. Os dados tinham uma forma limitada de independência física. A única garantia era que o sistema deveria recuperar os dados para as aplicações como se eles estivessem armazenados na maneira indicada nos esquemas. Os geradores de relatórios da CODASYL também definiram sintaxes para dois aspectos chaves dos sistemas gerenciadores de dados: concorrência e segurança.

O mecanismo de segurança fornecia uma facilidade na qual parte do banco de dados (ou área) pudesse ser bloqueada para prevenir acessos simultâneos, quando necessário. A sintaxe da segurança permitia que uma senha fosse associada a cada objeto descrito no esquema.

Ao contrário do Modelo Hierárquico, em que qualquer acesso aos dados passa pela raiz, o modelo em rede possibilita acesso a qualquer nó da rede sem passar pela raiz. No Modelo em Rede o sistema comercial mais divulgado é o CAIDMS da *Computer Associates*. O diagrama para representar os conceitos do modelo em redes consiste em dois componentes básicos: Caixas, que correspondem aos registros e Linhas, que correspondem às associações.

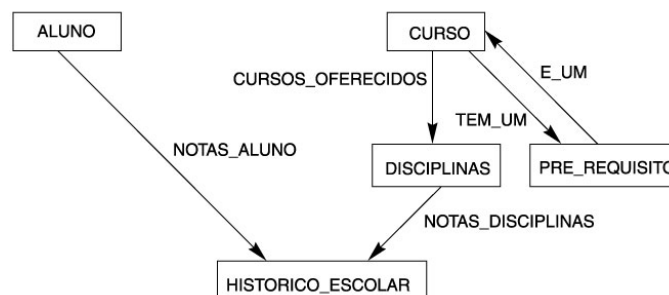


Figura 4 - Exemplo do Modelo de Rede

Tanto o Modelo Hierárquico como o de Redes são orientados a registros, isto é, qualquer acesso à base de dados – inserção, consulta, alteração ou remoção – é feito em um registro de cada vez. Como já foi mencionado, a organização do Modelo em Redes é semelhante a do Modelo Hierárquico, mas com a diferença de que cada registro filho pode ser ligado a mais de um registro pai, criando conexões bastante complexas e são bastante utilizados em sistemas para computadores de grande porte. Sendo que esse modelo é composto de uma estrutura mais completa, possui as propriedades básicas de registros, conjuntos e ocorrências, e utiliza a linguagem de definição de BD (DDL) e a linguagem de manipulação de dados (DML),

além de permitir evolução mais eficiente do modelo. A estrutura é formada de entidade (registros), atributos (itens de dados), tipo de registro e ocorrência do registro. Tanto o modelo hierárquico quanto o de rede são chamados de sistemas de navegação, pois as aplicações devem ser construídas para atravessar um conjunto de registros interligados previamente.

Apesar das características técnicas positivas já evidenciadas, os modelos Rede são pouco flexíveis no que se refere a alterações nas estruturas de dados, sua cultura é de difícil assimilação pelo usuário final e até mesmo por técnicos. Além disso, esses SGBD apresentam alguns problemas de desempenho e na manutenção de ponteiros. Em função dessas dificuldades e do salto tecnológico que experimentamos a partir da popularização dos microcomputadores, os SGBD Rede cederam espaço para os modelos Relacionais, que hoje constituem-se no padrão de mercado.

Modelo Relacional

O **modelo relacional** para gerência de bancos de dados (SGBD) é um modelo de dados baseado em lógica e na teoria de conjuntos. Historicamente ele é o sucessor do modelo hierárquico e do modelo em rede. Estas arquiteturas antigas são até hoje utilizadas em alguns data centers com alto volume de dados, onde a migração é inviabilizada pelo custo que ela demandaria; existem ainda os novos modelos baseados em orientação ao objeto, que na maior parte das vezes são encontrados como kits de construção de SGBD, ao invés de um SGBD propriamente dito.

O modelo relacional foi o primeiro modelo de banco de dados formal. Somente depois seus antecessores, os bancos de dados hierárquicos e em rede, passaram a ser também descritos em linguagem formal. O modelo relacional foi inventado pelo Dr. Codd e subsequentemente mantido e aprimorado por Chris Date e Hugh Darwen como um modelo geral de dados. No Terceiro Manifesto (1995) eles mostraram como o modelo relacional pode ser estendido com características de orientação a objeto sem comprometer os seus princípios fundamentais.

A linguagem padrão para os bancos de dados relacionais, SQL, é apenas vagamente remanescente do modelo matemático. Atualmente ela é adotada, apesar de suas restrições, porque ela é antiga e muito mais popular que qualquer outra linguagem de banco de dados.

A principal proposição do modelo relacional é que todos os dados são representados como relações matemáticas, isto é, um subconjunto do produto Cartesiano de n conjuntos. No modelo matemático (diferentemente do SQL), a análise dos dados é feita em uma lógica de predicados de dois valores (ou seja, sem o valor nulo); isto significa que existem dois possíveis valores para uma proposição: verdadeira ou falsa. Os dados são tratados pelo cálculo relacional ou álgebra relacional.

O modelo relacional permite ao projetista criar um modelo lógico consistente da informação a ser armazenada. Este modelo lógico pode ser refinado através de um processo de normalização. Um banco de dados construído puramente baseado no modelo relacional estará inteiramente normalizado. O plano de acesso, outras implementações e detalhes de operação são tratados pelo sistema SGBD, e não devem ser refletidos no modelo lógico. Isto se contrapõe à prática comum para SGBD SQL nos quais o ajuste de desempenho frequentemente requer mudanças no modelo lógico.

Os blocos básicos do modelo relacional são o domínio, ou tipo de dado. Uma tupla é um conjunto de atributos que são ordenados em pares de domínio e valor. Uma variável relacional (relvar) é um conjunto de pares ordenados de domínio e nome que serve como um cabeçalho para uma relação. Uma relação é um conjunto desordenado de tuplas. Apesar destes conceitos matemáticos, eles correspondem basicamente aos conceitos tradicionais dos bancos de dados. Uma relação é similar ao conceito de *tabela* e uma tupla é similar ao conceito de *linha*.

Esquemas e Instâncias

Em qualquer modelo de dados utilizado, é importante distinguir a “descrição” do banco de dados do “banco de dados” por si próprio. A descrição de um banco de dados é chamada de **esquema de um banco de dados** e é especificada durante o projeto do banco de dados. Geralmente, poucas mudanças ocorrem no esquema do banco de dados.

Os dados armazenados em um banco de dados em um determinado instante do tempo formam um conjunto chamado de **instância do banco de dados**. A instância altera toda vez que uma alteração no banco de dados é feita.

O SGBD é responsável por garantir que toda instância do banco de dados satisfaça ao esquema do banco de dados, respeitando sua estrutura e suas restrições. O esquema de um banco de dados também pode ser chamado de “intenção” de um banco de dados e a instância de “extensão” de um banco de dados.

A Arquitetura Três Esquemas

A principal meta da arquitetura “três esquemas” (figura 2) é separar as aplicações do usuário do banco de dados físico. Os esquemas podem ser definidos como:

- **Nível interno:** ou esquema interno, o qual descreve a estrutura de armazenamento físico do banco de dados; utiliza um modelo de dados e descreve detalhadamente os dados armazenados e os caminhos de acesso ao banco de dados;
- **Nível conceitual:** ou esquema conceitual, o qual descreve a estrutura do banco de dados para a comunidade de usuários. O esquema conceitual oculta os detalhes das estruturas de armazenamento físico e se concentra na descrição de entidades, tipos de dados, conexões, operações de usuários e restrições. Geralmente, um modelo de dados representacional é usado para descrever o esquema conceitual quando o sistema de banco de dados for implementado;
- **Nível externo:** ou esquema de visão, o qual descreve as visões do banco de dados para um grupo de usuários; cada visão descreve quais porções do banco de dados um grupo de usuários terá acesso.

Observe que os três esquemas são apenas descrições dos dados; na verdade, o dado que existe de fato está no nível físico. A maioria dos SGBDs não separa os três níveis completamente, mas suporta a arquitetura de três esquemas de alguma forma. Alguns SGBDs incluem detalhes do nível físico no esquema conceitual.

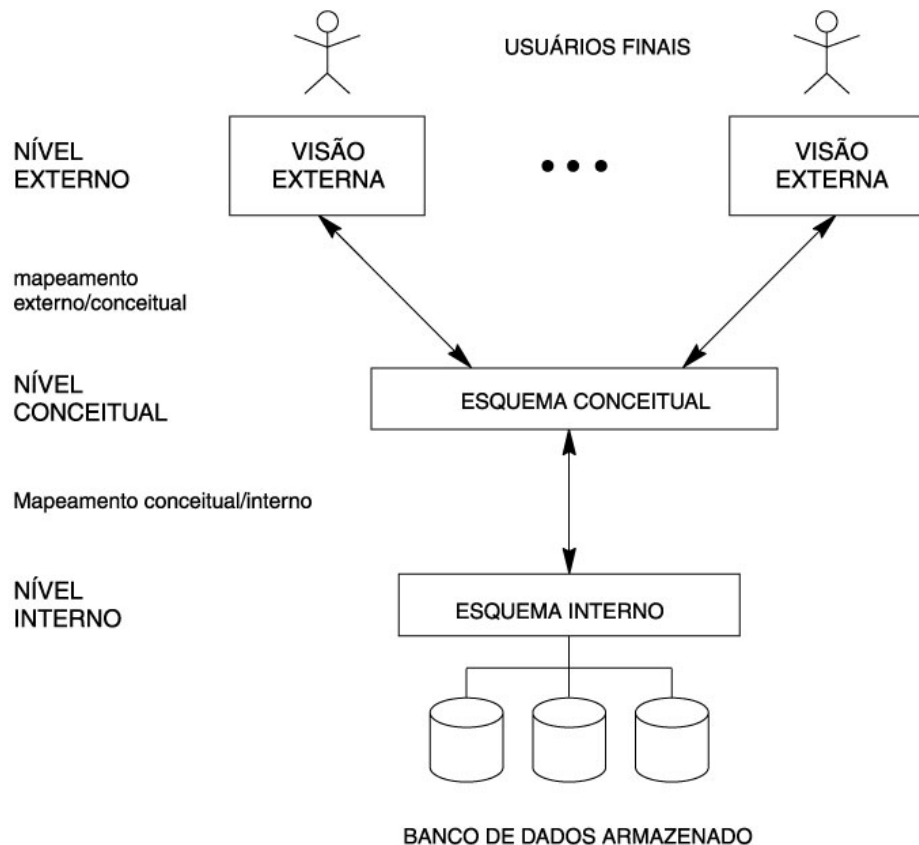


Figura 5 - A arquitetura de três-esquemas

Independência de Dados

A “independência de dados” pode ser definida como a capacidade de se alterar um esquema em um nível em um banco de dados sem ter que alterar um nível superior (vide figura anterior). Existem dois tipos de independência de dados:

- Independência de dados lógica: é a capacidade de alterar o esquema conceitual sem ter que alterar o esquema externo ou as aplicações do usuário;
- Independência de dados física: é a capacidade de alterar o esquema interno sem ter que alterar o esquema conceitual, o esquema externo ou as aplicações do usuário.

As Linguagens para Manipulação de Dados

Para a definição dos esquemas **conceitual** e **interno**, pode-se utilizar uma linguagem chamada DDL (Data Definition Language - Linguagem de Definição de Dados). O SGBD possui um compilador DDL que permite a execução das declarações para identificar as descrições dos esquemas e para armazená-las no catálogo do SGBD. A DDL é utilizada em SGBDs onde a separação entre os níveis interno e conceitual não é muito clara.

Exemplo de linguagem DDL: drop table produtos;

Em um SGBD em que a separação entre os níveis conceitual e interno são bem claras, é utilizado uma outra linguagem, a SDL (Storage Definition Language - Linguagem de Definição de Armazenamento) para a especificação do esquema interno. A especificação do esquema conceitual fica por conta da DDL.

Em um SGBD que utiliza a arquitetura três esquemas, é necessária a utilização de mais uma linguagem para a definição de visões, a VDL (Vision Definition Language - Linguagem de Definição de Visões).

Uma vez que o esquema esteja compilado e o banco de dados esteja populado, usa-se uma linguagem para fazer a manipulação dos dados, a DML (Data Manipulation Language - Linguagem de Manipulação de Dados).

Exemplo de linguagem DML: select * from produtos

Os Módulos Componentes de um SGBD

A figura abaixo ilustra, de forma simplificada, os componentes típicos de um SGBD. O banco de dados e o catálogo de SGBD são normalmente armazenados no disco. O acesso ao disco é controlado principalmente pelo sistema operacional (SO), que organiza as entradas e as saídas. Um módulo de gerenciamento de dados armazenados de algo nível do SGBD controla o acesso à informação do SGBD que está armazenado no disco, se for parte do banco de dados ou do catálogo.

Na figura abaixo, os pontos identificados pelas letras A,B,C,D e E ilustram os acessos controlados pelo gerenciador de dados armazenados.

O **compilador DDL** processa as definições do esquema, especificadas na DDL, e armazena as descrições dos esquemas (metadados) no catálogo do SGBD.

O **catálogo** inclui informações como nomes e tamanhos dos arquivos, nomes e tipos de itens de dados, detalhes de armazenamento de cada arquivo, informações de mapeamentos entre os esquemas e restrições,

além de muitas outras informações necessárias para os módulos do SGBD. Os módulos de software do SGBD acessam as informações do catálogo conforme necessário.

O **processador de banco de dados em tempo de execução** (runtime) controla o acesso ao banco de dados em tempo de execução, recebe os comandos para a recuperação ou atualização e os executa no banco de dados.

O **compilador de consulta** (query) manipula as consultas de alto nível que são feitas interativamente. Ele analisa a sintaxe, compila ou interpreta a consulta criando um código de acesso ao banco de dados e então gera as chamadas ao processador em tempo de execução para executar o código.

O **pré-compilador** extrai os comandos DML dos programas escritos em linguagem de programação hospedeira. Esses comandos são enviados para o compilador DML para compilação, gerando códigos para o acesso ao BD.

Atualmente é comum ter um programa cliente que acessa o SGBD de outro computador separado daquele em que está o banco de dados. O primeiro é chamado **computador cliente**, e o último, **servidor de banco de dados**. Em alguns casos, o cliente acessa um computador intermediário, o servidor de aplicação, que, por sua vez, acessa o servidor de banco de dados.

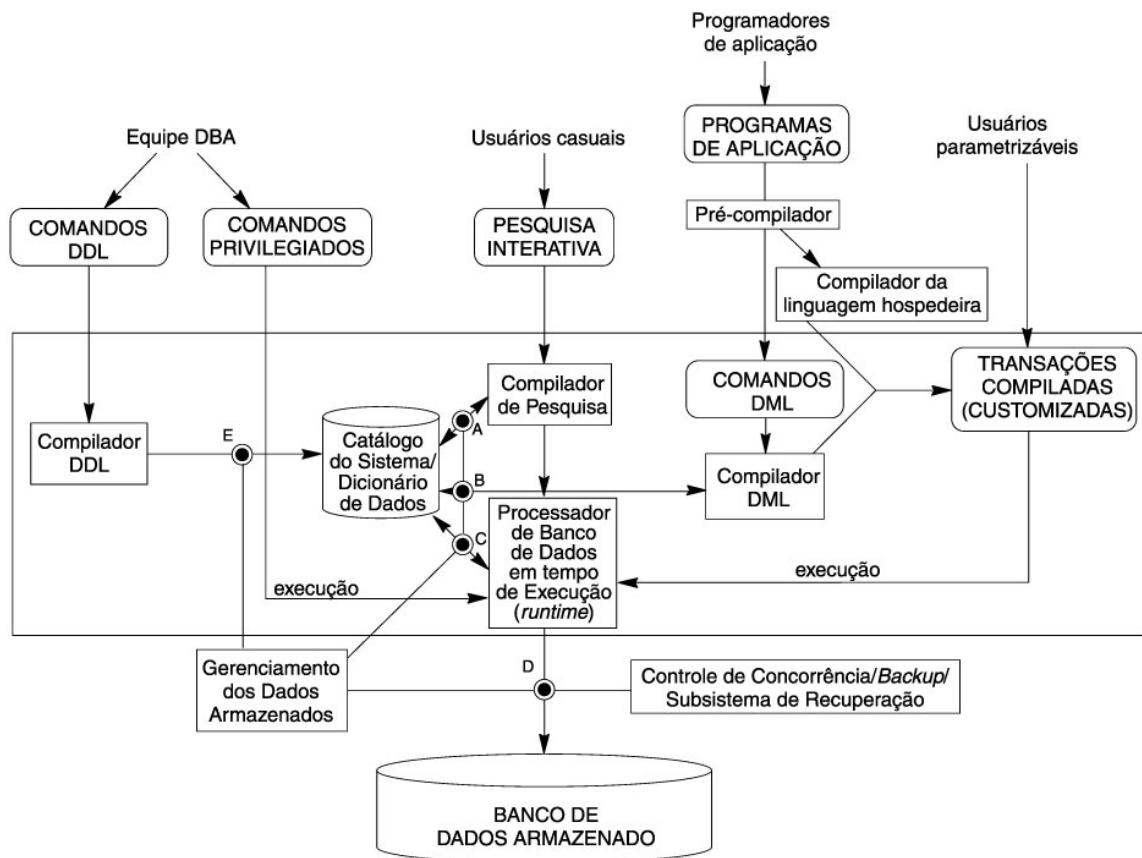


Figura 6 - Módulos componentes de um SGBD e suas interações

Arquitetura SGBD Centralizada

As primeiras arquiteturas utilizavam os grandes computadores centrais (mainframes) para processar todas as funções do sistema, incluindo os programas de aplicação e os de interface com os usuários, bem como todas as funcionalidades do SGBD. A razão disso é que os usuários acessavam o servidor através de terminais que não tinham poder de processamento e apenas ofereciam a possibilidade de exibição. Sendo assim, todos os processos eram executados pelo servidor.

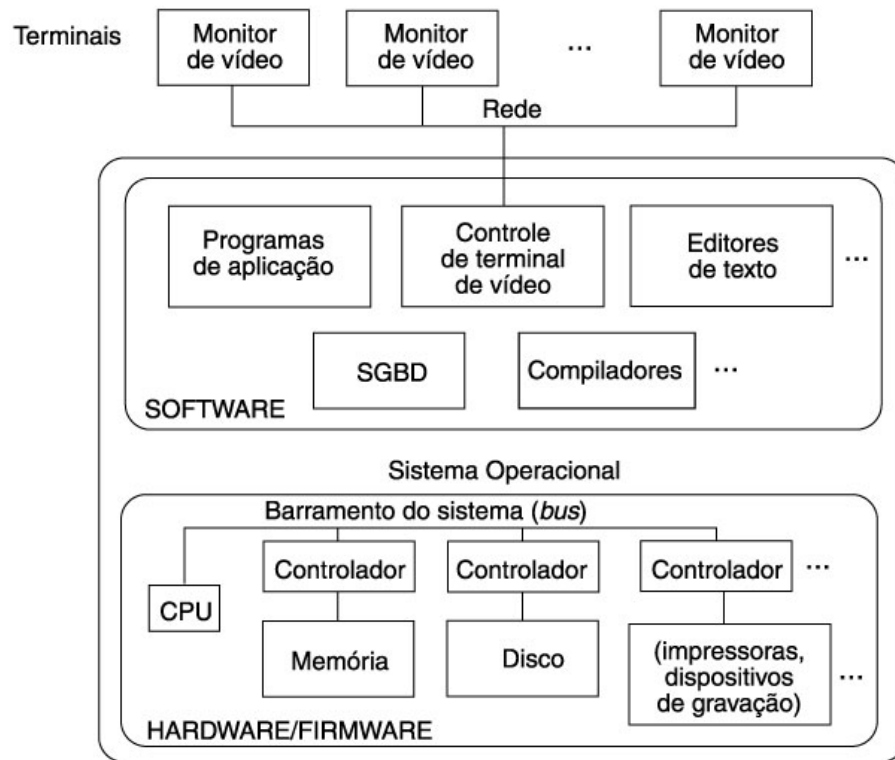


Figura 7 - A arquitetura física centralizada

Arquiteturas Cliente/Servidor

Esta arquitetura foi projetada pra trabalhar com ambientes computacionais, nos quais um grande número de PCs, estações de trabalho, servidores de arquivo, impressoras, servidores de banco de dados, servidores Web e outros equipamentos estão conectados em rede. As **máquinas locais** oferecem ao usuário as interfaces apropriadas para utilizar esses servidores, bem como o poder de processamento para executar as aplicações locais. O conceito de arquitetura cliente/servidor possui uma estrutura fundamental que consiste em muitos PCs e estações de trabalho, bem como um número pequeno de máquinas centrais (mainframes) conectadas via redes locais e outros tipos de redes de computadores.

Um **cliente** nessa estrutura é, em geral, uma máquina de usuário que tem as funcionalidades de interface com o usuário e processamento o local. Quando um cliente precisa de uma funcionalidade adicional, como o acesso ao banco de dados, inexistente naquela máquina, ele se conecta a um servidor que disponibiliza essa funcionalidade.

Um **servidor** é uma máquina que pode fornecer serviços para as máquinas clientes, como acesso a arquivos, impressão, arquivamento ou acesso a um banco de dados.



Figura 8 - A arquitetura lógica de duas camadas cliente/servidor.

Em geral, algumas máquinas instalam apenas o software cliente, outras apenas o software servidor. Algumas podem incluir ambos. Porém, normalmente os softwares de cliente e servidor são executados em máquinas separadas. Dois tipos principais de arquiteturas de SGBD foram criados utilizando-se os fundamentos da estrutura cliente/servidor: duas e três camadas. Na figura abaixo é apresentada uma arquitetura em 2 camadas. Esta arquitetura é chamada de arquitetura de duas camadas, pois os componentes de software são distribuídos em dois sistemas: cliente e servidor. Nesta arquitetura, geralmente o servidor é chamado de **servidor de transação**, **servidor de consulta** ou **servidor SQL**.

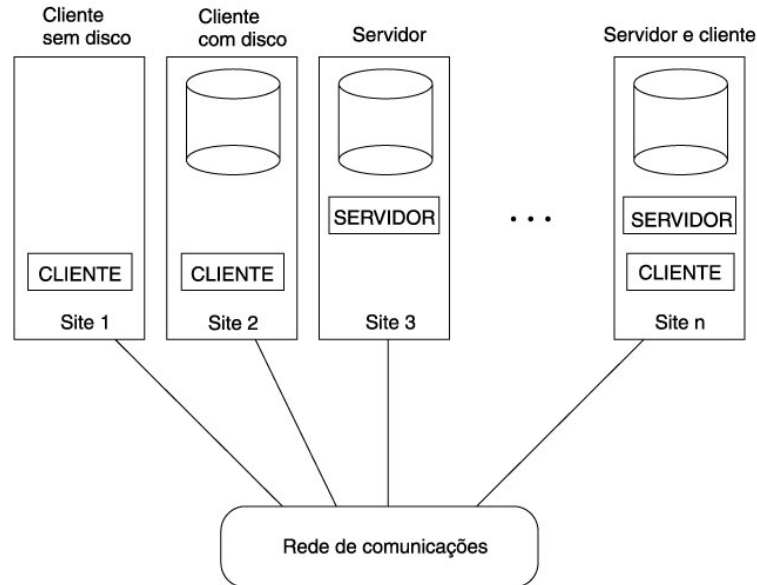


Figura 9 - Arquitetura física cliente/servidor de duas camadas.

Muitas aplicações Web usam uma arquitetura chamada **arquitetura três camadas**, que possui uma camada intermediária entre o cliente e o servidor de banco de dados, como ilustrado na figura abaixo. Essa camada intermediária, ou camada do meio, é, algumas vezes, chamada de **servidor de aplicações** ou **servidor Web**, dependendo da aplicação. Esse servidor desempenha um papel intermediário armazenando as regras de negócio que são usadas para acessar os dados do servidor de banco de dados. Também pode incrementar a segurança do banco de dados checando as credenciais do cliente antes de enviar uma solicitação ao servidor de banco de dados. Os clientes possuem interfaces GUI (*graphic user interface* – interface gráfica com o usuário) e algumas regras de negócio adicionais específicas para a aplicação. O servidor intermediário aceita as solicitações do cliente, processa e envia comandos de banco de dados ao servidor de banco de dados, retornando então os dados passados pelo BD ao cliente.

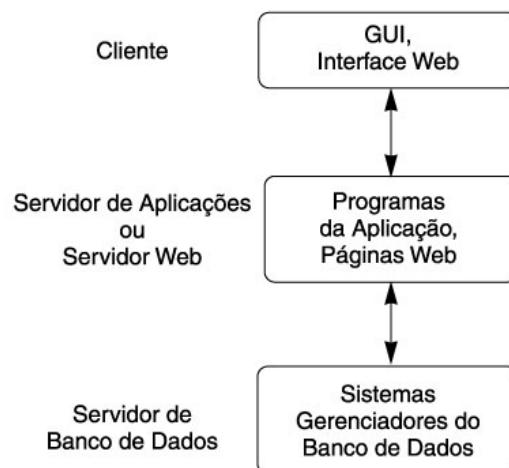


Figura 10 - Arquitetura lógica cliente/servidor de três camadas.

Classificação dos SGBDs

O principal critério para se classificar um SGBD é o modelo de dados no qual é baseado. A grande maioria dos SGBDs contemporâneos é baseada no modelo relacional, alguns em modelos conceituais e alguns em modelos orientados a objetos. Outras classificações são:

- **Usuários:** um SGBD pode ser mono-usuário, comumente utilizado em computadores pessoais ou multiusuários, utilizado em estações de trabalho, mini-computadores e máquinas de grande porte;
- **Localização:** um SGBD pode ser localizado ou distribuído; se ele for localizado, então todos os dados estarão em uma máquina (ou em um único disco) ou distribuído, onde os dados estarão distribuídos por diversas máquinas (ou diversos discos);
- **Ambiente:** ambiente homogêneo é o ambiente composto por um único SGBD e um ambiente heterogêneo é o ambiente compostos por diferentes SGBDs.

Modelagem de Dados Utilizando o Modelo Entidade Relacionamento (ER)

O modelo Entidade-Relacionamento é um modelo de dados conceitual de alto nível, cujos conceitos foram projetados para estar o mais próximo possível da visão que o usuário tem dos dados, não se preocupando em representar como estes dados estarão realmente armazenados. O modelo ER é utilizado principalmente durante o processo de projeto de banco de dados.

Entidades e Atributos

O objeto básico tratado pelo modelo ER é a “**entidade**”, que pode ser definida como um objeto do mundo real, concreto ou abstrato e que possui existência independente. Cada entidade possui um conjunto particular de propriedades que a descreve chamado “**atributos**”. Um atributo pode ser dividido em diversas sub-partes com significado independente entre si, recebendo o nome de “**atributo composto**”. Um atributo que não pode ser subdividido é chamado de “**atributo simples**” ou “**atômico**”.

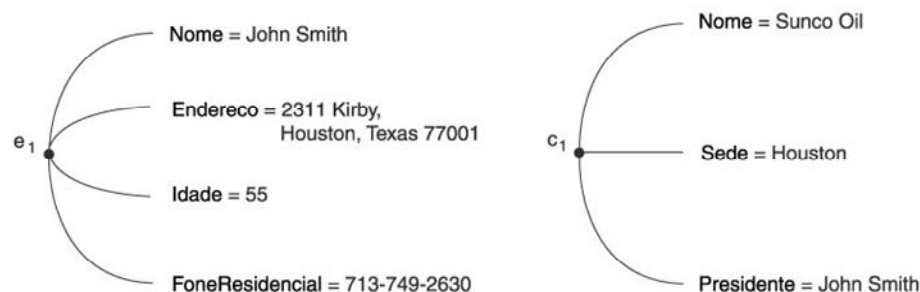


Figura 11 - Duas entidades, empregado **e1** e empresa **c1**, e seus atributos.

Os atributos que podem assumir apenas um determinado valor em uma determinada instância é denominado “**atributo simplesmente valorado ou monovalorado**”, enquanto que um atributo que pode assumir diversos valores em uma mesma instância é denominado “**atributo multivalorado**”.

Como exemplo, a idade seria um atributo simplesmente valorado de uma pessoa, enquanto cor ou titulação seriam atributos multivalorados pois possuem um conjunto de valores para uma pessoa. Uma pessoa pode não

ter um título acadêmico, outra pessoa pode ter um e, uma terceira pessoa pode ter dois ou mais títulos. Portanto, pessoas diferentes podem ter números de valores diferentes para o atributo titulação. Esses atributos são chamados multivalorados.

Um atributo que é gerado a partir de outro atributo é chamado de “**atributo derivado**”. Como exemplo, imagine o atributo DataDeNascimento: o atributo idade pode ser calculado a partir da data de nascimento.

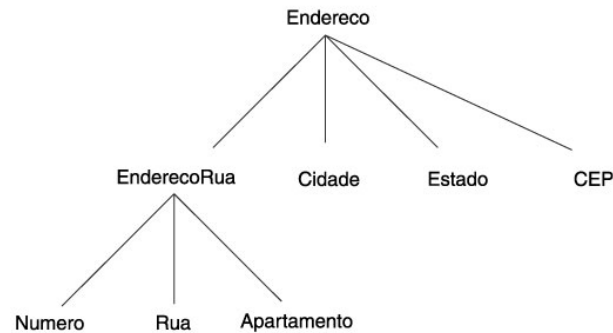


Figura 12 - Uma hierarquia de atributos compostos.

Valores Nulos (NULL)

Em alguns casos, determinada entidade pode não ter um valor aplicável a um atributo. Por exemplo, o atributo Apartamento de um endereço se aplica apenas a endereços que estão em edifícios de apartamentos, e não a outros tipos de residência, como as casas. O sentido do valor NULL pode ser aplicado a “não aplicável” ou mesmo a “falta da informação”.

Como exemplo, imagine um atributo que armazene um valor inteiro. Utiliza-se o valor NULL para informar que o atributo não possui nenhum valor, já que zero (0) é um valor.

Tipos Entidade, Conjunto de Valores, Atributo Chave e Domínio

Um banco de dados costuma conter grupos de entidades que são similares, possuindo os mesmos atributos, porém, cada entidade com seus próprios valores para cada atributo. Este conjunto de entidades similares define um “tipo entidade”. Cada tipo entidade é identificada por seu nome e pelo conjunto de atributos que definem suas propriedades. A descrição do tipo entidade é chamada de “esquema do tipo entidade”, especificando o nome do tipo entidade, o nome de cada um de seus atributos e qualquer restrição que incida sobre as entidades. Como exemplo, veja a figura abaixo.

NOME DO TIPO ENTIDADE:

EMPREGADO

Nome, Idade, Salario

EMPRESA

Nome, Sede Administrativa, Presidente

CONJUNTO DE ENTIDADE: (EXTENSÃO)

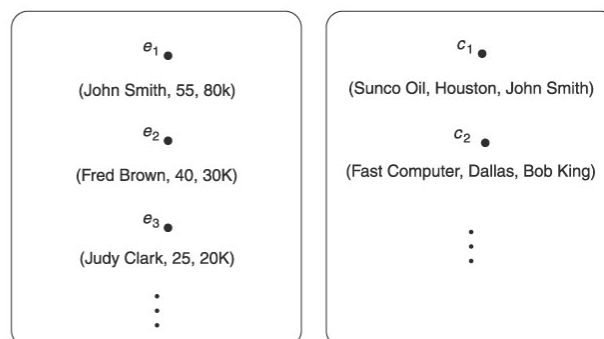


Figura 13 - Dois tipos entidade, EMPREGADO e EMPRESA, e algumas entidades-membro de cada um.

Uma restrição muito importante em uma entidade de um determinado tipo entidade é a “**chave**”. Um tipo entidade possui um atributo cujos valores são distintos para cada entidade individual. Este atributo é chamado “**atributo chave**” e seus valores podem ser utilizados para identificar cada entidade de forma única. Muitas vezes, uma chave pode ser formada pela composição de dois ou mais atributos. Uma entidade pode também ter mais de um atributo chave.

Cada atributo simples de um tipo entidade está associado com um conjunto de valores denominado “**domínio**”, o qual especifica o conjunto de valores que podem ser designados para este determinado atributo para cada entidade. Por exemplo, o **domínio** INTEGER é o conjunto de todos os inteiros possíveis.

Chave Primária e Chave Candidata

Em geral, uma entidade pode ter mais de uma chave. Nesse caso, cada uma das chaves é chamada de **chave candidata**. Por exemplo, na tabela abaixo, os atributos “codigo” e “CPF” podem ser utilizados como chave, portanto são **chaves candidatas**. Chave primária é a chave candidata indicada para identificar uma tupla (registro). Normalmente ela é indicada com um (*) ou sublinhada. Ao escolher a chave primária, ela não deve se repetir, não deve possuir valores NULL e, de preferência, deve ter um conteúdo pequeno.

Codigo *	CPF	Nome
1	123456789-09	Ana
2	987654321-01	João
3	569874512-31	Patrícia

Tipos e Instâncias de Relacionamento

Além de conhecer detalhadamente os tipos entidade, é muito importante conhecer também os relacionamentos entre estes tipos entidades.

Um “tipo relacionamento” **R** entre n entidades **E1, E2, ..., En**, é um conjunto de associações entre entidades deste tipo. Informalmente falando, cada instância de relacionamento **r1** em **R** é uma associação de entidades, onde a associação inclui exatamente uma entidade de cada tipo entidade participante no tipo relacionamento. Isto significa que estas entidades estão relacionadas de alguma forma no mini-mundo.

A figura abaixo mostra um exemplo entre dois tipos entidade (**empregado** e **departamento**) e o relacionamento entre eles (**trabalha para**).

Repare que para cada relacionamento, participam apenas uma entidade de cada tipo entidade, porém, uma entidade pode participar de mais do que um relacionamento.

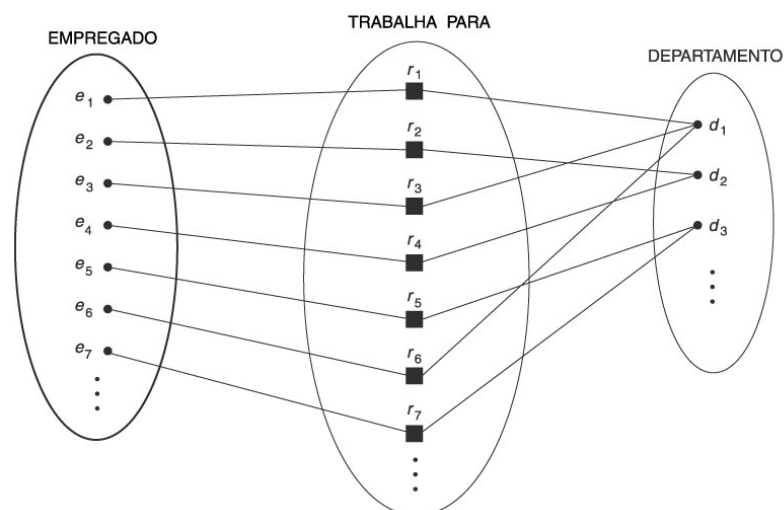


Figura 14 - Algumas instâncias do conjunto de relacionamento **TRABALHA_PARA**, que representa um tipo relacionamento **TRABALHA_PARA** entre **EMPREGADO** e **DEPARTAMENTO**.

Grau de um Relacionamento

O “grau” de um **tipo relacionamento** é o número de tipos entidade que participam do tipo relacionamento. No exemplo da figura anterior, temos um relacionamento binário. O grau de um relacionamento é ilimitado, porém, a partir do grau 3 (ternário), a compreensão e a dificuldade de se desenvolver a relação corretamente se tornam extremamente complexas. Na figura abaixo há um exemplo de um tipo relacionamento (fornece).

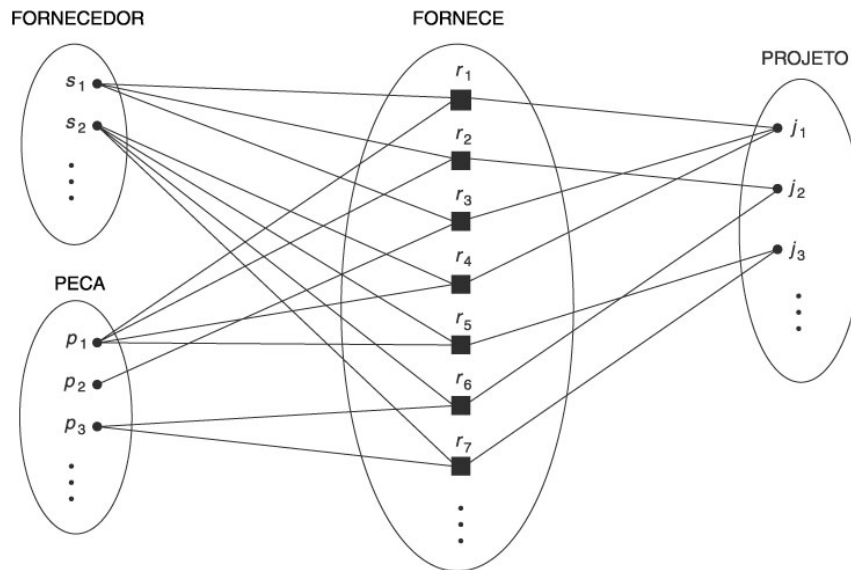


Figura 15 - Algumas instâncias de relacionamento do conjunto de relacionamento ternário **FORNECE**.

Outras Características de um Relacionamento

Relacionamentos como Atributos

Algumas vezes é conveniente pensar em um relacionamento como um atributo. Considere o exemplo da Figura 14. Podemos pensar departamento como sendo um atributo da entidade empregado, ou empregado como um atributo multivalorado da entidade departamento. Se uma entidade não possuir existência muito bem definida, talvez seja mais interessante que ela seja representada como um atributo.

Nomes de Papéis e Relacionamentos Recursivos

Cada tipo entidade que participa de um tipo relacionamento desempenha um **papel** particular no relacionamento. O nome do papel representa o papel que uma entidade de um tipo entidade participante desempenha no relacionamento. No exemplo da Figura 14, nós temos o papel empregado ou trabalhador para o tipo entidade EMPREGADO e o papel departamento ou empregador para a entidade DEPARTAMENTO. Nomes de papéis não são necessariamente importantes quando todas as entidades participantes desempenham papéis diferentes. Algumas vezes, o papel torna-se essencial para distinguir o significado de cada participação. Isto é muito comum em “**relacionamentos recursivos**”.

Um **relacionamento recursivo** é um relacionamento entre entidades do mesmo tipo entidade. Veja o exemplo da figura abaixo.

No exemplo, temos um relacionamento entre o tipo entidade EMPREGADO, onde um empregado pode supervisionar outro empregado e um empregado pode ser supervisionado por outro empregado. Sendo assim, e_1 supervisiona e_2 e e_3 , e_4 supervisiona e_6 e e_7 , e e_5 supervisiona e_1 e e_4 .

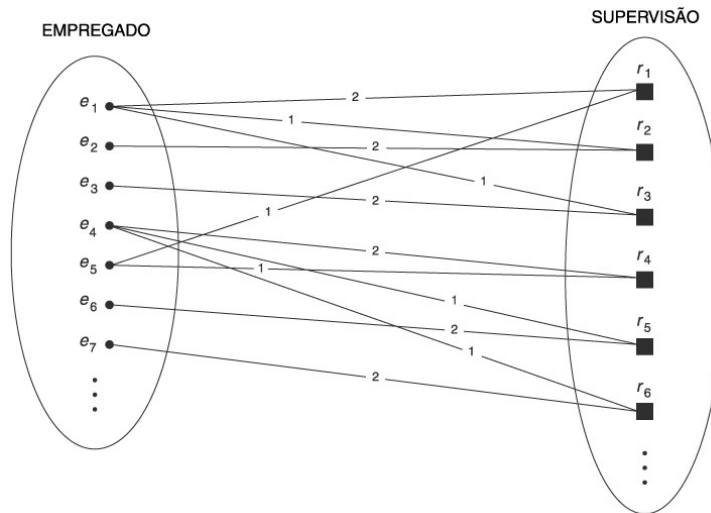


Figura 16 - Um relacionamento recursivo SUPERVISÃO entre EMPREGADO, no papel de *supervisor* (1), e EMPREGADO, no papel de *subordinado* (2).

Restrições em Tipos Relacionamentos

Geralmente, os tipos relacionamentos sofrem certas restrições que limitam as possíveis combinações das entidades participantes. Estas restrições são derivadas de restrições impostas pelo estado destas entidades no mini-mundo. Veja o exemplo da figura abaixo:

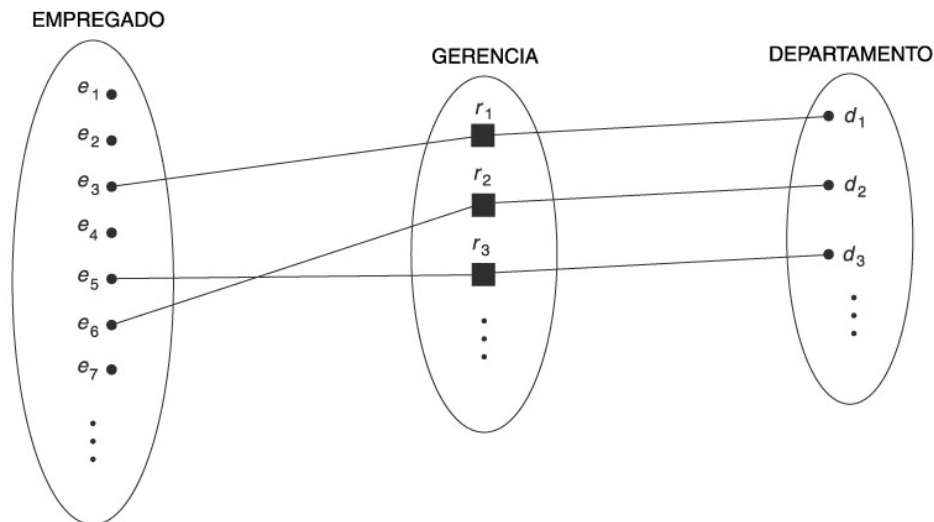


Figura 17 - Um relacionamento GERENCIA 1:1.

Nesta figura temos a seguinte situação: um empregado pode gerenciar apenas um departamento, enquanto que um departamento pode ser gerenciado por apenas um empregado. A este tipo de restrição, nós chamamos **cardinalidade**. A **cardinalidade** indica o número de relacionamentos dos quais uma entidade pode participar.

A cardinalidade pode ser: 1:1, 1:N, M:N. No exemplo da figura acima, a cardinalidade é 1:1, pois cada entidade empregado pode gerenciar apenas um departamento e um departamento pode ser gerenciado por apenas um empregado.

No exemplo da Figura 14, no relacionamento EMPREGADO Trabalha Para DEPARTAMENTO, o relacionamento é 1:N, pois um empregado pode trabalhar em apenas um departamento, enquanto que um departamento pode possuir vários empregados.

Na figura abaixo temos um exemplo de um relacionamento com cardinalidade N:M.

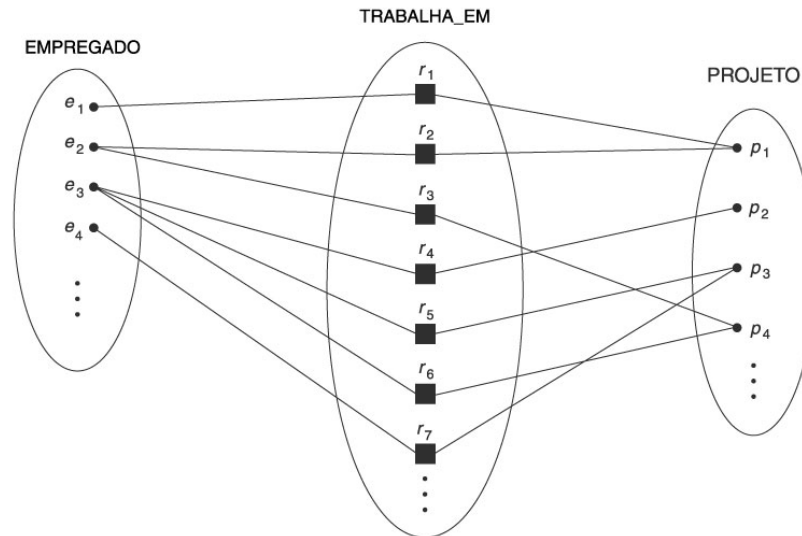


Figura 18 - Exemplo de relacionamento M:N.

No exemplo da figura anterior, nós temos que um empregado pode trabalhar em vários projetos enquanto que um projeto pode ter vários empregados trabalhando.

Outra restrição muito importante é a **participação**. A participação define a existência de uma entidade através do relacionamento, podendo ser **parcial** ou **total**. Veja o exemplo da Figura 17. A participação do empregado é **parcial**, pois nem todo empregado gerencia um departamento, porém a participação do departamento neste relacionamento é **total**, pois todo departamento precisa ser gerenciado por um empregado. Desta forma, **todas** as entidades do tipo entidade DEPARTAMENTO precisam participar do relacionamento, mas **nem todas** as entidade do tipo entidade EMPREGADO precisam participar do relacionamento.

Já no exemplo da Figura 14, ambas as participações são totais, pois, todo empregado precisa trabalhar em um departamento e todo departamento tem que ter empregados trabalhando nele. Estas restrições são chamadas de **restrições estruturais**.

- Algumas vezes, torna-se necessário armazenar um atributo no tipo relacionamento. Veja o exemplo da Figura 17. Pode-se querer saber em que dia o empregado passou a gerenciar o departamento. É difícil estabelecer a qual tipo entidade pertence atributo, pois o mesmo é definido apenas pela existência do relacionamento.
- Quando temos relacionamentos com cardinalidade 1:1, podemos colocar o atributo em uma das entidades, de preferência, em uma cujo tipo entidade tenha participação total. No caso, o atributo poderia ir para o tipo entidade departamento. Isto porque nem todo empregado participará do relacionamento.
- Caso a cardinalidade seja 1:N, então podemos colocar o atributo no tipo entidade com participação N.
- Se a cardinalidade for N:M, então o atributo deverá mesmo ficar no tipo relação. Veja o exemplo da Figura 18. Caso queiramos armazenar quantas horas cada empregado trabalhou em cada projeto, então este deverá ser um atributo do relacionamento.

Tipos Entidades Fracas

Alguns tipos entidade podem não ter seu próprio atributo chave. Isto implica que não poderemos distinguir algumas entidades por que as combinações dos valores de seus atributos podem ser idênticas. Estes tipos entidade são chamados **entidades fracas**. As entidades deste tipo precisam estar relacionadas com uma entidade pertencente ao tipo entidade **proprietária**. Este relacionamento é chamado de **relacionamento**

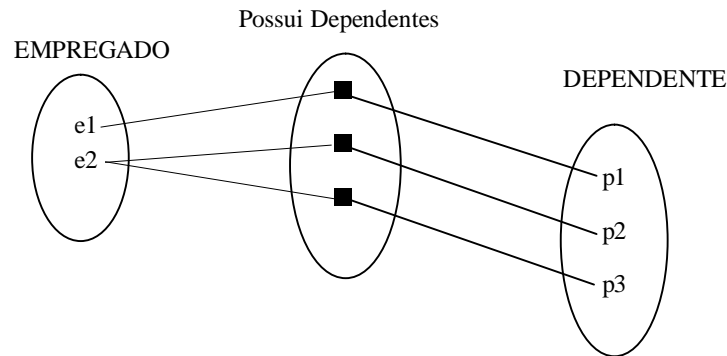


Figura 19 - Relacionamento com uma Entidade Fraca (Dependente)

Na figura acima, o tipo entidade DEPENDENTE é uma entidade fraca, pois não possui um método de identificar uma entidade única. O EMPREGADO não é uma entidade fraca, pois possui um atributo para identificação (atributo chave). O número do RG de um empregado identifica um único empregado. Porém, um dependente de 5 anos de idade não possui necessariamente um documento. Desta forma, esta entidade é um tipo entidade fraca. Um tipo entidade fraca possui uma **chave parcial**, que juntamente com a chave primária da entidade proprietária forma uma chave primária composta. Neste exemplo: a chave primária do EMPREGADO é o RG. A chave parcial do DEPENDENTE é o seu nome, pois dois irmãos não podem ter o mesmo nome. Desta forma, a chave primária desta entidade fica sendo o RG do pai ou mãe mais o nome do dependente.

Diagrama Entidade Relacionamento

O diagrama Entidade Relacionamento é composto por um conjunto de objetos gráficos que visa representar todos os objetos do modelo Entidade Relacionamento tais como entidades, atributos, atributos chaves, relacionamentos, restrições estruturais, etc.

O diagrama ER fornece uma visão lógica do banco de dados, fornecendo um conceito mais generalizado de como estão estruturados os dados de um sistema. Os objetos que compõem o diagrama ER estão listados a seguir, na figura abaixo:

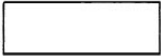






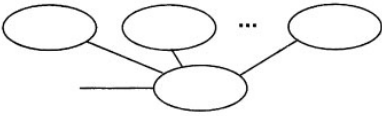
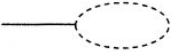
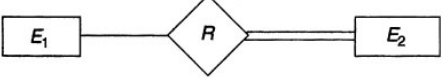
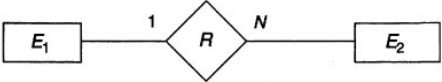
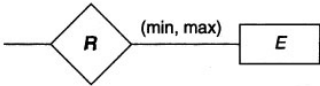
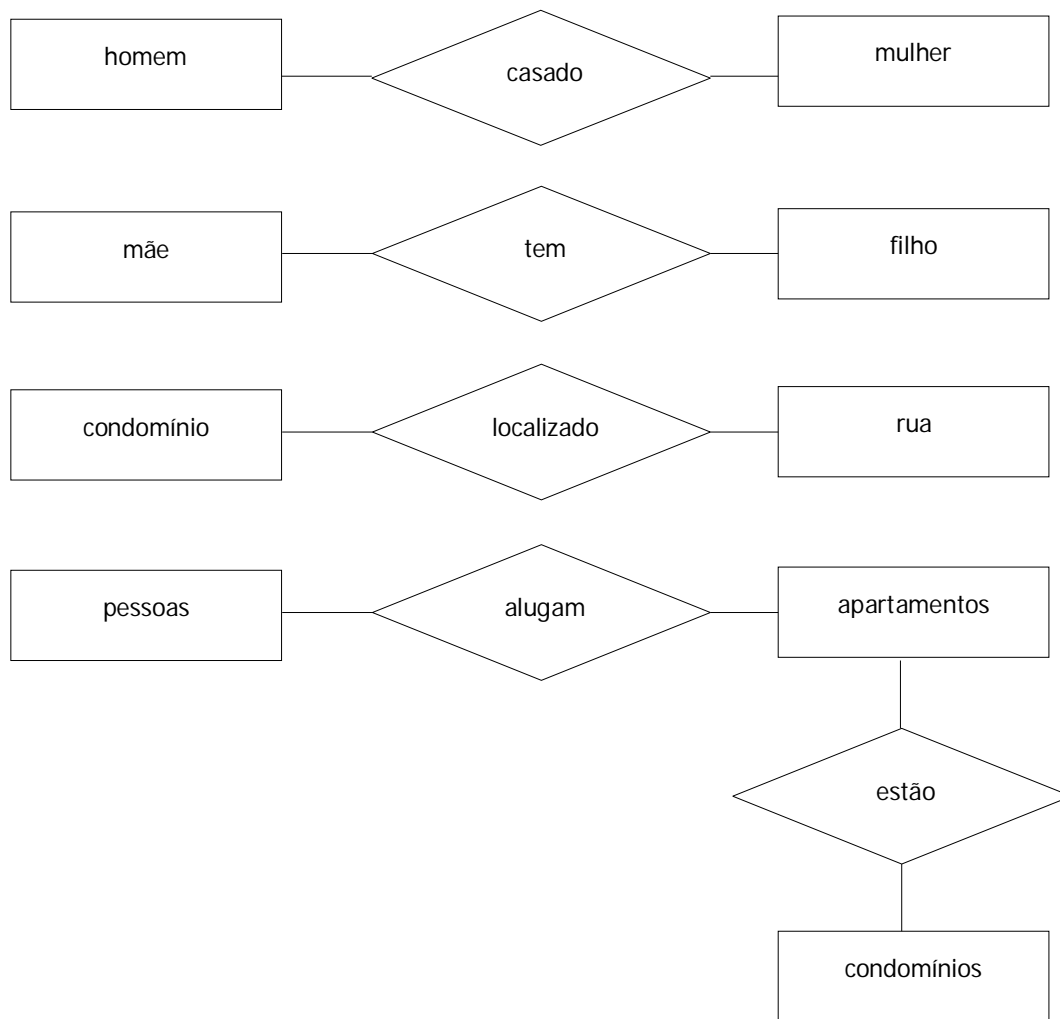
Símbolo	Significado
	ENTIDADE
	FRACA
	RELACIONAMENTO
	IDENTIFICADOR DE RELACIONAMENTO
	ATRIBUTO-CHAVE
	
	ATRIBUTO MULTIVALORADO
	ATRIBUTO COMPOSTO
	ATRIBUTO DERIVADO
	PARTICIPAÇÃO TOTAL DE E_2 EM R
	RAZÃO DE CARDINALIDADE 1:N PARA $E_1:E_2$ EM R
	RESTRIÇÃO ESTRUTURAL (MIN,MAX) DA PARTICIPAÇÃO DE E EM R

Figura 20 - Resumo da notação para diagramas ER.

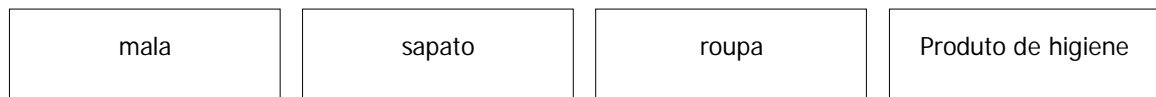
Exemplos de relacionamentos entre entidades:**Figura 21 – Representação de relacionamentos**

Exemplo prático: a Viagem

Imagine que você precisa se preparar para uma viagem ao caribe. Você tem em casa um jogo de malas com tamanhos, cores e de materiais diferentes, com rodinhas, sem rodinhas, ou seja, uma infinidade de características para cada uma. Em suas malas, você vai colocar roupas, sapatos, produtos de higiene, etc, tudo aquilo que se leva quando se sai de férias.

Como as roupas são diferentes umas das outras, criamos uma série de atributos que as caracterizam e individualizam, e da mesma forma iremos proceder com sapatos e produtos de higiene.

Logo teremos 4 entidades envolvidas com a nossa viagem, ou seja:



Vamos entender quais são os seus atributos básicos:

Entidade	Atributos
Mala	Cor , Volume , Material, Indicador de rodas, Tipo de fechamento
Roupa	Cor, material (tecido), descrição da roupa
Sapato	Cor, tipo (esporte/social), marca
Produto de higiene	Nome, Marca

Para que você saiba o que há em cada mala (para o caso de extravio da mesma), iremos relacioná-la com os produtos colocados nela.

Agora, iremos criar relacionamentos entre as entidades envolvidas na viagem, que tenham o mesmo efeito de uma lista de coisas colocadas em cada mala, isto é na realidade uma visão dos dados com relacionamentos entre eles. O que iremos fazer é o modelo E-R (entidade-relacionamento), que irá representar com a mesma simplicidade da vida real este relacionamento:

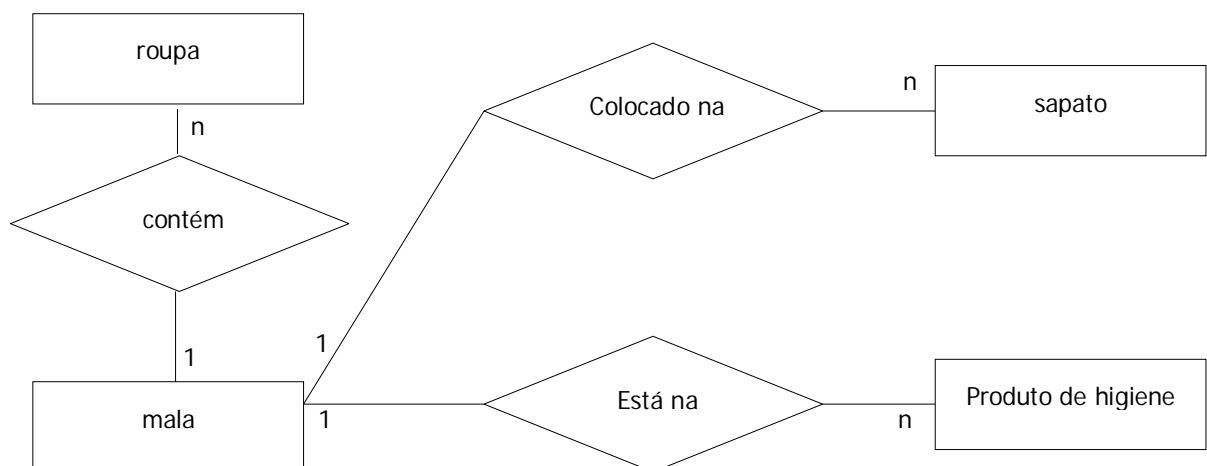


Figura 22 – Relacionamentos para “a viagem” já com a cardinalidade informada.

Normalização

O conceito de normalização foi introduzido por E.F. Codd em 1970 (primeira forma normal). Esta técnica é um processo matemático formal, que tem seus fundamentos na teoria dos conjuntos.

Através deste processo pode-se, gradativamente, substituir um conjunto de entidades e relacionamentos por um outro, o qual se apresenta “purificado” em relação às anomalias de atualização (inclusão, alteração e exclusão) as quais podem causar certos problemas, tais como:

- Grupos repetitivos (atributos multivalorados) de dados
- Dependências parciais em relação a uma chave concatenada
- Redundâncias de dados desnecessárias
- Perdas acidentais de informação
- Dificuldade na representação dos fatos da realidade observada
- Dependências transitivas entre atributos.

Código do vendedor: 1791
Prazo de entrega: 20 dias
Nome do vendedor: Aníbal da Silva

Num. Ped.	Cliente	Endereço	CGC	IE	Cód. Produto	Unid.	Quant.	Descrição Produto	Valor Unit.	Tot. Prod.	Total do Pedido
2610	Lopes..	R. 127...	23232	3434	45	L	50	Álcool	5,00	250,00	2650,00
2610	Lopes..	R. 127...	23232	3434	78	Kg	47	Cimento	30,00	1410,00	2650,00
2610	Lopes..	R. 127...	23232	3434	21	Kg	20	Pregos	5,00	100,00	2650,00
2610	Lopes..	R. 127...	23232	3434	98	L	15	Tinta azul	25,00	375,00	2650,00
2610	Lopes..	R. 127...	23232	3434	90	L	15	Cola	3,00	45,00	2650,00
2610	Lopes..	R. 127...	23232	3434	43	M	10	Arame	3,00	30,00	2650,00
2610	Lopes..	R. 127...	23232	3434	25	F	10	Algodão	2,00	20,00	2650,00
2610	Lopes..	R. 127...	23232	3434	65	L	5	Querosene	8,00	40,00	2650,00
2610	Lopes..	R. 127...	23232	3434	51	M	20	Fio elétrico	13,00	260,00	2650,00
2610	Lopes..	R. 127...	23232	3434	74	M	30	Linha 10	4,00	120,00	2650,00

Figura 23- Um formulário de PEDIDO

Pense: como você faria para identificar de forma única (definir uma chave primária) para a entidade acima?

Observando-se o formulário de PEDIDO apresentado acima, podemos considerar que uma entidade formada com os dados presentes neste documento, terá a seguinte apresentação:

Atributos da entidade PEDIDO:

- Número do pedido
- Prazo de entrega
- Código do vendedor
- Nome do vendedor
- Cliente
- Endereço
- CGC
- Inscrição estadual
- Código do produto (*)
- Unidade do produto (*)
- Quantidade do produto (*)
- Descrição do produto (*)
- Valor unitário do produto (*)
- Valor total do produto (*)
- Valor total do pedido (*)

(*) Atributos que se repetem no documento

Caso esta entidade fosse implementada como uma tabela em um banco de dados, as seguintes anomalias iriam aparecer:

- Anomalia da inclusão: ao ser incluído um novo cliente, o mesmo tem que estar relacionado a uma venda;
- Anomalia da exclusão: ao ser excluído um cliente, os dados referentes as suas compras serão perdidos;

Primeira forma normal (1FN)

Em uma determinada realidade, às vezes encontramos algumas informações que se repetem (atributos multivalorados), retratando ocorrências de um mesmo fato dentro de uma única linha e vinculadas a sua chave primária.

Ao observarmos a entidade PEDIDO, apresentada acima, visualizamos que um certo grupo de atributos (produtos solicitados) se repete (número de ocorrências não definidas) ao longo do processo de entrada de dados na entidade.

A 1FN diz que: cada ocorrência da chave primária deve corresponder a uma e somente uma informação de cada atributo, ou seja, a entidade não deve conter grupos repetitivos (multivalorados).

Para se obter entidades na 1FN, é necessário decompor cada entidade não normalizada em tantas entidades quanto for o número de conjuntos de atributos repetitivos. Nas novas entidades criadas, a chave primária é a concatenação da chave primária da entidade original mais o(s) atributo(s) do grupo repetitivo visualizado(s) com chave primária deste grupo.

Ao aplicarmos a 1FN sobre a entidade PEDIDO, obteremos mais uma entidade chamada de ITEM-DE-PEDIDO, que herdará os atributos repetitivos e destacados na entidade PEDIDO.

Ao aplicarmos a 1FN na entidade PEDIDO, a pergunta que deve fazer é:

Para um determinado número de pedido (chave primária), pode se ter :

- Quantos clientes? (resp: apenas 1)
- Quantos endereços? (resp: apenas 1)
- Quantos códigos de vendedor? (resp: apenas 1)
- Quantos códigos de produto? (resp: muitos)
- Quantas unidades de produto? (resp: muitas)

Etc...

PEDIDO

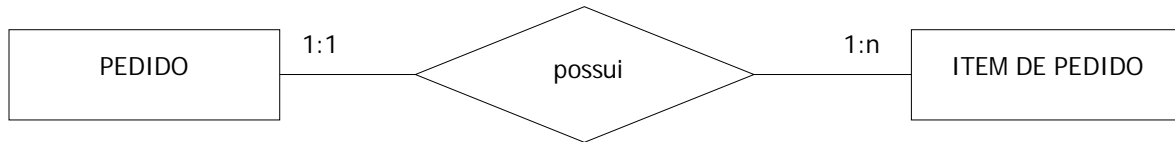
* Número do pedido
Cliente
Endereço
CGC
Inscrição estadual
Valor total do pedido
Prazo de entrega
Código do vendedor
Nome do vendedor

ITEM-DO-PEDIDO

* Número do pedido
* Código do produto
Unidade do produto
Quantidade do produto
Descrição do produto
Valor unitário do produto
Valor total do produto

Figura 24 – Entidades após aplicara 1FN

Representação no ER



Um PEDIDO possui no mínimo 1 e no máximo N elementos em ITEM-DE-PEDIDO e um ITEM-DE-PEDIDO pertence a 1 e somente 1 PEDIDO, logo o relacionamento POSSUI é do tipo 1:N.

PEDIDO

Num. Ped.	Cliente	Endereço	CGC	IE	Total do Pedido	Cod. Vendedor	Prazo de Entrega	Nome do Vendedor
2610	Lopes..	R. 127...	23232	3434	2650,00	1791	20 dias	Aníbal da silva

ITEM-DO-PEDIDO

Num. Ped. *	Cód. Produto *	Unid.	Quant.	Descrição Produto	Valor Unit.	Tot. Prod.
2610	45	L	50	Alcool	5,00	250,00
2610	78	Kg	47	Cimento	30,00	1410,00
2610	21	Kg	20	Pregos	5,00	100,00
2610	98	L	15	Tinta azul	25,00	375,00
2610	90	L	15	Cola	3,00	45,00
2610	43	M	10	Arame	3,00	30,00
2610	25	F	10	Algodão	2,00	20,00
2610	65	L	5	Querosene	8,00	40,00
2610	51	M	20	Fio elétrico	13,00	260,00
2610	74	M	30	Linha 10	4,00	120,00

Figura 25 – Entidades após a 1FN

Obs: Observe que resolvemos o problema da chave primária na entidade ITEM-DO-PEDIDO, mas na entidade PEDIDOS ainda há repetição da chave.

Dependência Funcional

Um atributo B possui uma dependência funcional do atributo A se, para cada valor do atributo A, existe exatamente um único valor do atributo B. A dependência funcional é representada por $A \rightarrow B$.

Exemplo 1: Na entidade PEDIDO, o atributo PRAZO-DE-ENTREGA depende funcionalmente de NUMERO-DO-PEDIDO

$$\text{NUMERO-DO-PEDIDO} \rightarrow \text{PRAZO-DE-ENTREGA}$$

Exemplo 2: Observe os conjuntos:

CPF **Nome**

12345678909 – Jose

35698712532 – Maria

33612148789 - Paula

Observe que existe uma dependência entre os valores dos conjuntos, ou seja, nome está vinculado ao CPF, se eu estiver com número do CPF, poderei encontrar o nome da pessoa correspondente.

Essa dependência é expressa por:

CPF → Nome

Leia da seguinte maneira: com um número de “CPF” posso encontrar “nome” da pessoa, ou “nome” depende da funcionalidade do “CPF”. Para cada número de CPF, há apenas um nome.

Dependência Funcional Total (completa) e Parcial

Na ocorrência de uma chave primária concatenada, dizemos que um atributo ou conjunto de atributos depende de forma completa ou total desta chave primária concatenada, se e somente se, a cada valor da chave (e não parte dela), está associado um valor para cada atributo. Ou seja, um atributo não se apresenta com dependência completa ou total quando só depende de parte da chave primária concatenada (dependência parcial) e não dela como um todo.

Ex: dependência total – Na entidade ITEM-DO-PEDIDO, o atributo QUANTIDADE-DO-PRODUTO depende de forma total ou completa da chave primária concatenada (NUMERO-DO-PEDIDO + CODIGO-DO-PRODUTO).

Ex: dependência parcial – Na entidade ITEM-DO-PEDIDO, o atributo DESCRICAO-DO-PRODUTO depende parcialmente da chave concatenada, pois ele depende apenas de um dos atributos da chave, no caso o atributo CODIGO-DO-PRODUTO.

A dependência total ou completa só ocorre quando a chave primária for composta por vários atributos (concatenados), ou seja, em uma entidade de chave primária composta de um único atributo não ocorre este tipo de dependência.

Segunda forma normal (2FN)

Devemos observar se alguma entidade possui chave primária concatenada, e para aquelas que satisfizerem esta condição, analisar se existe algum atributo ou conjunto de atributos com dependência parcial em relação a algum elemento da chave primária concatenada.

Com a finalidade de tornar ainda mais estável o modelo de dados, a aplicação da 2FN sobre as entidades em observação geram novas entidades, que herdarão a chave parcial e todos os atributos que dependem desta chave parcial, ou seja, uma entidade para estar na 2FN não pode ter atributos com dependência parcial em relação à chave primária.

Uma relação está na FN2 quando duas condições são satisfeitas:

1 - A relação está na 1FN;

2 - Todo atributo da tabela seja dependente funcional da chave completa e não de parte da chave. Todos os atributos não-chave dependem funcionalmente de toda a chave primária.

Uma tabela com uma chave formada por apenas um atributo está automaticamente na FN2.

Exemplo: A entidade ITEM-DO-PEDIDO apresenta uma chave primária concatenada (formada por mais de um atributo) e por observação, notamos que os atributos UNIDADE-DO-PRODUTO e VALOR UNITÁRIO dependem de forma parcial do atributo CODIGO-DO-PRODUTO, que faz parte da chave primária.

Logo, devemos aplicar a 2FN sobre esta entidade. Quando aplicamos a 2FN, sobre ITEM-DO-PEDIDO, será criada a entidade PRODUTO que herdará os atributos UNIDADE-DO-PRODUTO, DESCRICAO-DO-PRODUTO e VALOR-UNITARIO e terá como chave primária o CODIGO-DO-PRODUTO.

PEDIDO

*Número do pedido
Cliente
Endereço
CGC
Inscrição estadual
Valor total do pedido
Prazo de entrega
Código do vendedor
Nome do vendedor

ITEM-DO-PEDIDO

* Número do pedido
* Código do produto
Quantidade do produto
Valor total do produto

PRODUTO

* Código do produto
Descrição do produto
Unidade do produto
Valor unitário do produto

Ao aplicarmos a 2FN, a pergunta que se pode fazer para auxiliá-lo é:

Referente a Entidade ITEM-DO-PEDIDO:

DESCRICAO-DO-PRODUTO existe em função de quem? Resp: CODIGO-DO-PRODUTO

UNIDADE-DO-PRODUTO existe em função de quem? Resp: UNIDADE-DO-PRODUTO

VALOR-UNITARIO-DO-PRODUTO existe em função de quem? Resp: UNIDADE-DO-PRODUTO

QUANTIDADE-DO-PRODUTO existe em função de quem? Resp: ITEM-DO-PEDIDO

VALOR_TOTAL_PRODUTO existe em função de quem? Resp: ITEM-DO-PEDIDO

ITEM-DO-PEDIDO

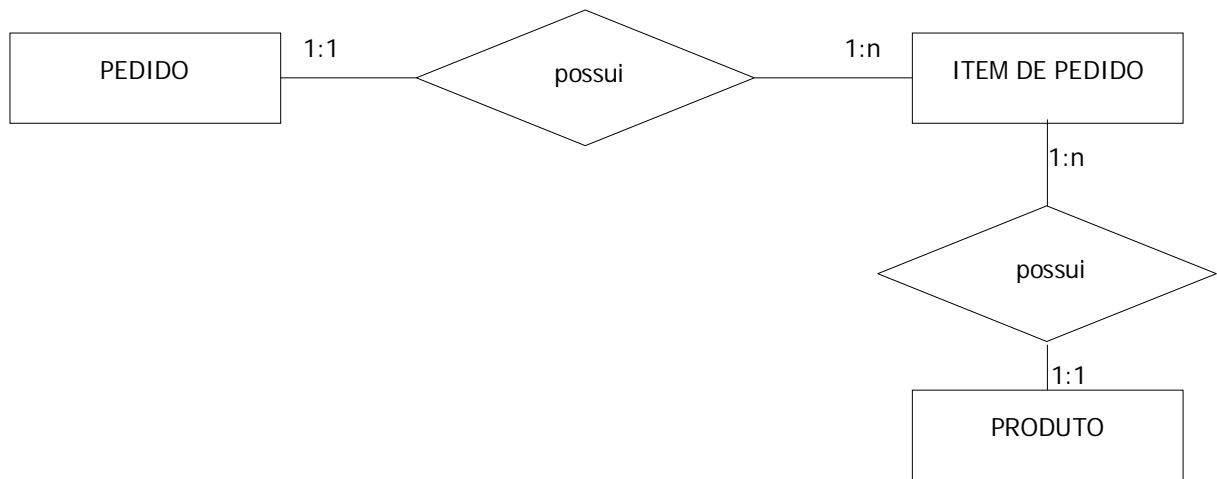
*Num. Ped. *	*Cód. Produto *	Quant.	Tot. Prod.
2610	45	50	250,00
2610	78	47	1410,00
2610	21	20	100,00
2610	98	15	375,00
2610	90	15	45,00
2610	43	10	30,00
2610	25	10	20,00
2610	65	5	40,00
2610	51	20	260,00
2610	74	30	120,00

PRODUTO

Cód. Produto *	Unid.	Descrição Produto	Valor Unit.
45	L	Álcool	5,00
78	Kg	Cimento	30,00
21	Kg	Pregos	5,00
98	L	Tinta azul	25,00
90	L	Cola	3,00
43	M	Arame	3,00
25	F	Algodão	2,00
65	L	Querosene	8,00
51	M	Fio elétrico	13,00
74	M	Linha 10	4,00

Figura 26 – Entidades ITEM-DO-PEDIDO e PRODUTO após a 2FN

Representação no ER



Um produto participa de no mínimo 1 e no máximo N elementos de ITEM-DE-PEDIDO e um ITEM-DE-PEDIDO só pode conter 1 e somente 1 PRODUTO. Logo, o novo relacionamento criado é do tipo N:1.

Chave Estrangeira

Chave estrangeira (FK - Foreign Key) é a chave formada através de um relacionamento com a chave primária de outra tabela. Define um relacionamento entre as tabelas e pode ocorrer repetidas vezes. Caso a chave primária seja composta (formada por 2 ou mais atributos) na origem, a chave estrangeira também o será.

Ex: Na tabela ITEM-DO-PEDIDO, o atributo CODIGO-DO-PRODUTO é chave estrangeira, pois este está relacionado com o campo de mesmo nome, e que é chave primária na tabela PRODUTO.

Em outro exemplo, na figura abaixo, o atributo CdDepto da tabela Funcionário fornece o número do departamento para o qual cada empregado trabalha, portanto seu valor em toda tupla Funcionário deve corresponder ao valor de cdDepto na tabela Depto.

* CdDepto	NmDepto	Ramal
D1	ADMINISTRACAO	221
D2	OFICINA	235
D3	SERVICOS GERAIS	243
D4	VENDAS	258

NrMatric	NmFunc	DtAdm	Sexo	CdCargo	CdDepto*
1001	JOAO SAMPAIO	10/08/93	M	C2	D2
1004	LUCIO TORRES	02/03/94	M	C2	D2
1034	ROBERTO PEREIRA	23/05/92	M	C3	D1

No exemplo acima, na tabela FUNCIONÁRIO, o atributo cdDepto refere-se ao departamento para o qual um funcionário trabalha, portanto, indicamos cdDepto como chave estrangeira de FUNCIONÁRIO, fazendo referência à relação funcionário. Isso significa que todo valor de cdDepto em qualquer tupla da relação FUNCIONÁRIO deve corresponder a um valor da chave primária de DEPTO, ou o valor de cdDepto pode ser **null** (nulo) se o empregado não pertencer a um departamento.

Exercício:

Continuando o exercício aplicado quando da explicação da 1FN, aplique agora a 2ª forma normal, definindo inclusive o diagrama ER e a cardinalidade.

Dependência Funcional Transitiva

Quanto um atributo ou conjunto de atributos A depende de outro atributo B que não pertence à chave primária, mas é dependente funcionalmente desta, dizemos que A é dependente transitivo de B.

Ex: dependência transitiva – Na entidade PEDIDO, os atributos ENDERECO, CGC e INSCRICAO-ESTADUAL são dependentes transitivos do atributo CLIENTE. Nesta mesma entidade, o atributo NOME-DO-VENDEDOR é dependente transitivo do atributo CODIGO-DO-VENDEDOR.

Terceira Forma Normal (3FN)

Uma entidade está na 3FN se nenhum de seus atributos possui dependência transitiva em relação a outro atributo da entidade que não participe da chave primária, ou seja, não exista nenhum atributo intermediário entre a chave primária e o próprio atributo observado.

Ao retirarmos a dependência transitiva, devemos criar uma nova entidade que contenha os atributos que dependem transitivamente de outro e a sua chave primária é o atributo que causou esta dependência.

Além de não conter atributos com dependência transitiva, entidades na 3FN não devem conter atributos que sejam o resultado de algum cálculo sobre outro atributo, que de certa forma pode ser encarada como uma dependência funcional.

Também devemos observar que todo atributo deve ser dependente da chave primária, ou seja, se houver algum atributo que não tenha dependência funcional da chave primária, este deve ser movido para uma nova entidade, deixando apenas na entidade original a sua chave.

Ex: na entidade PEDIDO, podemos observar que o atributo NOME-DO-VENDEDOR depende transitivamente do atributo CODIGO-DO-VENDEDOR que não pertence à chave primária. Para eliminarmos esta anomalia, devemos criar a entidade VENDEDOR, com o atributo NOME-DO-VENDEDOR e tendo como chave primária o atributo CODIGO-DO-VENDEDOR.

Encontramos ainda o conjunto de atributos formados por ENDERECO, CGC e INSCRICAO-ESTADUAL que dependem transitivamente do atributo NOME-DO-CLIENTE. Neste caso, devemos criar um atributo chamado CODIGO-DO-CLIENTE que funcionará melhor como chave primária do que NOME-DO-CLIENTE, deixando este último como simples atributo da entidade CLIENTE.

PEDIDO

*Número do pedido
Valor total do pedido
Prazo de entrega
CGC *
Código do vendedor *

ITEM-DO-PEDIDO

* Número do pedido *
* Código do produto *
Quantidade do produto
Valor total do produto

PRODUTO

* Código do produto
Descrição do produto
Unidade do produto
Valor unitário do produto

CLIENTE

* CGC
Nome Cliente
Endereço
Inscrição

VENDEDOR

* Código do vendedor
Nome do vendedor

PEDIDO

*Num. Ped.	Total do Pedido	Prazo de Entrega	CGC *	Código do Vendedor *
2610	2650,00	20 dias	23232	1727

CLIENTE

*CGC	Nome Cliente	Endereço	IE
23232	Lopes..	R. 127...	3434

VENDEDOR

* Cod. Vendedor	Nome do Vendedor
1791	Aníbal da silva

ITEM-DO-PEDIDO

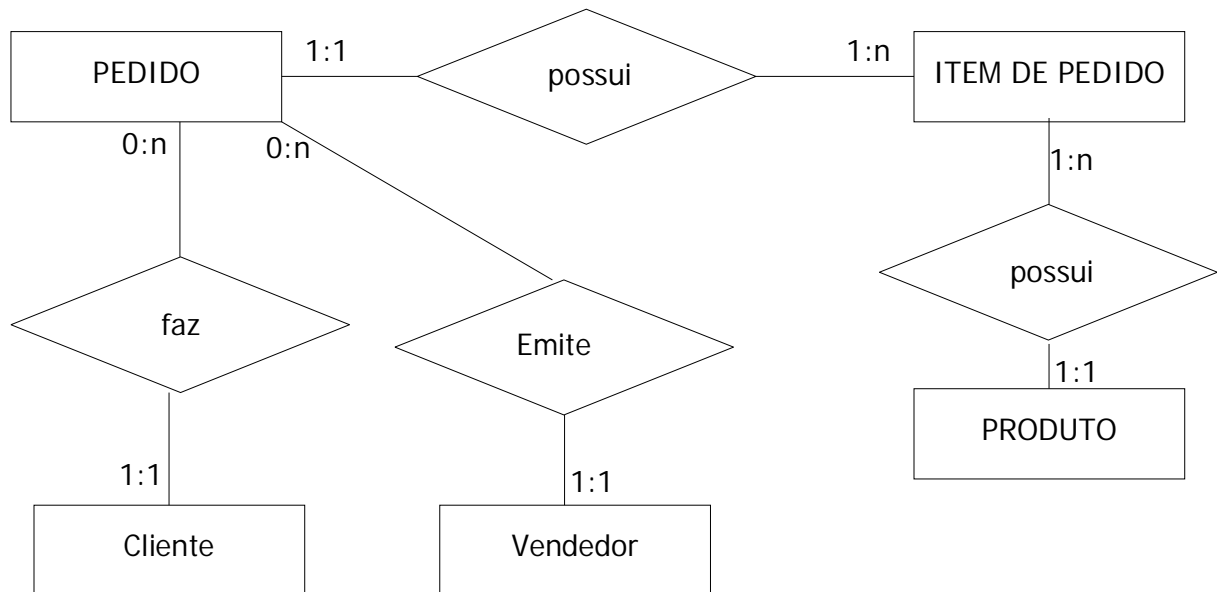
Num. Ped. *	Cód. Produto *	Quant.	Tot. Prod.
2610	45	50	250,00
2610	78	47	1410,00
2610	21	20	100,00
2610	98	15	375,00
2610	90	15	45,00
2610	43	10	30,00
2610	25	10	20,00
2610	65	5	40,00
2610	51	20	260,00
2610	74	30	120,00

PRODUTO

Cód. Produto *	Unid.	Descrição Produto	Valor Unit.
45	L	Álcool	5,00
78	Kg	Cimento	30,00
21	Kg	Pregos	5,00
98	L	Tinta azul	25,00
90	L	Cola	3,00
43	M	Arame	3,00
25	F	Algodão	2,00
65	L	Querosene	8,00
51	M	Fio elétrico	13,00
74	M	Linha 10	4,00

Figura 27 – Entidades após a 3FN

Representação no ER



Um pedido só é feito por um e somente um CLIENTE e um CLIENTE pode fazer de zero até N elementos de PEDIDO. Um PEDIDO só é tirado por um e somente um VENDEDOR e um VENDEDOR pode tirar de zero a N elementos de PEDIDO.

Roteiro de aplicação da Normalização

1 – Aplicação da 1FN

- Identificar a chave primária da tabela original.
- Decompor a entidade em uma ou mais entidades, movendo os atributos multivalorados para a nova entidade.
- Escolher um novo atributo na nova entidade para ser a chave primária; se não for possível, crie um novo campo. Este será concatenado com a chave primária da tabela original.
- Estabelecer o relacionamento e a cardinalidade entre as entidades.

2 – Aplicação da 2FN

- Para entidades que contenham chaves primárias concatenadas, destacar os atributos que tenham dependência parcial em relação à chave primária concatenada.
- Criar uma nova entidade que conterá estes atributos, e que terá como chave primária os atributos dos quais se tenha dependência parcial.
- Serão criadas tantas entidades quanto forem os atributos da chave primária concatenada, que gerem dependência parcial.
- Estabelecer o relacionamento e a cardinalidade entre as entidades.

3 – Aplicação da 3FN

- Verificar se existem atributos que sejam dependentes transitivos de outros que não pertencem à chave primária, sendo ela concatenada ou não, bem como atributos que sejam dependentes de cálculo realizado através de outros atributos.
- Destacar os atributos com dependência transitiva, gerando uma nova entidade com este atributo e cuja chave primária é o atributo que originou a dependência. Manter na tabela original apenas o atributo chave da nova tabela gerada.
- Eliminar os atributos obtidos através de cálculos realizados a partir de outros atributos.

- Destacar os atributos que não tenham dependência funcional da chave primária, gerando uma nova entidade. Manter na tabela original apenas o atributo chave da nova tabela gerada.

Exercício:

Entidade LOCACAO

Cód. locação	Cliente	Endereço do cliente	Tel. Do cliente	Data da Locação	Cód. Mídia	Descrição Da Mídia	Valor unit.	Valor Total da Locação	Data de Devolução
1	Ana...	Rua XyZ	4589-1..	01/03/09	1	Mad Max ...	4,50	10,00	03/03/09
1	Ana...	Rua XyZ	4589-1..	01/03/09	6	O senhor dos...	5,50	10,00	02/03/09
2	Patrícia..	Rua ABC	7856-...	05/03/09	1	Mad Max ...	4,50	15,00	07/03/09
2	Patrícia..	Rua ABC	7856-...	05/03/09	4	A hora do....	3,00	15,00	08/03/09
2	Patrícia..	Rua ABC	7856-...	05/03/09	8	Indiana Jones	2,50	15,00	08/03/09
2	Patrícia..	Rua ABC	7856-...	05/03/09	7	King Kong 2005..	5,00	15,00	06/03/09

Aplice as 3 formas normais na entidade acima, definindo inclusive o diagrama ER e a cardinalidade.

Entidades Na Forma Normal Final (Normalizadas)

O processo de normalização leva ao refinamento das entidades, retirando delas grande parte das redundâncias e inconsistências. Naturalmente, para que haja uma associação entre entidades é preciso que ocorram redundâncias mínimas de atributos que evidenciam estes relacionamentos. Sem estas redundâncias não haveria relacionamento entre entidades.

Operações relacionais e Álgebra relacional

Fonte do material deste capítulo:

Antonio Cesar de Barros Munari - <http://www.pucrs.campus2.br/~jiani/bd/OpRelacional.pdf>

A discussão sobre algumas operações básicas de álgebra relacional realizada a seguir considera um banco de dados composto pelas seguintes relações:

Funcionário	Cargo	Depto
* NrMatric NmFunc DtAdm Sexo CdCargo * CdDepto *	* CdCargo NmCargo VrSalario	* Cd Depto NmDepto Ramal

Campos com * do lado esquerdo representam a **chave primária**, enquanto campos com * do lado direito representam a **chave estrangeira**.

cargo

*CdCargo	NmCargo	VrSalário
C1	COZINHEIRA	350
C3	AUX. ESCRITÓRIO	450
C7	VIGIA	400
C2	MECANICO	750
C5	GERENTE	2300
C4	ESCRITURARIO	600

depto

* CdDeppto	NmDeppto	Ramal
D1	ADMINISTRACAO	221
D2	OFICINA	235
D3	SERVICOS GERAIS	243
D4	VENDAS	258

funcionário

NrMatric	NmFunc	DtAdm	Sexo	CdCargo	CdDeppto*
1001	JOAO SAMPAIO	10/08/93	M	C2	D2
1004	LUCIO TORRES	02/03/94	M	C2	D2
1034	ROBERTO PEREIRA	23/05/92	M	C3	D1
1021	JOSE NOGUEIRA	10/11/94	M	C3	D1
1029	RUTH DE SOUZA	05/01/92	F	C1	D3
1095	MARIA DA SILVA	03/09/92	F	C4	D1
1023	LUIZ DE ALMEIDA	12/01/93	M	C2	D2
1042	PEDRO PINHEIRO	29/07/94	M	C4	D1
1048	ANA SILVEIRA	01/06/93	F	C5	D1
1015	PAULO RODRIGUES	17/08/92	M	C2	D2

Figura 28 - Tabelas utilizadas nos exemplos de álgebra relacional

Estamos interessados em obter informações armazenadas nesse banco de dados, e para isso deveremos formular expressões em álgebra relacional combinando apenas algumas operações primitivas clássicas, que serão apresentadas uma a uma de forma simplificada em termos de seu significado prático e notação.

Toda operação relacional opera (age) sobre um ou mais conjuntos de dados e fornece como resultado um novo conjunto. Devido a essa característica, podemos combinar mais de uma operação relacional em uma única expressão algébrica, fazendo com que o resultado de uma operação seja utilizado como entrada para outra operação, aumentando com isso grandemente o poder dessa linguagem de consulta.

Suponha que, inicialmente, precisamos obter o nome completo de todos os funcionários de nosso banco de dados. Para isso será necessário executar uma operação chamada **Projeção**.

Projeção (π)

Geralmente indicada na literatura por π (a letra grega pi) produz um conjunto onde há um elemento para cada elemento do conjunto de entrada, sendo que a estrutura dos membros do conjunto resultante é definida nos argumentos da operação. Pode ser entendida como uma operação que filtra as colunas de uma tabela. Por operar sobre apenas um conjunto de entrada, a projeção é classificada como uma operação unária.

Ex.: π NmFunc (funcionário)

Essa expressão produz um conjunto contendo um elemento para cada funcionário, e cada elemento contém apenas a informação referente a NmFunc da relação funcionário original.

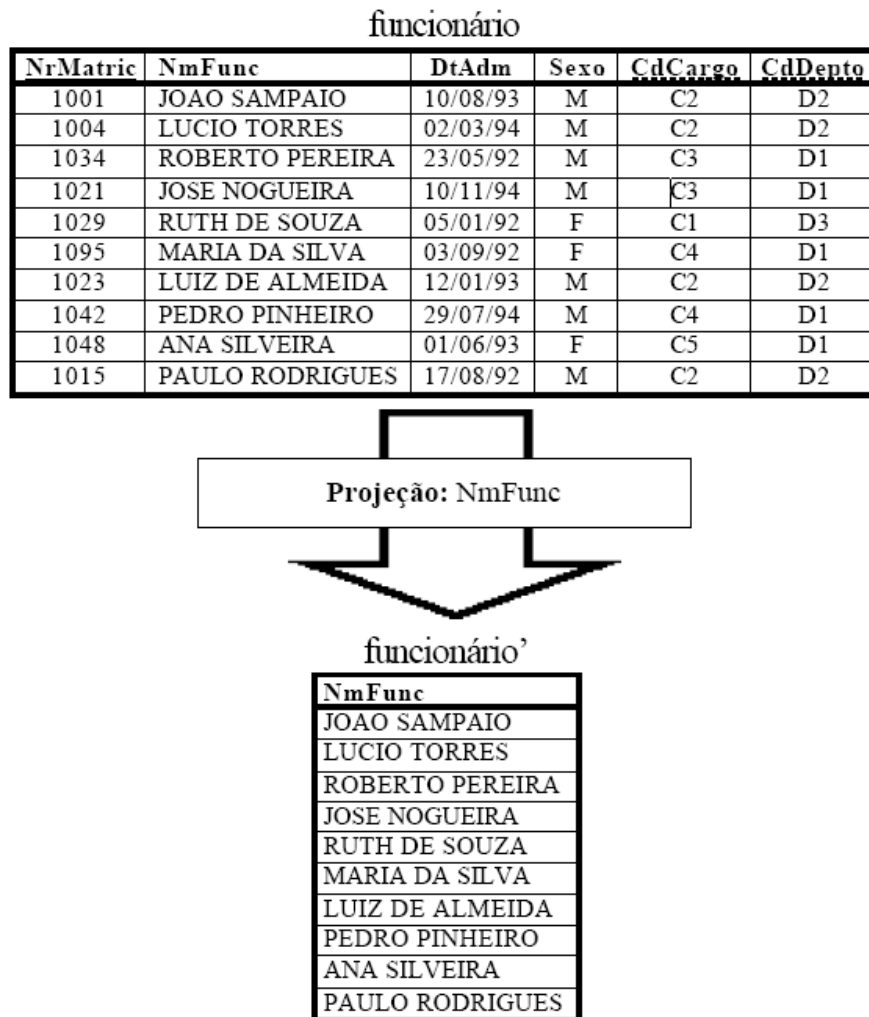


Figura 29 – Resultado da projeção

Agora estamos interessados em identificar todos os funcionários de sexo masculino existentes no banco de dados. É uma situação que não podemos resolver com projeções apenas, uma vez que deveremos descartar elementos do conjunto inicial. Para casos desse tipo existe uma operação relacional chamada **Seleção**.

Seleção ou Restrição (σ)

Indicada por σ (a letra grega sigma), é uma operação que para um conjunto inicial fornecido como argumento, produz um subconjunto estruturalmente idêntico, mas apenas com os elementos do conjunto original que atendem a uma determinada condição (também chamada de predicado). A seleção pode ser entendida como uma operação que filtra as linhas de uma tabela, e é também uma operação unária, já que opera sobre um único conjunto de dados.

Ex.: $\sigma_{\text{Sexo} = 'M'}(\text{funcionário})$

Produz o conjunto dos elementos de funcionário que atendem ao predicado [Sexo = 'M'], ou seja, representa um subconjunto dos funcionários para o qual essa condição é avaliada como verdadeira.

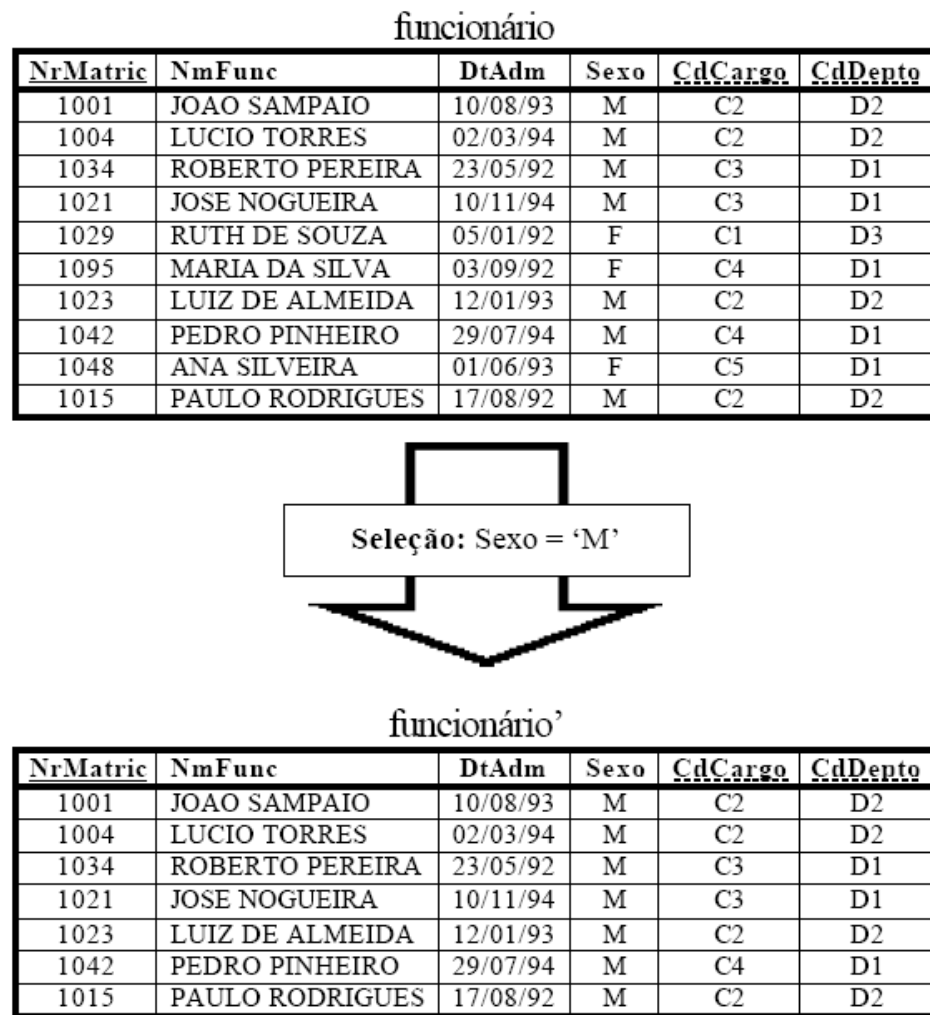
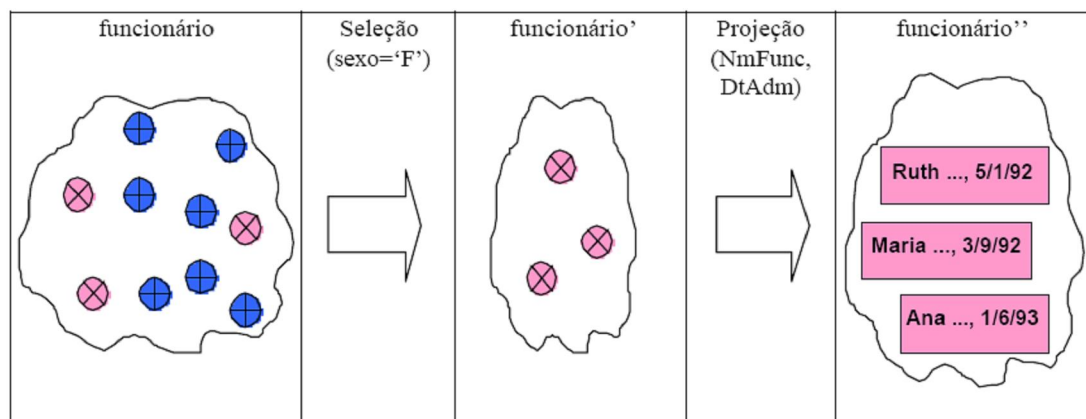


Figura 30 – Resultado da Seleção

No caso de querermos descobrir o nome completo e a data de admissão de todos os funcionários do sexo feminino existentes na empresa, será necessário combinar uma projeção com uma seleção. Isso porque se decidirmos projetar as colunas desejadas diretamente a partir da relação funcionário estaremos considerando também os elementos do sexo masculino, o que não queremos. Como a projeção não permite descartar linhas, apenas colunas, deveremos fornecer a essa operação o subconjunto resultante de uma filtragem (seleção) da relação de funcionários original, como mostram as duas figuras a seguir, que representam as relações e as operações de duas maneiras diferentes.



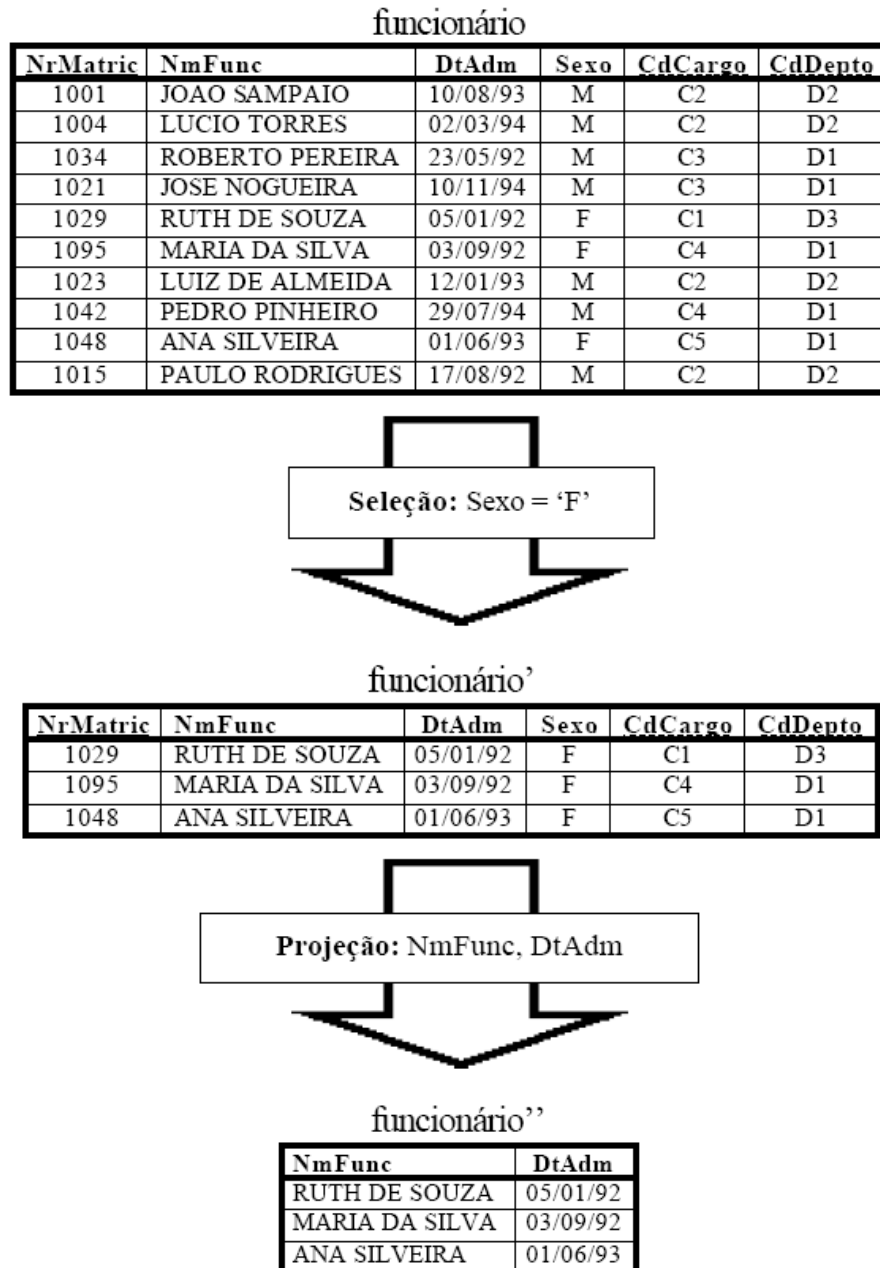


Figura 31 – Utilizando projeção (π) em conjunto com seleção (σ)

Assim, a expressão que atende nossos objetivos nesse caso é:

π NmFunc, DtAdm (σ Sexo = 'F' (funcionário))

Cabendo observar que devido ao aninhamento das operações está implícito que primeiro será executada a seleção e depois a projeção, sendo que nesse exemplo não poderíamos inverter essa ordem (você poderia explicar por quê?). Por esse motivo, dizemos que a álgebra relacional é uma **linguagem procedural**, já que requer alguma definição quanto à ordem em que as operações serão realizadas. Linguagens em que apenas mencionamos o resultado desejado, sem fazer menção alguma à forma como isso deve ser feito são chamadas de **linguagens não-procedurais**.

Suponha agora que precisamos obter o nome completo, a data de admissão e o salário de cada funcionário cadastrado. Para essa consulta temos um fato novo, que é a referência a colunas de mais de uma tabela, uma vez que o nome e a data de admissão fazem parte da relação funcionário, enquanto que o salário existe apenas em cargos. Isso é problemático, pois as duas operações que conhecemos até o momento são unárias, e temos necessidade de combinar os dados de mais de uma relação. Para situações como essa existe uma operação chamada **Produto Cartesiano**.

Produto Cartesiano (conjunto1 x conjunto2)

A notação geralmente adotada (na forma 'conjunto1 x conjunto2') para representar essa operação binária indica bem a sua natureza: **o resultado do produto cartesiano de duas tabelas é uma terceira tabela contendo todas as combinações possíveis entre os elementos das tabelas originais.** Essa tabela resultante possuirá um número de colunas que é igual à soma das quantidades de colunas das duas tabelas iniciais, e um número de linhas igual ao produto do número de suas linhas.

Portanto, se fizermos o produto cartesiano de uma tabela A que possua 2 colunas e 2 linhas com uma tabela B onde existem 3 colunas e 3 linhas, a tabela resultante terá $2+3=5$ colunas e $2*3=6$ linhas. Assim, cada linha dessa tabela corresponderá à concatenação de uma linha da primeira tabela com uma linha da segunda.

Tabela A		Tabela B		
*CampoA1	CampoA2	*CampoB1	CampoB2	CampoA1 *
A	AAAAAAAAA	V1	Desc1	A
B	BBBBBBBBB	V2	Desc2	A
		V3	Desc3	B

Produto Cartesiano A x B				
CampoA1	CampoA2	CampoB1	CampoB2	CampoA1
A	AAAAAAAAA	V1	Desc1	A
A	AAAAAAAAA	V2	Desc2	A
A	AAAAAAAAA	V3	Desc3	B
B	BBBBBBBBB	V1	Desc1	A
B	BBBBBBBBB	V2	Desc2	A
B	BBBBBBBBB	V3	Desc3	B

Figura 32 – Produto Cartesiano entre a tabela A e a tabela B

O produto cartesiano não é muito usado como um fim em si mesmo, ou seja, dificilmente estaremos interessados em saber quais são todas as combinações possíveis entre as linhas de duas tabelas, pois a utilidade prática desse tipo de conhecimento é muito discutível. Entretanto, é a única forma primitiva de que dispomos para fundir informações de duas tabelas heterogêneas para posterior processamento. Nesse caso, tipicamente será necessário executar uma Seleção sobre o resultado do Produto Cartesiano, de maneira a descartar as combinações inválidas entre as linhas das tabelas originais.

Ex.: π NmFunc, DtAdm, VrSalário (σ funcionário.CdCargo = cargo.CdCargo (funcionário x cargo))

Observe que primeiro é produzido o **produto cartesiano** correspondente a todas as combinações possíveis entre funcionários e cargos. Essa relação vai conter linhas onde um funcionário estará associado a cargos que não são o seu, e devemos então aplicar um filtro (uma **seleção**) para gerar um subconjunto apenas com as combinações logicamente válidas (aquelas em que a chave estrangeira CdCargo de funcionário tem valor igual à chave primária CdCargo de cargo).

Como temos nesse subconjunto duas colunas com o mesmo nome (CdCargo que veio de funcionário e CdCargo proveniente de cargo), sempre que precisarmos mencionar uma delas será necessário especificar exatamente a qual das duas colunas estamos nos referindo, senão teremos uma situação ambígua, formalmente inaceitável. Dizemos, nesse caso, que é necessário qualificar a coluna, e isso é feito escrevendo o nome da relação original antes do nome da coluna, separando-os por um ponto, ou seja, <nome-da-relação>.<nome-da-coluna>. É por esse motivo que escrevemos o predicado da seleção como sendo funcionário.CdCargo = cargo.CdCargo.

Finalmente, a **projeção** é realizada a partir desse subconjunto, fornecendo os dados inicialmente desejados.

funcionário

NrMatric	NmFunc	DtAdm	Sexo	CdCargo	CdDepto
1001	JOAO SAMPAIO	10/08/93	M	C2	D2
1004	LUCIO TORRES	02/03/94	M	C2	D2
1034	ROBERTO PEREIRA	23/05/92	M	C3	D1
1021	JOSE NOGUEIRA	10/11/94	M	C3	D1
1029	RUTH DE SOUZA	05/01/92	F	C1	D3
1095	MARIA DA SILVA	03/09/92	F	C4	D1
1023	LUIZ DE ALMEIDA	12/01/93	M	C2	D2
1042	PEDRO PINHEIRO	29/07/94	M	C4	D1
1048	ANA SILVEIRA	01/06/93	F	C5	D1
1015	PAULO RODRIGUES	17/08/92	M	C2	D2

cargo

CdCargo	NmCargo	VrSalário
C1	COZINHEIRA	350
C3	AUX. ESCRIT.	450
C7	VIGIA	400
C2	MECANICO	750
C5	GERENTE	2300
C4	ESCRITURARIO	600

Produto cartesiano: funcionário x cargo

funcionário_cargo

NrMatric	NmFunc	DtAdm	Sexo	CdCargo	CdDepto	CdCargo	NmCargo	VrSalario
1001	JOAO SAMPAIO	10/08/93	M	C2	D2	C1	COZINHEIRA	350
1001	JOAO SAMPAIO	10/08/93	M	C2	D2	C3	AUX. ESCRITÓRIO	450
1001	JOAO SAMPAIO	10/08/93	M	C2	D2	C7	VIGIA	400
1001	JOAO SAMPAIO	10/08/93	M	C2	D2	C2	MECANICO	750
1001	JOAO SAMPAIO	10/08/93	M	C2	D2	C5	GERENTE	2300
1001	JOAO SAMPAIO	10/08/93	M	C2	D2	C4	ESCRITURARIO	600
...
1021	JOSE NOGUEIRA	10/11/94	M	C3	D1	C1	COZINHEIRA	350
1021	JOSE NOGUEIRA	10/11/94	M	C3	D1	C3	AUX. ESCRITÓRIO	450
1021	JOSE NOGUEIRA	10/11/94	M	C3	D1	C7	VIGIA	400
...
1015	PAULO RODRIGUES	17/08/92	M	C2	D2	C4	ESCRITURARIO	600

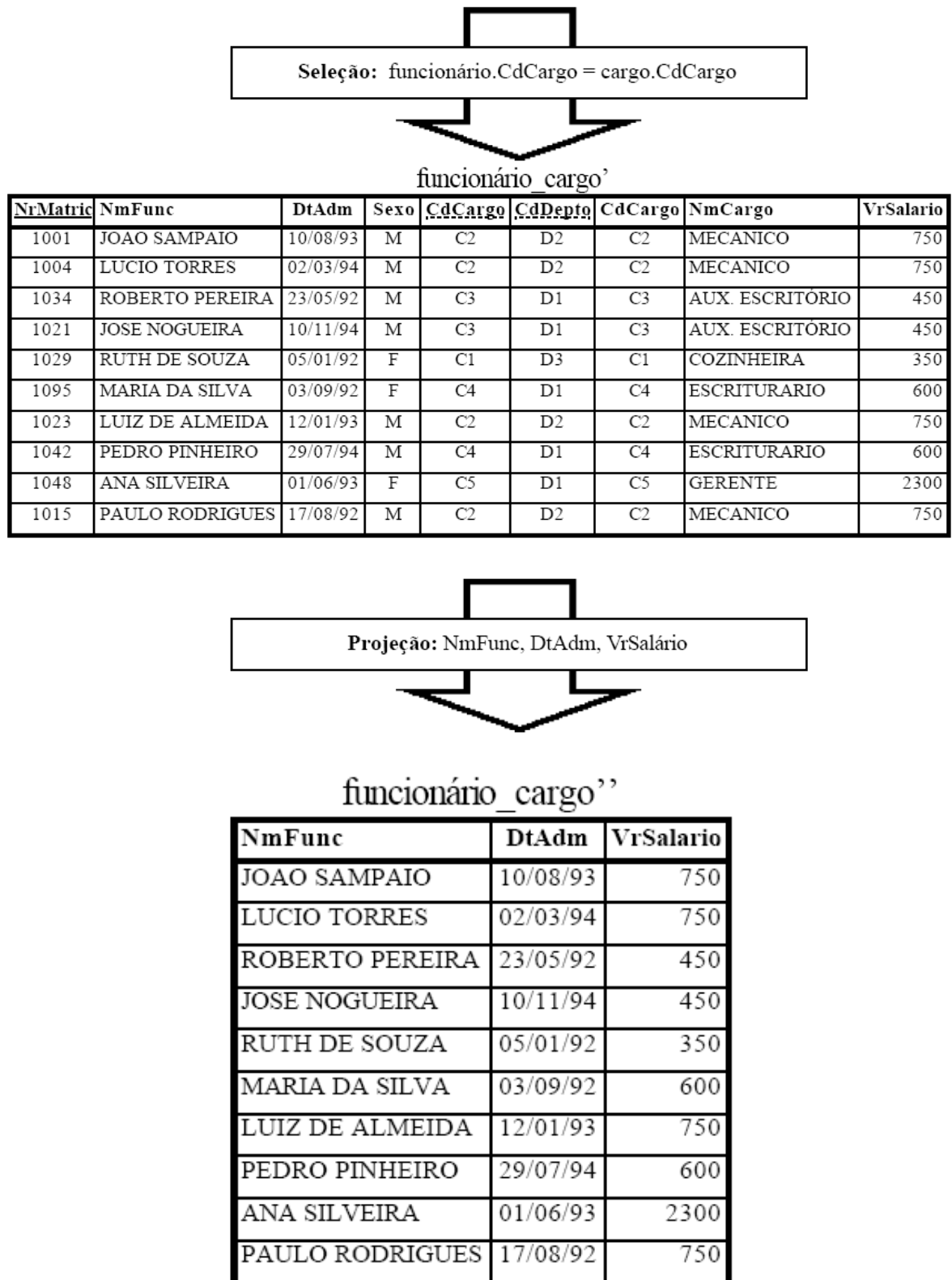


Figura 33 - π NmFunc, DtAdm, VrSalário (σ funcionário.CdCargo = cargo.CdCargo (funcionário x cargo))

Caso desejemos obter uma variação ligeiramente diferente dessa consulta, acrescentando a restrição de que precisamos dos dados apenas dos funcionários do sexo masculino, teríamos a seguinte expressão algébrica:

π NmFunc, DtAdm, VrSalário (σ funcionário.CdCargo = cargo.CdCargo \wedge Sexo = 'M' (funcionário x cargo))

onde o símbolo \wedge presente no predicado representa o conectivo lógico "E".

Portanto, com apenas 3 operações relacionais básicas foi possível extrair do banco de dados de exemplo várias informações importantes, representativas de uma grande parcela das consultas que um sistema gerenciador de bancos de dados deve processar. As consultas realizadas foram:

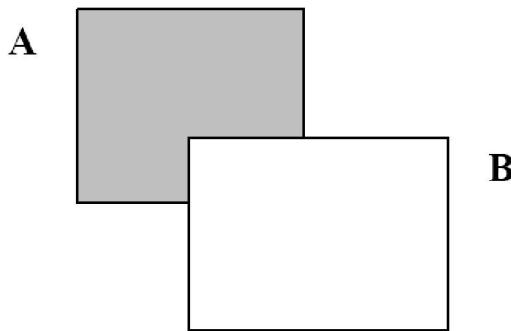
1. Obter o nome completo de todos os funcionários;
2. Identificar todos os funcionários do sexo masculino;
3. Obter o nome completo e a data de admissão de todos os funcionários do sexo feminino;
4. Obter o nome completo, a data de admissão e o salário de todos os funcionários;
5. Descobrir o nome completo, a data de admissão e o salário de todos os funcionários do sexo masculino.

Exercício: Resolva passo a passo:

π CampoA1, CampoA2, CampoB1, CampoB2 (σ A.CampoA1 = B.CampoA1 (A x B))

Diferença entre conjuntos: $A - B$

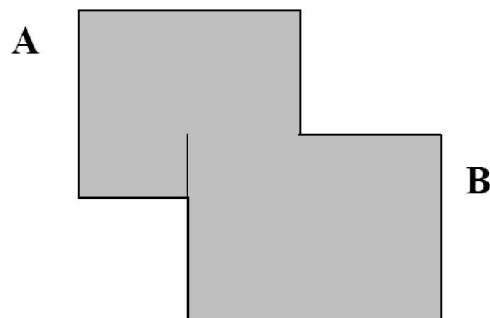
É uma operação primitiva que requer como operando duas tabelas união-compativeis, ou seja, estruturalmente idênticas. O resultado é uma tabela que possui todas as linhas que existem na primeira tabela e não existem na segunda.



Observe que $A - B$ é diferente de $B - A$.

União: $A \cup B$

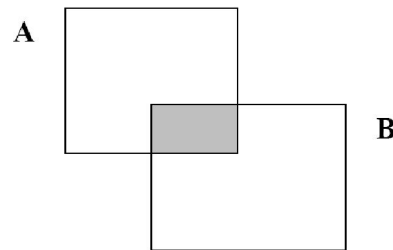
Esta operação primitiva também requer como operando tabelas união-compativeis. Produz como resultado uma tabela que contém todas as linhas da primeira tabela seguidas de todas as linhas da segunda tabela. A tabela resultante possui a mesma quantidade de colunas que as tabelas originais, e tem um número de linhas que é no máximo igual à soma das linhas das tabelas fornecidas como operando, já que as linhas que são comuns a ambas as tabelas aparecem uma única vez no resultado.



Observe que $A \cup B = B \cup A$.

Intersecção: $A \cap B$

Esta é uma operação adicional que produz como resultado uma tabela que contém, sem repetições, todos os elementos que são comuns às duas tabelas fornecidas como operando. As tabelas devem ser união-compatíveis. O mesmo efeito pode ser obtido fazendo-se uma combinação de diferenças entre conjuntos



$$A \cap B = A - (A - B)$$

ou com uniões e diferenças

$$A \cap B = (A \cup B) - (A - B) - (B - A) \text{ ou ainda}$$

$$A \cap B = (A \cup B) - ((A - B) \cup (B - A))$$

Junção: $A \bowtie B$

É uma operação que produz uma combinação entre as linhas de uma tabela com as linhas correspondentes de outra tabela, sendo em princípio correspondente a uma seleção pelos atributos de relacionamento sobre um produto cartesiano dessas tabelas:

$$A \bowtie B = \sigma_{A.chave1 = B.chave2} (A \times B)$$

A operação de junção foi criada justamente porque esse tipo de combinação de tabelas é de uso muito comum, facilitando com isso a escrita de expressões. A tabela resultante de uma junção tem todas as colunas da primeira tabela e todas da segunda tabela. Isso faz com que os valores dos campos utilizados como critério para a correspondência entre as linhas apareça duplicado, já que um vem da primeira tabela e outro da segunda. Existe uma variação da junção, chamada junção natural, que fornece o mesmo resultado, mas sem essa repetição de valores: uma das colunas correspondentes aos atributos de relacionamento é descartada.

Resumo

<i>Símbolo</i>	<i>Operação</i>	<i>Sintaxe</i>	<i>Tipo</i>
σ	Seleção / Restrição	$\sigma_{\text{condição}} (\text{Relação})$	Primitiva
π	Projeção	$\pi_{\text{expressões}} (\text{Relação})$	Primitiva
\cup	União	$\text{Relação1} \cup \text{Relação2}$	Primitiva
\cap	Intersecção	$\text{Relação1} \cap \text{Relação2}$	Adicional
-	Diferença de conjuntos	$\text{Relação1} - \text{Relação2}$	Primitiva
\times	Produto cartesiano	$\text{Relação1} \times \text{Relação2}$	Primitiva
\bowtie	Junção	$\text{Relação1} \bowtie \text{Relação2}$	Adicional

Exercícios

Com base nas tabelas Funcionário, Cargo e Depto apresentadas neste material, elaborar as expressões da álgebra relacional que obtenham:

- 1) Todos os funcionários do departamento 'D1'.
- 2) O nome e a matrícula de todos os funcionários do departamento 'D1'.
- 3) A matrícula e o nome do respectivo departamento de todos os funcionários.
- 4) O nome dos funcionários que ganham mais de \$500.
- 5) O ramal do funcionário 'ANA SILVEIRA'.
- 6) Os nomes de todos os funcionários com cargo de 'MECANICO'.
- 7) Os nomes de todos os funcionários que trabalham no mesmo departamento que 'JOSE NOGUEIRA'.

Respostas na próxima página (resista à tentação... não as veja antes de ter respondido!!!!)

cargo

*CdCargo	NmCargo	VrSalário
C1	COZINHEIRA	350
C3	AUX. ESCRITÓRIO	450
C7	VIGIA	400
C2	MECANICO	750
C5	GERENTE	2300
C4	ESCRITURARIO	600

depto

* CdDeppto	NmDeppto	Ramal
D1	ADMINISTRACAO	221
D2	OFICINA	235
D3	SERVICOS GERAIS	243
D4	VENDAS	258

funcionário

NrMatric	NmFunc	DtAdm	Sexo	CdCargo	CdDeppto*
1001	JOAO SAMPAIO	10/08/93	M	C2	D2
1004	LUCIO TORRES	02/03/94	M	C2	D2
1034	ROBERTO PEREIRA	23/05/92	M	C3	D1
1021	JOSE NOGUEIRA	10/11/94	M	C3	D1
1029	RUTH DE SOUZA	05/01/92	F	C1	D3
1095	MARIA DA SILVA	03/09/92	F	C4	D1
1023	LUIZ DE ALMEIDA	12/01/93	M	C2	D2
1042	PEDRO PINHEIRO	29/07/94	M	C4	D1
1048	ANA SILVEIRA	01/06/93	F	C5	D1
1015	PAULO RODRIGUES	17/08/92	M	C2	D2

Respostas

1-) Todos os funcionários do departamento 'D1'.

$$\sigma_{cdDeppto = 'D1'} (\text{funcionarios})$$

2-) O nome e a matrícula de todos os funcionários do departamento 'D1'.

$$\pi_{nrMatric, nmFunc} (\sigma_{cdDeppto = 'D1'} (\text{funcionarios}))$$

3-) A matrícula e o nome do respectivo departamento de todos os funcionários.

$$\pi_{nrMatric, nmDeppto} (\sigma_{funcionarios.cdDeppto = depto.cdDeppto} (\text{funcionarios x depto}))$$

Ou

$$\pi_{nrMatric, nmDeppto} (\text{funcionarios |x| depto})$$

4) O nome dos funcionários que ganham mais de \$500.

$$\pi_{nmFunc} (\sigma_{funcionarios.cdCargo = Cargo.cdCargo \wedge vrSalario > 500} (\text{funcionarios x Cargo}))$$

Ou

$$\pi_{nmFunc} (\sigma_{vrSalario > 500} (\text{funcionarios |x| Cargo}))$$

5) O ramal do funcionário 'ANA SILVEIRA'.

$$\pi_{ramal} (\sigma_{funcionarios.cdDeppto = depto.cdDeppto \wedge nmFunc = 'ANA SILVEIRA'} (\text{funcionarios x depto}))$$

6) Os nomes de todos os funcionários com cargo de 'MECANICO'.

$$\pi_{nmFunc} (\sigma_{funcionarios.cdCargo = Cargo.cdCargo \wedge cargo = 'MECANICO'} (\text{funcionarios x Cargo}))$$

7) Os nomes de todos os funcionários que trabalham no mesmo departamento que 'JOSE NOGUEIRA'.

$$\pi_{nmFunc} (\sigma_{cdDeppto = (\pi_{cdDeppto} (\sigma_{nmFunc = 'JOSE NOGUEIRA'} (\text{funcionarios})))} (\text{funcionarios}))$$

Executando a 7 passo a passo:

$$(\pi_{cdDeppto} (\sigma_{nmFunc = 'JOSE NOGUEIRA'} (\text{funcionarios}))) = 'D1'$$

$$\pi_{nmFunc} (\sigma_{cdDeppto = 'D1'} (\text{funcionarios}))$$

Integridade Referencial

A restrição de integridade de entidade estabelece que nenhum valor de chave primária pode ser null. Isso porque o valor da chave primária é usado para identificar as tuplas individuais em uma relação. Ter valores null para a chave primária implica não podermos identificar alguma tupla. Por exemplo, se duas ou mais tuplas tiverem **null** em suas chaves primárias, poderemos não ser capazes de distingui-las, se tentarmos fazer referência a elas por intermédio de outras relações.

As restrições de chave e as de integridade de entidade são especificadas na própria relação. A restrição de integridade referencial é classificada entre duas relações e é usada para manter a consistência entre as tuplas nas duas relações. Informalmente, a restrição de integridade referencial declara que uma tupla em uma relação, que faz referência a outra relação, deve referir a uma tupla existente nessa relação.

Por exemplo, na figura abaixo, o atributo **CdDeppto** da tabela **Funcionário** fornece o número do departamento para o qual cada empregado trabalha, portanto seu valor em toda tupla da tabela **Funcionário** deve corresponder ao valor de **cdDeppto** na tabela **Deppto**.

* CdDeppto	NmDeppto	Ramal
D1	ADMINISTRACAO	221
D2	OFICINA	235
D3	SERVICOS GERAIS	243
D4	VENDAS	258

NrMatric	NmFunc	DtAdm	Sexo	CdCargo	CdDeppto*
1001	JOAO SAMPAIO	10/08/93	M	C2	D2
1004	LUCIO TORRES	02/03/94	M	C2	D2
1034	ROBERTO PEREIRA	23/05/92	M	C3	D1

Para especificar essas restrições devemos, primeiro, ter um entendimento claro do significado, ou do papel, que cada conjunto de atributos desempenha nos vários esquemas de relação do banco de dados. As restrições de integridade referencial surgem dos relacionamentos entre as entidades representadas em esquemas de relação.

Por exemplo, considere as tabelas mostradas acima. Na tabela FUNCIONÁRIO, o atributo cdDeppto refere-se ao departamento para o qual um funcionário trabalha, portanto, indicamos cdDeppto como chave estrangeira de FUNCIONÁRIO, fazendo referência à relação funcionário. Isso significa que todo valor de cdDeppto em qualquer tupla da relação FUNCIONÁRIO deve corresponder a um valor da chave primária de DEPTO, ou o valor de cdDeppto pode ser **null** (nulo) se o empregado não pertencer a um departamento.

Observe que um chave estrangeira pode se referir à sua própria relação. Como exemplo, na tabela EMPREGADOS da figura abaixo, o campo gerente_id é uma chave estrangeira que aponta para a própria tabela FUNCIONÁRIOS, fazendo referência à chave primária func_id. Isso porque o gerente também é um funcionário.

Funcionários

func_id	func_nome	gerente_id
1	Maria	NULL
2	Ana	NULL
3	Carla	NULL
4	Antonio	1
5	Nestor	1
6	Eduardo	1
7	Anderson	1

Podemos exibir diagramaticamente as restrições de integridade referencial puxando um arco direto de cada chave estrangeira à relação que ela faz referência. Para maior clareza, a seta deve apontar para a chave primária da relação referida.

Todas as restrições de integridade deveriam ser especificadas no esquema do banco de dados relacional, caso queira impor essas restrições aos estados do banco de dados. Portanto, a DDL possui recursos para especificar os vários tipos de restrições e, assim, o SGBD pode, automaticamente, garanti-las. A maioria dos SGBDS relacionais suporta restrições de chave e integridade de entidade, e providenciam suporte para a integridade referencial. Essas restrições são especificadas como da definição de dados.