

BANCO DE DADOS I

Prof. Marcos Alexandre Marques

Introdução e Conceitos Gerais

• Conceitos

Um **banco de dados** pode ser definido como um conjunto de **dados** devidamente relacionados.

Por **dados** podemos compreender como **fatos conhecidos** que podem ser armazenados e que possuem um significado implícito. 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 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;
- 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 pode ser criado e mantido por um conjunto de aplicações desenvolvidas especialmente para esta tarefa ou por um **Sistema Gerenciador de Banco de Dados (SGBD)**. Um SGBD permite aos usuários criarem e manipularem bancos de dados de propósito geral. O conjunto formado por um banco de dados mais as aplicações que manipulam o mesmo é chamado de **Sistema de Banco de Dados**.

Um **SGBD** é uma coleção de programas que permitem ao usuário definir, construir e manipular Bases de Dados para as mais diversas finalidades

• Banco de Dados X Processamento Tradicional de Arquivos

Auto Informação

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. 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.

Vantagens dos SGBD

- **Consistência e Eliminação da Redundância de Dados** - Evitam que os dados estejam duplicados em diversos arquivos diferentes, e cada arquivo correndo o risco de estar com as estruturas diferentes entre si, em decorrência do uso de linguagens diferentes entre os sistemas que manipulam estes arquivos.
- **Integridade** - Mantém os valores dos dados satisfazendo as exigências das regras de negócio que estão armazenadas no banco de dados. Por exemplo, considere que o balanço de uma conta bancária não possa ser menor que R\$ 25,00. Usando bancos de dados podemos colocar estas restrições direto no banco de dados, não sendo necessário alterar os sistemas sempre que uma nova restrição surgir.
- **Controle de concorrência** - O banco de dados garante o controle ao acesso aos dados, quando vários usuários estão tentando acessar os mesmos dados ao mesmo tempo. Por exemplo, dois usuários estão pagando saques de uma mesma conta bancária, ao mesmo tempo. Esta conta está com saldo de R\$ 400,00. O usuário **A** paga o valor de R\$ 50,00, e o usuário **B** paga o valor de R\$ 100,00. Se o controle não for correto, o saldo poderá ficar igual a R\$ 350,00 ou R\$ 300,00, quando deveria ser de R\$ 250,00.
- **Controle de Segurança** - Garantem o acesso ao banco de dados, com mecanismos de senhas, hierarquias e permissões de acesso e execução, por usuário e grupos de usuários.

- **Recuperação e Backup** - Garantem que os dados sejam recuperados de forma adequada, se algo ocorre ao sistema de computadores ou de arquivos, a partir de backups (cópias de segurança) feitas pelo mesmo banco de dados, enquanto estão sendo utilizados.

• **Usuários**

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

Administrador de Dados (DA)

Desenvolve e administra centralizadamente estratégias, procedimentos, práticas e planos capazes de disponibilizar os dados corporativos necessários, quando necessários, com integridade, privacidade, documentação e compartilhamento. Participa dos levantamentos de dados, e regras de negócio da empresa. Elabora e/ou acompanha a confecção de modelos. Participa da compatibilização do planejamento de sistemas com os modelos lógicos. Participa de pesquisa de softwares de apoio, relacionados a área de AD, assim como SGBD.

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, bem como da criação das estruturas, restrições e integridades, definidas no projeto.

Usuários Finais

Existem basicamente três categorias de usuários finais que são os usuários finais do banco de dados, fazendo consultas, atualizações e gerando documentos:

- 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 (programas);
- usuários sofisticados: são usuários que estão familiarizados com o SGBD e realizam consultas complexas.

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.

• Linguagens de Manipulação de Dados

Para que possamos criar a estrutura de um banco de dados, e manipular seu conteúdo, é necessário que exista uma ou mais linguagens que trabalhem com estas situações. Existem duas definições para estas linguagens:

- **DDL (Data Definition Language - Linguagem de Definição de Dados)** - É a linguagem que permite a definição e manipulação de toda a estrutura de um banco de dados (campos, tipos, arquivos, etc). Estas definições de dados devem ser armazenadas em algum lugar no banco de dados. Desta forma, estas definições são mantidas no **DD (Data Dictionary - Dicionário de Dados)**. O DD é um arquivo que contém metadados; isto é, dados acerca de dados. Este arquivo é consultado antes de dados reais serem lidos ou modificados no sistema de banco de dados

DML (Data Manipulation Language - Linguagem de Manipulação de Dados) - É a linguagem que permite aos usuários do banco de dados manipularem os dados. Com esta linguagem é possível inserir, alterar e excluir os dados nas estruturas criadas. Existem dois tipos básicos de DML:

Procedimental ou Procedural: *requer do usuário a especificação de **quais** dados são desejados e **como** chegar até eles (EX: Álgebra Relacional).*

Não-Procedimental ou Não-Procedural: *requer do usuário a especificação de **quais** dados são desejados, **sem** especificar como chegar até eles (EX: Cálculo Relacional)*

• Estrutura Geral de um SGDB

Componentes funcionais:

- **Gerenciador de Arquivos:** gerencia a alocação de espaço e armazenamento em disco e estruturas de dados.
- **Gerenciador do Banco de Dados:** proporciona interface entre os dados de baixo nível e os programas de aplicação e consultas.
- **Processador de Consultas:** traduz comandos de uma linguagem de consulta em instruções de baixo nível para que o GBD entenda. Tenta otimizar os pedidos de consulta dos usuários.
- **Pré-Compilador DML:** compila comandos DML em rotinas da linguagem do host. Precisa interagir com o processador de consultas para gerar código apropriado.
- **Compilador DDL:** converte comandos DDL em um conjunto de tabelas contendo metadados, que são armazenados no DD.
- **Gerenciador de Buffer:** é responsável pela transferência de informações entre o disco e a memória principal
- **Seletor de Estratégias:** tenta transformar uma requisição do usuário em um forma equivalente, mas mais eficiente, encontrando assim uma boa estratégia para executar a consulta
- **Gerenciador de Autorização e Integridade:** verifica o cumprimento de restrições de integridade e verifica a autorização de usuários para acessar os dados
- **Gerenciador de Recuperação:** que assegura a permanência do banco de dados em um estado consistente a despeito de falhas de sistema
- **Controlador de Concorrência:** assegura que interações concorrentes no banco de dados procedam sem conflitos umas com as outras.

Estruturas de Dados

- **Arquivo de Dados:** armazenam os dados propriamente ditos.
- **Dicionário de Dados:** armazena informações sobre a estrutura do banco de dados.
- **Índices:** proporcionam acesso rápido aos itens de dados com valores específicos. Definem restrições e garantem a

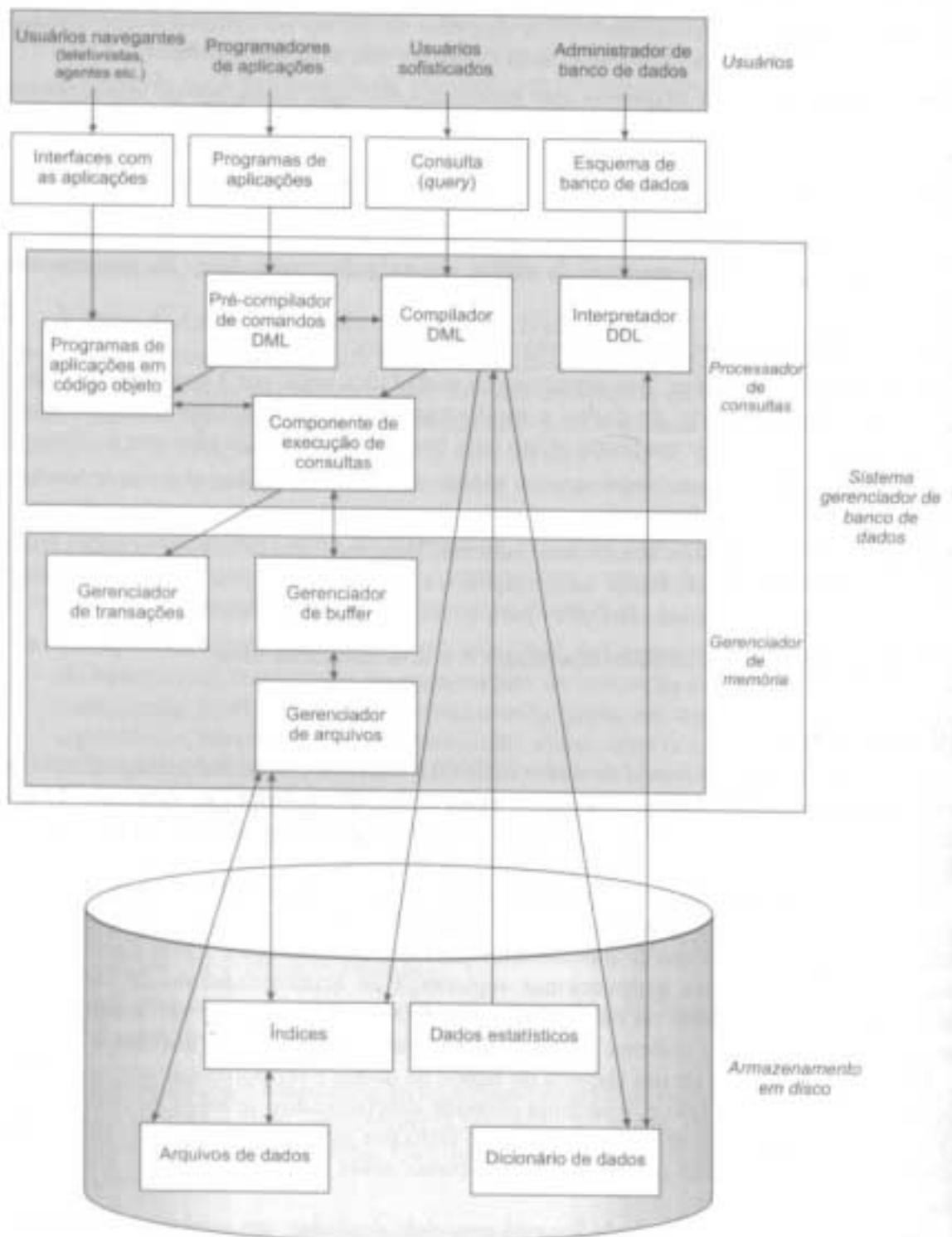


Figura 1 - Estrutura de um SGBD

integridade referencial.

- **Níveis de Abstração de Informação**

- **Conceito**

"Habilidade mental que permite aos seres humanos visualizarem os problemas do mundo real com vários graus de detalhe, dependendo do contexto do problema." (J.Rumbaum - Modelagem e Projetos Baseados em Objetos)

- **Níveis de Abstração**

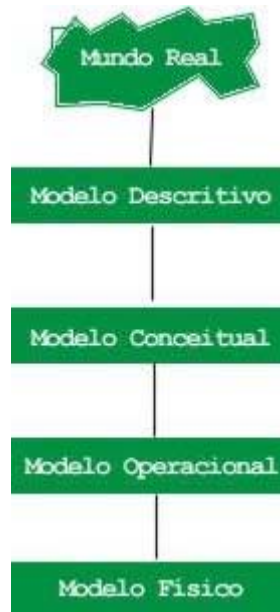


Figura 2 - Níveis de Abstração de Dados

(Waldemar Setzer)

MUNDO REAL: Visão que se tem do mundo real. Primeiro nível de abstração das informações que estarão contidas no banco de dados. Seres, objetos, organismos, fatos. Organização, alterações;

MODELO DESCRITIVO: Segundo nível de abstração, nesta etapa, o projetista deve colocar as informações obtidas no mundo real, em um modelo descritivo, com suas próprias palavras, como se estivesse escrevendo um livro sobre o que está aprendendo. Informações informais. Descrições das estruturas e das transações;

MODELO CONCEITUAL: Neste nível são usadas ferramentas gráficas, e métodos formais para demonstrar o modelo descritivo. Neste nível ainda não devemos nos preocupar com as ferramentas que serão usadas para implementar o banco de dados e sistemas (linguagens de programação, marca de SGDB, etc). Informações formais. Descrição das estruturas, especificações de manipulação;

MODELO OPERACIONAL: Com base no modelo conceitual, este modelo se preocupa em como implementar as estruturas criadas no nível anterior, de acordo com as ferramentas escolhidas para desenvolver o banco de dados e sistemas. Estruturas externas de dados, especificações e programas de manipulação;

MODELO FÍSICO: Nível mais complexo da abstração, que se preocupa em como os dados serão armazenados no disco rígido. Cadeia de bits e bytes. Estruturas internas de arquivos e tabelas, programas interpretáveis ou executáveis.

VISÕES: Nível mais alto da abstração, não aparece no gráfico de Setzer. É o nível de abstração que os usuários mais simples têm, são visões de partes do banco de dados. Ex: Cadastro de clientes, consulta no estoque, etc.

• Modelo de Dados

"Coleção de ferramentas conceituais para descrição de dados, relacionamento entre os dados, semântica e restrições de dados." (Henry F. Korth)

Antes de iniciar a construção de um banco de dados, é essencialmente necessário que se faça o seu projeto.

Devemos comparar a construção de bancos de dados (bem como de sistemas), com a construção de uma casa. Se uma casa está para ser construída, e não existe um projeto, um planejamento correto, com certeza existirão alterações no decorrer da construção, o que implica em um aumento considerável no tempo de execução e principalmente no custo. Portanto, para evitar este tipo de problema na construção de bancos de dados, deveremos estudar o problema, analisa-lo e projetá-lo, antes do início da sua implementação. Para isto, utilizaremos modelos que nos ajudam a demonstrar gráficamente o banco de dados que será construído.

É importante fazer a distinção entre a *descrição do banco de dados* e o banco de dados em si. Para isto utilizamos vários níveis de abstração para o projeto de banco de dados, com o auxílio de ferramentas próprias a cada nível.

• Arquitetura em Três Camadas

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

- **nível interno ou físico:** 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. Descreve como os dados estão armazenados;
- **nível conceitual ou lógico:** o qual descreve a estrutura do banco de dados como um todo; é uma descrição global do banco de dados, que não fornece detalhes do modo como os dados estão fisicamente armazenados. Descreve os relacionamentos entre os esquemas e dados e as restrições. Neste nível o usuário não se preocupa com quais ferramentas de banco de dados deverá trabalhar. Os usuários deste nível são os administradores de dados e analistas de sistema;
- **nível externo ou 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. Os usuários deste nível não precisam se preocupar com a complexidade do banco de dados.

• Instâncias e Esquemas

• Esquema

Concepção global do banco de dados. Definição das estruturas, a forma como serão armazenados os dados. Definição, modelo

• Instância

Coleção de informações armazenadas no banco de dados em um instante particular. Cada ocorrência de um esquema. São os valores atribuídos a um esquema. Valor, conteúdo.

O SGBD é responsável por garantir que toda instância do banco de dados satisfaça ao esquema, respeitando sua estrutura e suas restrições.

EX:



Figura 4 - Ex. Esquemas e Instâncias

• 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 (). Existem dois tipos de independência de dados:

- **independência de dados lógica:** é a capacidade de alterar o esquema conceitual sem ter que alterar a aplicação ou o modelo físico;
- **independência de dados física:** é a capacidade de alterar o esquema físico sem ter que alterar o as aplicações do usuário.

MER - MODELO ENTIDADE RELACIONAMENTO

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. Ele representa um **esquema de empreendimento**, tal esquema é a demonstração lógica global do banco de dados.

Através do MER é possível demonstrar graficamente as regras de negócio de uma empresa.

• Entidades

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"

Exemplos de entidades:

Aluno, Conta Bancaria, Cliente, Nota Fiscal, etc.

Entidade: objeto que existe e é distinguível de outros objetos.

Conjunto de Entidades: grupo de entidades do mesmo tipo.

• Atributos

Como já foi dito, as propriedades que caracterizam uma entidade são chamadas de **atributos**.

Exemplos de atributos:

Para entidade Aluno: Idade, Nome, Data Nascimento, Endereço

Para entidade Conta: Numero, Cliente, Saldo

Domínio de Atributos: conjunto de valores permitidos aos atributos. Ex: O atributo Data de Nascimento somente pode conter valores no padrão dd/mm/aaaa (dia/mês/ano). O atributo Idade somente pode conter números inteiros positivos.

Atributos Compostos: Um atributo composto é formado por vários atributos da entidade. Ex: Endereço pode ser formado por Rua + Numero + Bairro + Cidade + Estado + CEP

Atributos Monovalorados: atributos que somente contém um valor para cada instância. Ex: Idade

Atributos Multivalorados: atributos que podem conter mais de um valor para cada instância. Ex: Nome_Dependente em uma entidade Funcionário.

Atributos Derivados: atributos que podem ser obtidos a partir de cálculo sobre o valor de outros atributos. Ex: Idade pode ser obtida a partir do atributo Data_Nascimento. Exemplos:

Cliente

Nome	Endereço	Cidade
João	Rua B	Ponta Grossa
Maria	Rua C	Curitiba
Carlos	Rua H	Curitiba

Conta

Número	Cliente	Saldo
342	João	499
443	Carlos	3838

Tabela 2 - Exemplo Tabela Conta

Tabela 1 - Exemplo Tabela Cliente

Chaves

Uma entidade deve ter a capacidade de identificar cada uma de suas instâncias separadamente em um banco de dados. Para fazer isto utilizamos o conceito de chaves.

Superchave: Conjunto de **um ou mais** atributos que permite identificar unicamente uma entidade no conjunto de entidades.

Ex: Cliente (Nome, CPF, RG, Endereço). Na entidade cliente, os conjuntos de atributos [Nome, CPF] e [RG] são superchaves.

Chaves Candidatas: Conjuntos com o **menor número possível** de atributos que permite identificar unicamente uma entidade no conjunto de entidades.

Ex: Na entidade cliente do exemplo anterior, o conjunto de atributos [CPF] e [RG] são chaves candidatas.

Chaves Primárias: Chave candidata **escolhida** como identificação de entidades no conjunto de entidades.

Cada entidade pode ter apenas **uma** chave primária, mas várias candidatas. A chave primária de uma entidade não poderá ter valores nulos ou duplicados.

Ex: Na entidade cliente do exemplo anterior, o conjunto de atributos [CPF] pode ser o escolhido para ser o conjunto de chave primária.

Chaves Estrangeiras: Conjunto de atributos de uma entidade que é chave primária da entidade com a qual possui relacionamento.

Ex: Funcionário(Nome, Cod, CodDep) , se relaciona com Depto(CodDep, Nome)

• Relacionamento

Associação entre uma ou mais entidades.

• Restrições de Mapeamento

• **Cardinalidade**

Restrição que expressa o número de entidades ao qual outra entidade pode estar associada via um relacionamento.

• ***Tipos de Cardinalidade:***

Um-para-um (1:1): uma entidade em **A** está associada a no máximo uma entidade em **B**, e uma entidade em **B** está associada a no máximo uma entidade em **A**.

EX: Considere um relacionamento entre Funcionário e Dependente, onde somente é permitido um Dependente por

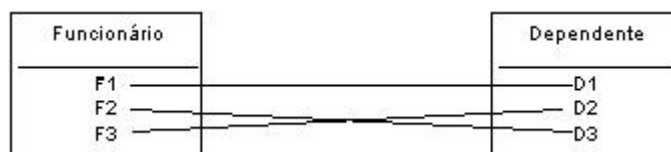
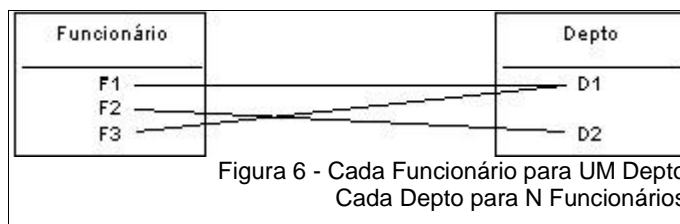


Figura 5 - Cada Funcionário para UM Dependente
Cada Dependente para UM Funcionário

Funcionário

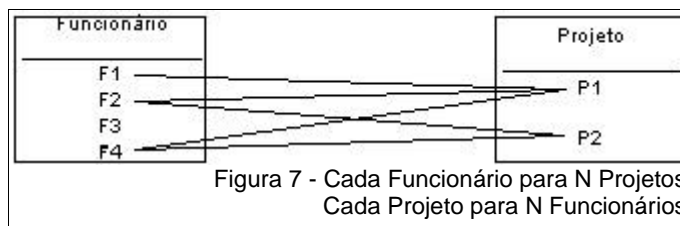
Um-para-muitos (1:N): uma entidade em **A** está associada a qualquer número de entidades em **B**, entretanto uma entidade em **B** está associada a no máximo uma entidade em **A**.

Ex: Considere um relacionamento entre Funcionário e Depto, onde cada funcionário somente pode estar lotado em um Depto, mas cada Depto pode ter vários funcionários:



Muitos-para-muitos (N:N): uma entidade em **A** está associada a qualquer número de entidades em **B**, e uma entidade em **B** está associada a qualquer número de entidades em **A**.

Ex: Considere um relacionamento entre Funcionário e Projeto, onde cada funcionário pode participar de vários



projetos e cada Projeto pode ter vários funcionários trabalhando:

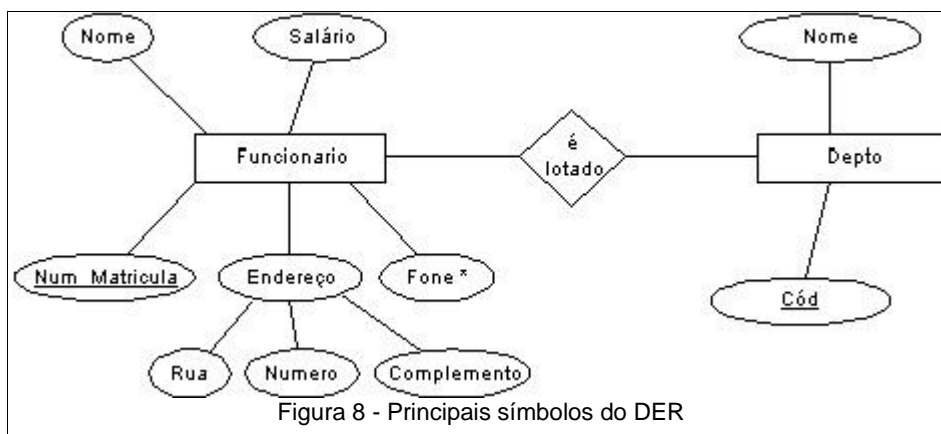
- **Dependência Existencial**

Se a existência da entidade **x** depende da existência da entidade **y**, então diz-se que **x** é **existencialmente dependente de y**. A entidade **y** é chamada **dominante** e **x** é chamada **subordinada**. Este conceito é importante para manter a integridade dos dados contidos em um Banco de Dados. Suponha a existência de uma entidade NF (Notas Fiscais) em um Banco de Dados. Esta entidade se relaciona com a entidade Cliente (cada NF esta relacionada ao Cliente que efetuou determinada compra). A entidade NF é subordinada a entidade Cliente, pois não é possível emitir uma Nota Fiscal sem a existência de um Cliente. Desta forma, quando informamos ao Banco de Dados esta Dependência Existencial, aplica mecanismos que garantem que uma NF não conterá (por exemplo) o Código de Cliente igual a 5, pois o Cliente 5 não existe em nosso Banco de Dados. Da mesma forma que garante que o Cliente com Código igual a 2 não será excluído de nosso Banco de Dados, se existir um ou mais NF relacionadas a ele.

- **Entidades Fortes e Fracas**

Entidade Forte: é uma entidade que possui chave primária, por definição é uma entidade dominante.

Entidade Fraca: é uma entidade que não possui chave primária, por definição é uma entidade subordinada. Para formarmos a chave primária de uma entidade fraca, utilizamos a chave primária da entidade forte da qual ela é existencialmente dependente, mais o conjunto mínimo de atributos que possa identificar uma entidade em um



conjunto de entidades fracas.

Ex: O relacionamento entre as entidades Funcionário e Histórico de Pagamento podem ser modeladas respectivamente como entidades Forte e Fraca. Isto porque a entidade Funcionário (Nome, Num.Matricula, CPF, RG, Endereco) tem, por natureza, atributos que são chaves candidatas, e por consequência, possui uma chave primária. Já a entidade Histórico de Pagamento (Data, Vlr Pago) não possui um conjunto de atributos que possa identificar cada uma de suas instâncias. Desta forma, para montarmos a chave primária de Histórico de Pagamento, usamos a chave primária da entidade Funcionário. Definindo assim a entidade Histórico de Pagamento (Num Matricula, Data, Vlr Pago).

• Diagrama Entidade-Relacionamento (DER)

O DER 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 DER fornece uma visão lógica do banco de dados, fornecendo um conceito mais generalizado de como estão estruturados os dados de um sistema.

Ao desenhar o DER, estaremos na fase Conceitual da Modelagem de um Banco de Dados. Esta fase nos dá, através do DER, uma "leitura visual" da estrutura do Banco de Dados. Isto é extremamente importante, pois facilita o entendimento da estrutura e regras de negócio¹.

Os principais símbolos de objetos do MER são:

Entidade: representada por retângulos.

Atributos: representados por círculos ou elipses, ligados as entidades por linhas.

Relacionamentos: representados por losangos ligados às entidades por linhas.

Atributos Determinantes (Chave Primária): sublinhado ou bola que acompanha preenchida

Atributos Compostos: Linhas a partir do nome principal

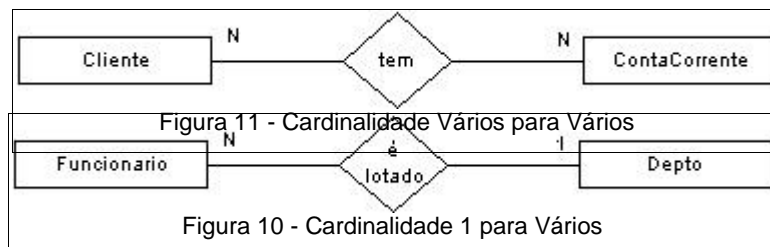
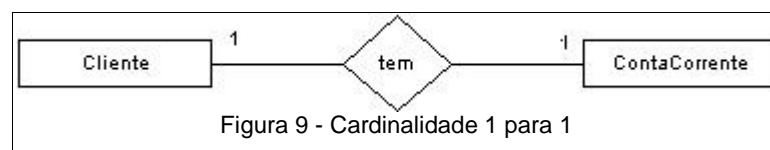
Atributos Multivalorados: * ao lado do nome

Cardinalidade: Indica-se a cardinalidade sempre ao lado oposto da entidade que a possui. Usamos os literais:

"1": para permissão de relacionamento com no máximo uma instância da entidade que se relaciona;

"N": para permissão de vários relacionamentos com instâncias da entidade que se relaciona, sem limites;

"3": ou qualquer outro número, para permissão de relacionamento com no máximo "3" (ou outro número que seja usado) instâncias da entidade que se relaciona;



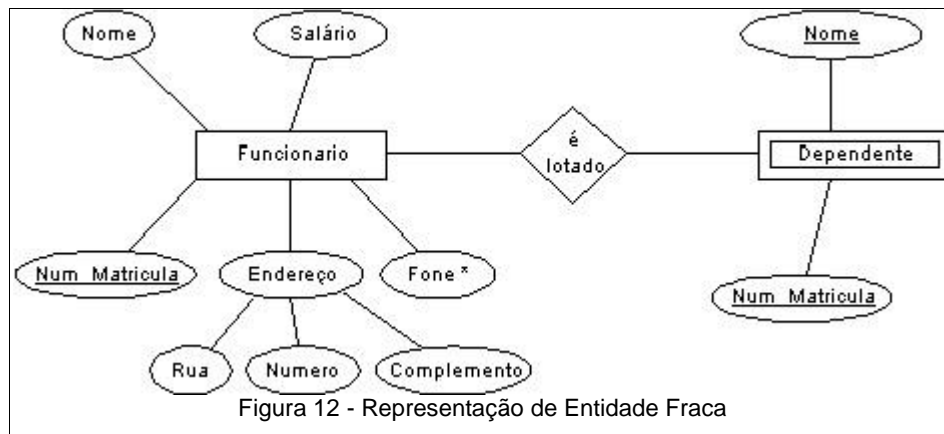
¹Regras de Negócio- funcionamento da empresa, regras de como a empresa trabalha. Ex: a afirmativa de que "Cada projeto deve ser gerenciado por apenas um funcionário" é uma regra de negócio de uma empresa, e deve ser representado na modelagem do Banco de Dados.

Para se definir a cardinalidade, fazemos as seguintes perguntas:

P: “Um Funcionário pode estar lotado em, no máximo, quantos Departamentos?” - **R:** 1 (um)

P: “Um Departamento pode ter lotado, no máximo, quantos Funcionários?” - **R:** N (vários)

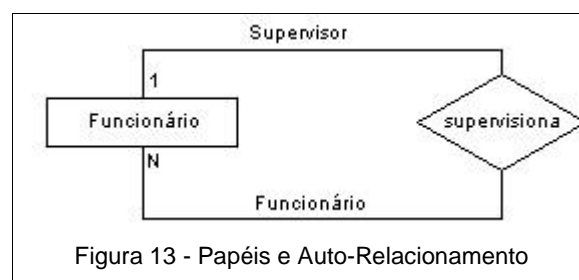
Entidade Fraca: representada por um retângulo com borda dupla



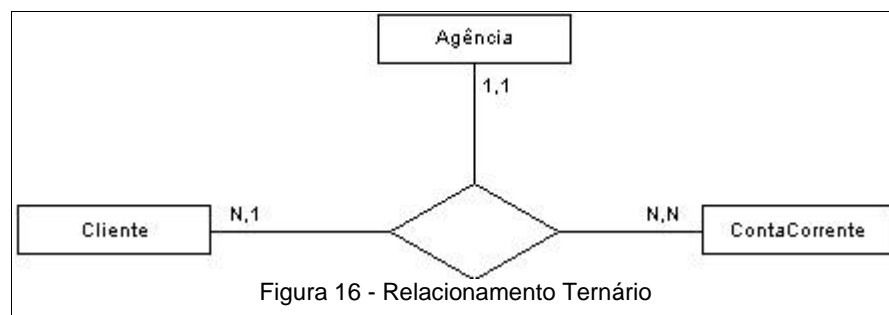
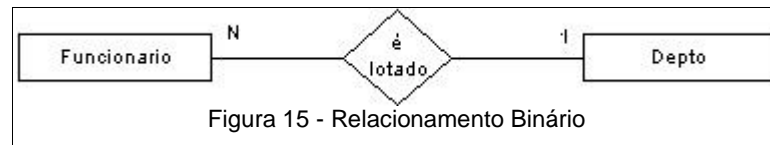
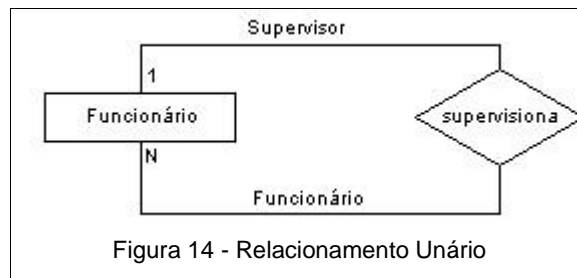
Note que a entidade Dependente não possui chave primária. Desta forma, pegamos a chave primária de Funcionário mais o conjunto mínimo de atributos de Dependente, para formar a chave primária.

Papéis: Cada tipo entidade que participa de um relacionamento desempenha um **papel** (tem uma função) particular. O nome do papel representa esta função.

Relacionamento Recursivo ou Auto-Relacionamento: O auto-relacionamento ocorre quando uma instância de uma entidade se relaciona com uma ou mais instâncias da mesma entidade. Considere o caso em que cada Funcionário em uma empresa tem um Supervisor. Sendo o Supervisor também um Funcionário. Para demonstrarmos que um Supervisor supervisiona vários Funcionários, temos:



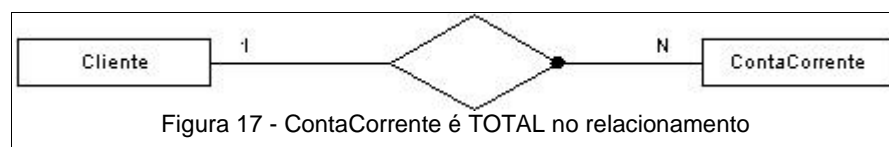
Grau do Relacionamento: O grau do relacionamento é determinado pela quantidade de entidades que fazem parte deste relacionamento.



Acima estão exemplificados relacionamentos de graus unário, binário e ternário. Podemos ter relacionamentos com graus maiores, mas estes dificilmente ocorrem. Note que a cardinalidade no relacionamento ternário torna-se mais complexa de se expressar. Para podermos trabalhar nesta apostila, seguimos o seguinte padrão: Iniciamos com a entidade mais à esquerda, e seguimos no sentido horário. Desta forma:

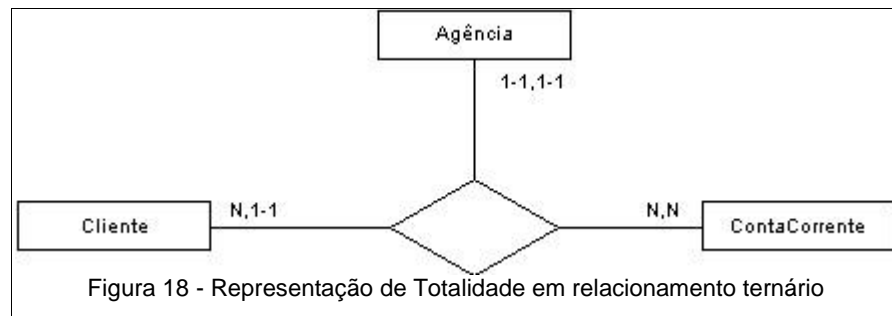
P - Um Cliente pode se relacionar com quantas Agências?	R	-	1
P - Um Cliente pode se relacionar com quantas Contas?	R	-	N
P - Uma Agência pode se relacionar com quantos Clientes?	R	-	N
P - Uma Agência pode se relacionar com quantas Contas?	R	-	N
P - Uma Conta pode se relacionar com quantos Clientes?	R	-	1
P - Uma Conta pode se relacionar com quantas Agências?	R - 1		

Dependência Existencial ou Relacionamento Total: Dizemos que uma entidade é total em um relacionamento, quando todas as suas instâncias, obrigatoriamente, se relacionam com uma instância da outra entidade. Ex: Uma Conta Corrente em um banco somente pode existir se estiver relacionada a pelo menos um Cliente.



Para podermos expressar que uma entidade é total no relacionamento, colocamos uma bola com o centro preenchido, próximo ao relacionamento, do lado de quem faz a pergunta:

P - Para cada Cliente, quantas ContaCorrente no <u>mínimo</u> deve haver relacionamento?	R	-	0
P - Para cada ContaCorrente, quantos Clientes no <u>mínimo</u> deve haver relacionamento?	R - 1		



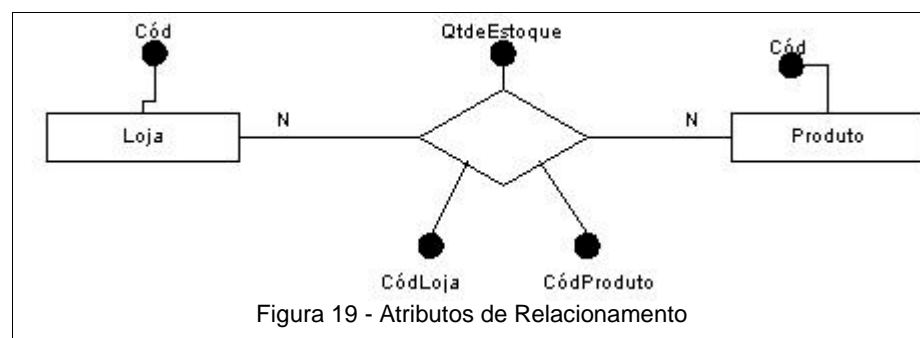
Note que a representação com bolinhas, em relacionamentos com grau maior que 2, não é aconselhável, pois não teríamos como saber a qual par de entidades pertence a Totalidade. Desta forma, colocamos o numero 1 separando por um traço ("-") a cardinalidade deste relacionamento. Assim:

Cada Cliente deve no mínimo se relacionar com 1 Agência.

Cada ContaCorrente deve se relacionar no mínimo com 1 Cliente.

Cada ContaCorrente deve se relacionar no mínimo com 1 Agência.

Atributos de Relacionamento: Em relacionamentos com cardinalidade N:N, devemos então "exportar" as chaves primárias das entidades que participam do relacionamento, e coloca-las como atributos (que formarão a chave primária) no relacionamento. Outros atributos também podem fazer parte do relacionamento, como no exemplo abaixo, onde cada loja tem vários produtos, e cada produto está em várias lojas. Como saber então a qtde em estoque de cada loja do produto Maçã, por exemplo? Simples, colocamos o atributo QtdeEstoque no relacionamento.

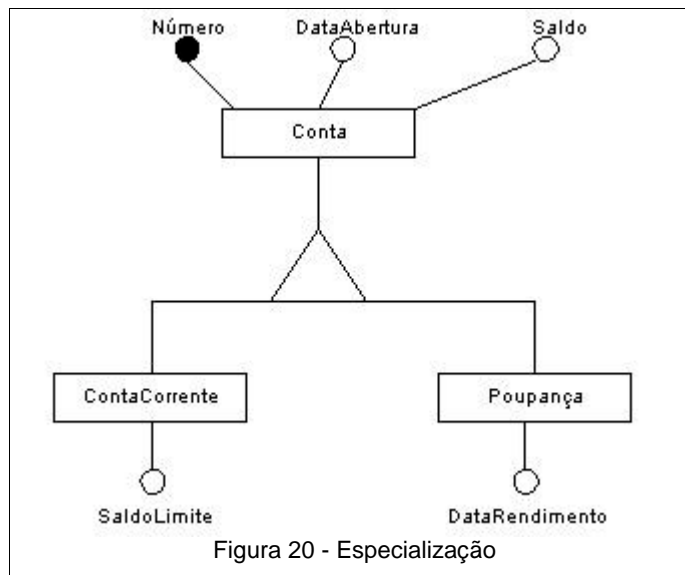


Especialização e Generalização: Através do conceito de especialização, podemos aproveitar entidades já criadas, para definir novas entidades em menos tempo de trabalho. Isto é verdade para entidades de tipo semelhante. Por exemplo, um banco possui dois tipos de contas: corrente e poupança. As contas possuem número, data de abertura e saldo. Contas corrente possuem saldo limite e as contas poupança possuem data de rendimento.

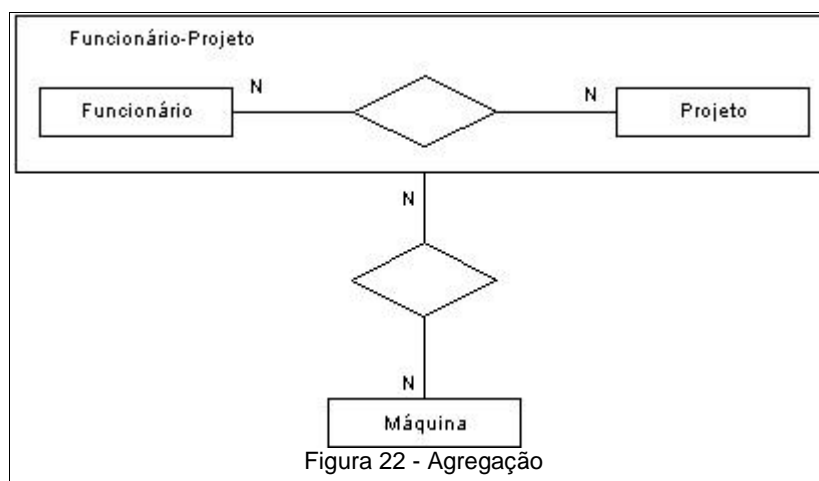
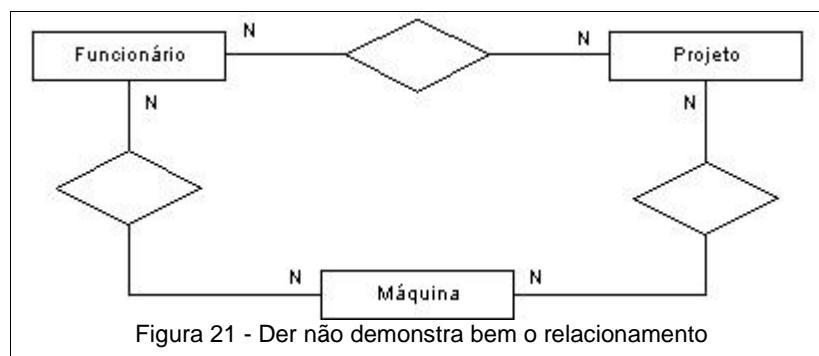
Para evitarmos a criação de duas entidades separadas, com praticamente todos os atributos iguais, e de espécie semelhante, generalizamos as duas entidades, criando uma terceira chamada superclasse, com os atributos comuns entre as duas entidades.

A superclasse também é chamada de classe mãe, e as subclasses são chamadas classes filho. As subclasses herdam todos os atributos, relacionamentos, cardinalidade e totalidade da superclasse. Se alguma destas características for diferente para alguma das subclasses, deve ser explicitamente demonstrada no diagrama. A especialização é demonstrada através de um triângulo.

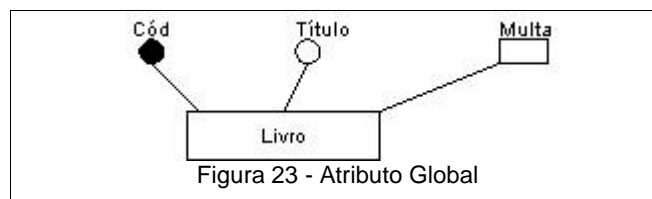
Também chamamos de Relacionamento ISA (is a), ou "É UM" em português. Desta forma, a ContaCorrente é uma Conta, e a Poupança é uma Conta.



Agregação: Usamos agregação para simplificar o diagrama. Lembre-se que não são permitidos relacionamentos entre relacionamentos.



Atributos Globais: Estão associados a uma Entidade ou a um Relacionamento. Seu valor é **CONSTANTE** para todo o conjunto ao qual pertence. Exemplo: numa biblioteca considere que a **MULTA** por dia de atraso de um exemplar seja constante, desta forma:



• O Modelo Relacional

O **modelo relacional** foi criado por Codd em 1970 e tem por finalidade representar os dados como uma coleção de relações (entidades), onde cada relação é representada por uma **tabela**.

Quando uma relação é pensada como uma tabela de valores, cada linha nesta tabela representa uma coleção de dados relacionados. Estes valores podem ser interpretados como fatos descrevendo uma instância de uma entidade ou de um relacionamento. O nome da tabela e das colunas desta tabela são utilizados para facilitar a interpretação dos valores armazenados em cada linha da tabela. Todos os valores em uma coluna são necessariamente do mesmo tipo.

Na terminologia do modelo relacional, cada tabela é chamada de **relação**; uma linha de uma tabela é chamada de **tupla (registro)**; o nome de cada coluna é chamado de **campo (atributo)**; o tipo de dado que descreve cada coluna é chamado de **domínio**.

• Domínios, Tuplas, Atributos e Relações

Um **domínio D** é um conjunto de valores atômicos, sendo que por atômico, podemos compreender que cada valor do domínio é indivisível. Durante a especificação do domínio é importante destacar o tipo, o tamanho e a faixa do atributo que está sendo especificado. Por exemplo:

Coluna	Tipo	Tamanho	Faixa
RG	Numérico	10	03000000-25999999
Nome	Caracter	30	a-z, A-Z
Salário	Numérico	5,2	00100,00-12999,99

Tabela 3 - Exemplo de Tabela Funcionário

• Esquema de comparação entre o DER de Nível Conceitual e Físico

Nível Conceitual	Nível Físico
ENTIDADE	TABELA (Relação)
ATRIBUTO	CAMPO / COLUNA
INSTÂNCIA	LINHA / TUPLA / REGISTRO
RELACIONAMENTO	Será feito através da criação da chave estrangeira na tabela Filha
Entidades que não digam respeito a Regras de Negócio, ou que venham a ter somente uma instância não são demonstradas	Entidades que não digam respeito a Regras de Negócio (ex. Parâmetros, Valores Globais, etc), ou que venham a ter somente uma instância (ex. Empresa) são demonstradas. Isto ocorre porque temos que guardar estes valores no banco de dados. Por exemplo, como saberemos qual o endereço da empresa, na hora de imprimir uma carta a um cliente, se não o temos guardado em lugar algum?

Tabela 4 - Características dos Níveis Conceitual e Físico

• Mapeamento para o Modelo Físico (Modelo Entidade Relacionamento para o Modelo Relacional)

O mapeamento do modelo entidade relacionamento para o Modelo Relacional segue oito passos básicos a saber:

1. Para cada entidade **E** no modelo ER é criada uma tabela **T₁** no Modelo Relacional que inclua todos os atributos simples de **E**; para cada atributo composto, são inseridos apenas os componentes simples de cada um; um dos atributos chaves de **E** deve ser escolhida como a chave primária de **T₁**;
2. Para cada entidade fraca **EF** com entidade proprietária **E** no modelo ER, é criada uma tabela **T₁** no Modelo Relacional incluindo todos os atributos simples de **EF**; para cada atributo composto, são inseridos apenas os componentes simples de cada um; a chave primária desta relação **T₁** será composta pela chave parcial da entidade fraca **EF** mais a chave primária da entidade proprietária **E**;
3. Para cada relacionamento regular com cardinalidade 1:1 entre entidades **E₁** e **E₂** que geraram as tabelas **T₁** e **T₂** respectivamente, devemos escolher a chave primária de uma das relações (**T₁**, **T₂**) e inseri-la como chave estrangeira na outra relação; se um dos lados do relacionamento tiver participação total e outro parcial, então é interessante que a chave do lado com participação **parcial** seja inserido como chave estrangeira no lado que tem participação **total**;
4. Para cada relacionamento regular com cardinalidade 1:N entre entidades **E₁** e **E₂** respectivamente e que geraram as tabelas **T₁** e **T₂** respectivamente, deve-se inserir a chave primária de **T₁** como chave estrangeira em **T₂**;
5. Para cada relacionamento regular com cardinalidade N:N entre entidades **E₁** e **E₂**, cria-se uma nova tabela **T₁**, contendo todos os atributos do relacionamento mais o atributo chave de **E₁** e o atributo chave de **E₂**; a chave primária de **T₁** será composta pelos atributos chave de **E₁** e **E₂**;
6. Para cada atributo multivalorado **A₁**, cria-se uma tabela **T₁**, contendo o atributo multivalorado **A₁**, mais o atributo chave **C** da tabela que representa a entidade ou relacionamento que contém **A₁**; a chave primária de **T₁** será composta por **A₁** mais **C**; se **A₁** for composto, então a tabela **T₁** deverá conter todos os atributos de **A₁**;
7. Para cada relacionamento n-ário, $n > 2$, cria-se uma tabela **T₁**, contendo todos os atributos do relacionamento; a chave primária de **T₁** será composta pelos atributos chaves das entidades participantes do relacionamento;
8. Converta cada especialização com m subclasses $\{S_1, S_2, \dots, S_m\}$ e superclasse **SC**, onde os atributos de **SC** são $\{c, a_1, a_2, \dots, a_n\}$ onde **c** é a chave primária de **SC**, em tabelas utilizando uma das seguintes opções:
 1. Crie uma tabela **T** para **SC** (Super Chave) com os atributos $A(T) = \{c, a_1, a_2, \dots, a_n\}$ e chave $C(T) = c$; crie uma tabela **T_i** para cada subclasse **S_i**, $1 \leq i \leq m$, com os atributos $A(T_i) = \{c\} \cup A(S_i)$, onde $C(T) = c$;
 2. Crie uma tabela **T_i** para cada subclasse **S_i**, $1 \leq i \leq m$, com os atributos $A(T_i) = A(S_i) \cup \{c, a_1, a_2, \dots, a_n\}$ e $C(T_i) = c$;
 3. Crie uma tabela **T** com os atributos $A(T) = \{c, a_1, a_2, \dots, a_n\} \cup A(S_1) \cup \dots \cup A(S_m) \cup \{t\}$ e $C(T) = c$, onde **t** é um atributo **tipo** que indica a subclasse à qual cada tupla pertence, caso isto venha a ocorrer;
 4. Crie uma tabela **T** com atributos $A(T) = \{c, a_1, a_2, \dots, a_n\} \cup A(S_1) \cup \dots \cup A(S_m) \cup \{t_1, t_2, \dots, t_m\}$ e $C(T) = c$; esta opção é para generalizações com "overlapping", e cada **t_i**, $1 \leq i \leq m$, é um atributo "booleano" indicando se a tupla pertence ou não à subclasse **S_i**; embora funcional, esta opção pode gerar uma quantidade muito grande de valores nulos;

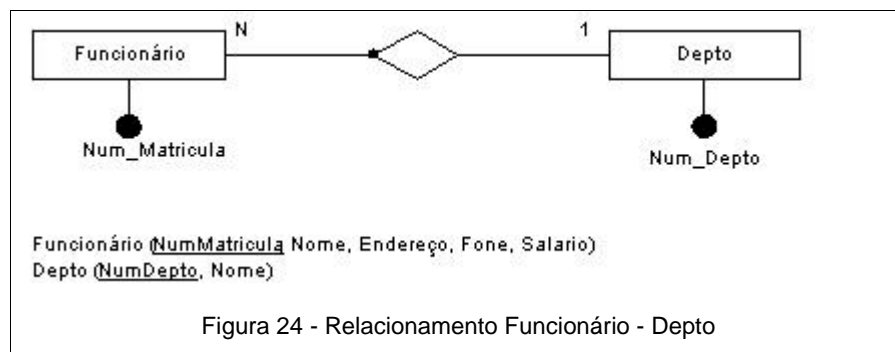


TABELA FUNCIONÁRIO

Nome Campo	Tipo	Tam	Não Nulo	Máscara	OBS
NumMatricula	Inteiro Longo		X	999.999.999	
Nome	String	30	X		
Endereco	String	50	X		
Fone	String	13		(99)#999-9999	
Salario	Monetario			999.999.999,99	
NumDepto	Inteiro		X	999	Ch. Estrangeira

INDICES E CHAVES

Nome Índice	Campos	Primario	Estrangeiro	Tab.Refere	Campo Tab.Ref.	naOrdem
funPK	NumMatricula	X				Asc
funFKdepto	NumDepto		X	Depto	NumDepto	Asc
FunNome	Nome					Asc

TABELA DEPTO

Nome Campo	Tipo	Tam	Não Nulo	Máscara	OBS
NumDepto	Inteiro		X	999	
Nome	String	30	X		

INDICES E CHAVES

Nome Índice	Campos	Primario	Estrangeiro	Tab.Refere	Campo Tab.Ref.	naOrdem
deptoPK	NumDepto	X				Asc

• Dependências Funcionais

Síntese 1: Generalização de dependências de chave. Elas requerem que um valor em um certo conjunto de atributos determine univocamente o valor em outro conjunto de atributos.

Síntese 2: Uma **dependência funcional** é uma restrição entre dois conjuntos de atributos de uma base de dados. Suponha que o esquema de uma base de dados **R** possua n atributos A_1, A_2, \dots, A_n ; pense em $R = \{ A_1, A_2, \dots, A_n \}$ como a representação universal da base de dados. Uma dependência funcional, representada por $X \rightarrow Y$ entre dois conjuntos de atributos **X** e **Y** que são subconjuntos de **R** especificam uma restrição nas tuplas que podem compor uma instância relação r de **R**. A restrição estabelece que para qualquer par de tuplas t_1 e t_2 em r de forma que $t_1[X] = t_2[X]$, é obrigado a existir $t_1[Y] = t_2[Y]$. Isto significa que os valores do componente **Y** em uma tupla em r **depende de**, ou **é determinada pelos** valores do componente **X**. Para $X \rightarrow Y$ lê-se: **Y é funcionalmente dependente de X**, ou **X infere sobre Y**.

Síntese 3: Se repetirmos um valor para um atributo **A**, (ou um conjunto de atributos) e outro(s) atributo(s) **B** também se repetirem (redundantemente), dizemos que existe uma dependência de **B** em relação a **A**, ou que **A** determina **B**, escreve-se: $A \rightarrow B$. Se uma coluna ou conjunto de colunas não é D.F. de uma coluna ou conjunto de colunas **A**, escreve-se: $A \nrightarrow B$. (**A** não determina **B**)

Síntese 4: Dada uma relação **R**, dizemos que uma coluna ou conjunto de colunas **B** de **R** é dependente funcional de uma coluna ou um conjunto de colunas **A** de **R**, denotado $A \rightarrow B$, se a cada valor V_a de **A** existir nas linhas de **R** em que aparece V_a um único valor V_b . Em outras palavras, se V_a ocorrer em duas linhas diferentes, o mesmo V_b deve ocorrer

em ambas

Exemplo de dependências funcionais:

Funcionário (RG, Nome, CIC, Depto, RG_Supervisor, Salário)

$RG \rightarrow \{ Nome, CIC, Depto., RG_Supervisor, Salário \}$

$CIC \rightarrow \{ Nome, RG, Depto., RG_Supervisor, Salário \}$

Projeto (Número_Projeto, Nome_Projeto, Localização)

$Número_Projeto \rightarrow \{ Nome_Projeto, Localização \}$

Funcionário_Projeto (RG_Empregado, Número_Projeto, Horas)

$\{ RG_Empregado, Número_Projeto \} \rightarrow Horas$

Item-Venda (NrVenda, CodProd, Qtde, DescrProd, QtdeEst)

$\{ NrVenda, CodProd \} \rightarrow Qtde$

$\{ NrVenda, CodProd \} \rightarrow QtdeEst$

$\{ CodProd \} \rightarrow DescrProd$

- **Dependente funcional parcial:**

Sejam X, Y e Z conjuntos de colunas de uma relação. Se $(X,Y) \rightarrow Z$ e $X \not\rightarrow Z$ dizemos que Z é dependente funcional parcial de (x,y). Se $(X,Y) \rightarrow Z$ e $X \not\rightarrow Z$ e $Y \not\rightarrow Z$ então dizemos que Z é dependente funcional completo de (X,Y)

ex de DFP:

Item-Venda (NrVenda,CodProd,Qtde,descrProd,qtdeEst)

$NrVenda+CodProd \rightarrow Qtde, descrProd, qtdeEst$

$CodProd \rightarrow QtdeEst$

$CodProd \rightarrow DescrProd$

- **Dependência Funcional Transitiva:**

Dada uma relação R(A,B,C) onde A,B,C são colunas ou conjuntos de colunas, dizemos que existe uma dependência funcional transitiva se $A \rightarrow B, A \rightarrow C, B \rightarrow C$, e $B \not\rightarrow A$.

Ex:

A	B	C	D	E
---	---	---	---	---

Aluguel (RegCasa, Endereco, ValorAluguel, R.G., NomeInq)

$RegCasa \rightarrow Endereco, valorAluguel, R.G., NomeInq$

$RegCasa \rightarrow R.G, RegCasa \rightarrow NomeInq, R.G. \rightarrow NomeInq$ porém $RG \not\rightarrow RegCasa$

$A \rightarrow D, A \rightarrow E, D \rightarrow E, D \not\rightarrow A$

- **Normalização**

O processo de **normalização** pode ser visto como o processo no qual são eliminados esquemas de relações (tabelas) não satisfatórios, decompondo-os, através da separação de seus atributos em esquemas de relações menos complexas mas que satisfaçam as propriedades desejadas.

O processo de normalização como foi proposto inicialmente por Codd conduz um esquema de relação através de um bateria de testes para certificar se o mesmo está na **1ª, 2ª e 3ª Formas Normais**. Estas três Formas Normais são baseadas em dependências funcionais dos atributos do esquema de relação.

- **1ª. Forma Normal:**

- Uma relação não deve conter outras relações. Todos seus atributos devem ser monovalorados, isto é, não deve conter atributos multivalorados.
- A relação entre a chave primária de uma tabela e cada uma de suas colunas tem que ser um-para-um, nesta direção.

Procedimentos para colocar uma relação na 1a FN:

1. Identificar a chave primária da relação;
2. Identificar o grupo repetitivo e remove-lo da entidade;
3. Criar uma nova entidade com a chave primária da entidade anterior e o grupo repetitivo.
4. A chave primária da nova entidade será obtida pela concatenação da chave primária da entidade inicial e a do grupo

repetitivo.

Ex: Em um sistema de estoque é necessário obter-se o preço médio de cada item em estoque (produto), como o sistema permite devolução de produtos em garantia (90 dias) é necessário registrar cada quantidade entrada com respectiva data e valor para posterior reprocessamento. O desenvolvedor chegou a algumas “soluções”:

Exemplo de valores para Produto

Código	Descrição	Quantidades	Datas	Preço
001	Monitor	110	01/03/95	100,00
		500	10/04/95	97,00
		300	15/05/95	103,00
		210	12/06/95	99,00

Tabela Produto

Código	numérico
Descrição	Alfa [30]
Quantidades	Array [1..10]
Data	Array [1..10]
Preços	Array [1..10]

Quais os Problemas??

- É possível incluir até quantas entradas? R: 10 no máximo
- Há simples entrada de dados!? R: Não

A solução simples (e até intuitiva):

Tabela Produto

Código	Descrição
001	Monitor

Tabela Entradas

Código	Quantidade	Data	Preço
001	110	01/03/95	100,00
001	500	10/04/95	97,00
001	300	15/05/95	103,00
001	210	12/06/95	99

- É possível incluir até quantas entradas? R: Quantas forem necessárias
- Há simples entrada de dados? R: Sim

Ex: Considere a relação Cliente (Cod, Nome, Endereco, Fone), onde o cliente pode ter mais de um telefone. Qual seria a solução para colocar a relação na 1FN?

Cliente (Cod, Nome, Endereco)

ClienteFone (Cod, Fone)

• **2ª. Forma Normal:**

Uma relação está na 2. FN se estiver na 1. FN e qualquer atributo da(s) chave(s) (candidatas) for DF completo em relação a cada CHAVE. Em outras palavras não há atributos fora da(s) chaves que seja DF parcial em relação a cada chave.

- Uma tabela **T** está na 2ª Forma Normal se estiver na 1ª Forma Normal e todo atributo que não compõem a chave primária **C** for totalmente funcionalmente dependente da chave primária **C**. Se uma tabela não está na 2ª Forma Normal a mesma pode ser normalizada gerando outras tabelas cujos atributos que não façam parte da chave primária sejam totalmente funcionalmente dependente da mesma, ficando a tabela na 2ª Forma Normal.

Procedimentos para colocar uma relação na 2a FN:

1. Identificar os atributos que não são funcionalmente dependentes de toda a chave primária.
2. Remover da relação todos esses atributos identificados e criar uma nova relação com eles.

3.A chave primária da nova relação será o atributo do qual os atributos removidos são funcionalmente dependentes.

EX:

Relação fora da 2. FN.:

Item-Venda (NrVenda, CodProd, Qtde, DescrProd, qtdeEst)

chaves candidatas: NrVenda+CodProd.

NrVenda+CodProd ->Qtde

CodProd -> QtdeEst

CodProd ->DescrProd

Colocando na 2. FN:

Item-Venda (NrVenda, CodProd, Qtde, descrProd, qtdeEst)

chaves candidatas: NrVenda+CodProd.

- Dependência total:

NrVenda+CodProd ->Qtde

então: **Item-Venda** (NrVenda,codprod,Qtde)

- CodProd -> QtdeEst

CodProd -> DescrProd

então: **Produto** (CodProd,DescrProd,QtdeEst)

• 3ª. Forma Normal:

Uma relação está na 3a. FN se está em 2a. FN e qualquer coluna fora de qualquer chave candidata não é dependente transitiva desta (cada atributo for funcionalmente dependente apenas dos atributos componentes da chave primária ou se todos os seus atributos não chave forem independentes entre si).

Procedimentos para colocar uma relação na 3a FN:

1.Identificar todos os atributos que são funcionalmente dependentes de outros atributos não chave;

2.Remove-os e criar uma nova relação com os mesmos.

3.A chave primária da nova relação será o atributo do qual os atributos removidos são funcionalmente dependentes.

Ex:

Fora da 3. FN:

Aluguel (RegCasa, Endereco, ValorAluguel, R.G.,NomeInq)

chaves candidatas: RegCasa, Endereco

RegCasa -> valorAluguel, R.G., NomeInq {Endereco}

Endereco -> valorAluguel, R.G., NomeInq {RegCasa}

RegCasa -> R.G. e R.G. -> NomeInq porém RG -> RegCasa

Normalizando (3a. F.N.):

Aluguel (RegCasa, Endereco, ValorAluguel, RG)

Inquilino (R.G., NomeInq)

• Tabelas Dependentes

Uma tabela Dependente é uma tabela não primária que tem, pelo menos, um componente de chave primária que não é chave estrangeira ou componente de chave estrangeira.

Ex:

Bloco	<i>Bloco-Sala</i>	
L	Bloco	Número Sala
M	L	108
L	L	109
	M	108

Regras:

- Objetos (entidade) dependentes não existem sem o objeto pai
- Objetos (entidade) dependentes não podem pertencer a mais que um objeto pai
- Objetos (entidade) dependentes não podem ser movidos de um para outro objeto pai.

"As tabelas dependentes sempre modelam coisas que estão verdadeiramente dependente de uma entidade ou dependente de nível superior:"

Ex:

	<i>Bloco-Sala</i>		<i>Universidade-Bloco-Sala</i>		
Universidade	Bloco	Número Sala	Universidade	Bloco	Número Sala
UEPG	L	108	UEPG	L	108
UEL	L	109	UEPG	L	109
	M	108	UEL	M	108

A chave-primária composta de uma tab.dependente pode vir a ser uma chave estrangeira composta em algum relacionamento.

Ex:

Acadêmico

Ra	Nome	Bloco	Número Sala
9100	João	L	108
9200	Pedro	L	109
9300	Gabriel	M	108

Como as tabelas dependentes afetam o projeto do Banco de Dados?

É muito importante diferenciar tabelas verdadeiramente dependentes de tabelas primárias modeladas erroneamente e as anomalias verificadas na formação da chave primária destas tabelas.

Na tabela abaixo: avalie as 3 primeiras formas normais e possíveis anomalias de manutenção.

Departamento-Funcionario

* Código Depto	* Código do Funcionário	Nome Empregado
VEN	001	Joao
VEN	002	Miguel
VEN	003	Maria
COM	001	Gabriel

1. Se transferirmos o funcionário João para o Departamento COM (Compras)?

2. Quantos funcionários podemos ter no máximo em cada departamento?

Isto é consequência de Departamento-Funcionário não ser uma tabela dependente ela é uma tabela primária. Um funcionário pode ser movido e até mesmo existir (ser recrutado) antes de determinarmos seu departamento.

Esta anomalia de formação de chave-primária é denominada SUPER-CHAVE ou CHAVE INTELIGENTE pois era usada comumente em sistemas para facilitar o processamento da informação, como saber pelo último código de cada departamento quantos funcionários existem neste, obviamente antes da facilidade das linguagens de consulta. Devido a sistemática manual de muitos usuários e até exigência é comum manter-se este tipo de composição, porém não como chave primária (para não prejudicar o projeto do BD) mas apenas como atributos que não se permite duplicação.

Ex:

Fucionário

* Código do Funcionário	Nome Empregado	Código Depto	Seqüência
001	Joao	VEN	1
002	Miguel	VEN	2
003	Maria	VEN	3
004	Gabriel	COM	1

• A Álgebra Relacional

A **álgebra relacional** é uma coleção de operações canônicas que são utilizadas para manipular as relações. Estas operações são utilizadas para selecionar tuplas de relações individuais e para combinar tuplas relacionadas de relações diferentes para especificar uma consulta em um determinado banco de dados. O resultado de cada operação é uma nova operação, a qual também pode ser manipulada pela álgebra relacional.

Todos os exemplos envolvendo álgebra relacional implicam na utilização do banco de dados descrito no apêndice A.

• A Operação *Select*

A operação **select** é utilizada para selecionar um subconjunto de tuplas de uma relação, sendo que estas tuplas devem satisfazer uma **condição de seleção**. A forma geral de uma operação **select** é:

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

A letra grega σ é utilizada para representar a operação de seleção; **<condição de seleção>** é uma expressão *booleana* aplicada sobre os atributos da relação e **<nome da relação>** é o nome da relação sobre a qual será aplicada a operação **select**.

Exemplos:

$$\text{consulta1} = \sigma_{\text{salário} < 2.500,00} (\text{EMPREGADO})$$

gera a seguinte tabela como resultado:

Tabela consulta1					
Nome	<u>RG</u>	CIC	Deppto.	RG Supervisor	Salário
Ricardo	30303030	33333333	2	10101010	2.300,00
Renato	50505050	55555555	3	20202020	1.300,00

$$\text{consulta2} = \sigma_{(\text{relação} = \text{"Filho"}) \text{ .and. } (\text{sexo} = \text{"Feminino"})} (\text{DEPENDENTES})$$

gera a seguinte tabela como resultado:

Tabela consulta2				
<u>RG Responsável</u>	<u>Nome Dependente</u>	Dt. Nascimento	Relação	Sexo
30303030	Adreia	01/05/90	Filho	Feminino

As operações relacionais que podem ser aplicadas na operação **select** são:

$<, >, \leq, \geq, =, \neq$

além dos operadores booleanos:

and, or, not.

A operação **select** é unária, ou seja, só pode ser aplicada a uma única relação. Não é possível aplicar a operação sobre tuplas de relações distintas.

• A Operação Project

A operação **project** seleciona um conjunto determinado de colunas de uma relação. A forma geral de uma operação **project** é:

$$\pi_{\langle \text{lista de atributos} \rangle} (\langle \text{nome da relação} \rangle)$$

A letra grega π representa a operação **project**, **<lista de atributos>** representa a lista de atributos que o usuário deseja selecionar e **<nome da relação>** representa a relação sobre a qual a operação **project** será aplicada.

Exemplos:

$$\text{consulta3} = \pi_{\text{Nome, Dt. Nascimento}} (\text{DEPENDENTES})$$

gera a seguinte tabela como resultado:

Tabela consulta3	
<u>Nome Dependente</u>	Dt. Nascimento
Jorge	27/12/86
Luiz	18/11/79
Fernanda	14/02/69
Angelo	10/02/95
Adreia	01/05/90

• Sequencialidade de Operações

As operações **project** e **select** podem ser utilizadas de forma combinada, permitindo que apenas determinadas colunas de determinadas tuplas possam ser selecionadas.

A forma geral de uma operação sequencializada é:

$$\pi_{\langle \text{lista de atributos} \rangle} (\sigma_{\langle \text{condição de seleção} \rangle} (\langle \text{nome da relação} \rangle))$$

Veja o seguinte exemplo:

$$\text{consulta4} = \pi_{\text{nome, depto., salario}} (\sigma_{\text{salario} < 2.500,00} (\text{EMPREGADO}))$$

produz a tabela a seguir como resultado:

Tabela consulta4		
Nome	Depto.	Salário
Ricardo	2	2.300,00
Renato	3	1.300,00

A **consulta4** pode ser reescrita da seguinte forma:

$$\text{consulta5} = \sigma_{\text{salario} < 2.500,00} (\text{EMPREGADO})$$

Tabela consulta5					
Nome	<u>RG</u>	CIC	Depto.	RG Supervisor	Salário

Ricardo	30303030	33333333	2	10101010	2.300,00
Renato	50505050	55555555	3	20202020	1.300,00

consulta6 = $\pi_{\text{nome, depto., salario}}$ (CONSULTA5)

Tabela consulta6		
Nome	Depto.	Salário
Ricardo	2	2.300,00
Renato	3	1.300,00

porém é mais elegante utilizar a forma descrita na **consulta4**.

• Operações Matemáticas

Levando em consideração que as relações podem ser tratadas como conjuntos, podemos então aplicar um conjunto de operações matemáticas sobre as mesmas. Estas operações são: **união** (\cup), **intersecção** (\cap) e **diferença** ($-$). Este conjunto de operações não é unário, ou seja, podem ser aplicadas sobre mais de uma tabela, porém, existe a necessidade das tabelas possuírem tuplas exatamente do mesmo tipo.

Estas operações podem ser definidas da seguinte forma:

- **união** - o resultado desta operação representada por $R \cup S$ é uma relação **T** que inclui todas as tuplas que se encontram em **R** e todas as tuplas que se encontram em **S**;
- **intersecção** - o resultado desta operação representada por $R \cap S$ é uma relação **T** que inclui as tuplas que se encontram em **R** e em **S** ao mesmo tempo;
- **diferença** - o resultado desta operação representada por $R - S$ é uma relação **T** que inclui todas as tuplas que estão em **R** mas não estão em **S**.

Leve em consideração a seguinte consulta:

Selecione todos os empregados que trabalham no departamento número 2 ou que supervisionam empregados que trabalham no departamento número 2.

Vamos primeiro selecionar todos os funcionários que trabalham no departamento número 2.

consulta7 = $\sigma_{\text{depto} = 2}$ (EMPREGADOS)

Tabela consulta7					
Nome	RG	CIC	Depto.	RG Supervisor	Salário
Fernando	20202020	22222222	2	10101010	2.500,00
Ricardo	30303030	33333333	2	10101010	2.300,00
Jorge	40404040	44444444	2	20202020	4.200,00

Vamos agora selecionar os supervisores dos empregados que trabalham no departamento número 2.

consulta8 = $\pi_{rg_supervisor}$ (CONSULTA7)

Tabela consulta8
RG Supervisor
10101010
20202020

Vamos projetar apenas o rg dos empregados selecionados:

consulta9 = π_{rg} (CONSULTA7)

Tabela consulta9
RG
20202020
30303030
40404040

E por fim, vamos unir as duas tabelas, obtendo o resultado final.

consulta10 = CONSULTA8 \cup CONSULTA9

Tabela consulta10
RG
20202020
30303030
40404040
10101010

Leve em consideração a próxima consulta:

selecione todos os empregados que desenvolvem algum projeto e que trabalham no departamento número 2;

Vamos primeiro selecionar todos os empregados que trabalham em um projeto.

consulta11 = $\pi_{rg_empregado}$ (EMPREGADO/PROJETO)

Tabela consulta11
RG Empregado
20202020
30303030
40404040

50505050

Vamos agora selecionar todos os empregados que trabalham no departamento 2.

consulta12 = $\pi_{rg} (\sigma_{depto = 2} (EMPREGADOS))$

Tabela consulta12
<u>RG</u>
20202020
30303030
40404040

Obtemos então todos os empregados que trabalham no departamento 2 e que desenvolvem algum projeto.

consulta13 = CONSULTA11 \cap CONSULTA12

Tabela consulta13
<u>RG</u>
20202020
30303030
40404040

Leve em consideração a seguinte consulta:

selecione todos os usuários que não desenvolvem projetos;

consulta14 = $\pi_{rg_empregado} (EMPREGADO/PROJETO)$

Tabela consulta14
<u>RG Empregado</u>
20202020
30303030
40404040
50505050

consulta15 = $\pi_{rg} (EMPREGADOS)$

Tabela consulta15
<u>RG</u>
10101010
20202020
30303030
40404040

50505050

consulta16 = CONSULTA15 – CONSULTA14

Tabela consulta16
RG
10101010

• Produto Cartesiano

O **produto cartesiano** é uma operação binária que combina todas as tuplas de duas tabelas. Diferente da operação **união**, o **produto cartesiano** não exige que as tuplas das tabelas possuam exatamente o mesmo tipo. O **produto cartesiano** permite então a consulta entre tabelas relacionadas utilizando uma condição de seleção apropriada. O resultado de um **produto cartesiano** é uma nova tabela formada pela combinação das tuplas das tabelas sobre as quais aplicou-se a operação.

O formato geral do **produto cartesiano** entre duas tabelas **R** e **S** é:

R X S

Leve em consideração a seguinte consulta:

encontre todos os funcionários que desenvolvem projetos em Campinas;

consulta16 = EMPREGADO/PROJETO X PROJETO

Tabela consulta16				
RG Empregado	Número Projeto	Nome	Número	Localização
20202020	5	Financeiro 1	5	São Paulo
20202020	5	Motor 3	10	Rio Claro
20202020	5	Prédio Central	20	Campinas
20202020	10	Financeiro 1	5	São Paulo
20202020	10	Motor 3	10	Rio Claro
20202020	10	Prédio Central	20	Campinas
30303030	5	Financeiro 1	5	São Paulo
30303030	5	Motor 3	10	Rio Claro
30303030	5	Prédio Central	20	Campinas
40404040	20	Financeiro 1	5	São Paulo
40404040	20	Motor 3	10	Rio Claro
40404040	20	Prédio Central	20	Campinas
50505050	20	Financeiro 1	5	São Paulo
50505050	20	Motor 3	10	Rio Claro
50505050	20	Prédio Central	20	Campinas

Vamos agora selecionar as tuplas resultantes que estão devidamente relacionadas que são as que possuem o mesmo valor em número do projeto e número e cuja localização seja 'Campinas'.

consulta17 = $\pi_{rg_empregado, \text{ número}} (\sigma_{(número_projeto = número) \text{ .and. } (localização = 'Campinas')})$ (CONSULTA16)

Tabela consulta17	
RG	Número
40404040	20
50505050	20

A operação **produto cartesiano** não é muito utilizada por não oferecer um resultado otimizado. Veja o item seguinte.

• Operação Junção

A operação **junção** atua de forma similar á operação **produto cartesiano**, porém, a tabela resultante conterá apenas as combinações das tuplas que se relacionam de acordo com uma determinada condição de junção. A forma geral da operação **junção** entre duas tabelas **R** e **S** é a seguinte:

$$R \bowtie_{\langle \text{condição de junção} \rangle} S$$

Leve em consideração a consulta a seguir:

encontre todos os funcionários que desenvolvem projetos em Campinas;

consulta18 = EMPREGADOS/PROJETOS $\bowtie_{número_projeto = número}$ PROJETO

Tabela consulta18				
RG Empregado	Número Projeto	Nome	Número	Localização
20202020	5	Financeiro 1	5	São Paulo
20202020	10	Motor 3	10	Rio Claro
30303030	5	Financeiro 1	5	São Paulo
40404040	20	Prédio Central	20	Campinas
50505050	20	Prédio Central	20	Campinas

consulta19 = $\sigma_{localização = 'Campinas'}$ (CONSULTA18)

Tabela consulta18				
RG Empregado	Número Projeto	Nome	Número	Localização
40404040	20	Prédio Central	20	Campinas
50505050	20	Prédio Central	20	Campinas

ORGANIZAÇÃO DE ARQUIVOS

É a forma com que os registros estão fisicamente organizados e armazenados em um arquivo, possibilitando diferentes modos de acessos e tratamento por parte da aplicação. Os tipos de organizações disponíveis, bem como as maneiras como podem ser acessadas estão descritas abaixo:

• Seqüencial

Tipo de organização em que os registros estão armazenados lado a lado. O único modo de acesso possível é o seqüencial, ou seja, para acessarmos o registro N, temos que fazer N - 1 acessos. A única técnica utilizada para atualizar um arquivo seqüencial é a técnica de balanced-line.

• Indexado

Tipo de organização em que os registros são identificados por um índice, chamado de chave do acesso ou chave do registro ou chave principal, que representa o valor com que as demais informações são identificadas no arquivo. Esta chave de acesso deverá ser única para cada registro, ou seja, não é possível ter dois registros com o mesmo valor de chave.

ESTRUTURA INTERNA

O arquivo de organização indexada possui duas áreas:

- 1) Área de Índices - Contém a chave de acesso e o endereço de cada registro na área de dados. Todo acesso aos registros será feito através desta área, que é mantida sempre ordenada, pela chave de acesso, pelo Sistema Operacional.
- 2) Área de Dados - Armazena os demais campos de todos os registros do arquivo, independente de qualquer ordenação ou seqüência.

MODO DE ACESSO NA ORGANIZAÇÃO INDEXADA

- Seqüencial - É feito através da leitura seqüencial da área de índices do arquivo. Para cada ocorrência desta área (equivalente a um índice), será obtido o endereço físico do registro deste índice na área de dados e com este endereço lido o registro. Na recuperação seqüencial de um arquivo, os dados serão lidos sempre ordenados pela chave.
- Aleatório ou Randômico - Caracteriza-se por não existir uma seqüência lógica na recuperação dos registros, ou seja, após a leitura do último registro, pode ser lido o quinto ou primeiro, ou qualquer outro, uma vez que, esta seqüência é determinada pelo programa de aplicação. No processo de leitura, o Sistema Operacional localiza o valor da chave desejada na área de índices, identificando o endereço do registro correspondente na área de dados, possibilitando o acesso, por este endereço, para a recuperação do registro.

PRINCIPAIS OPERAÇÕES PARA ATUALIZAÇÃO DOS DADOS

Inclusão - O Sistema Operacional localiza na área de índice o local onde será armazenado a chave a ser incluída, reorganizando se necessário as demais chaves mantendo-as sempre ordenadas. Após este processo, ele procura na área de dados, um endereço disponível para armazenamento do registro. Identificado o endereço, o registro será incluído e o valor do seu endereço, atualizado na área de índices.

Alteração - O Sistema Operacional localiza na área de índices, o valor da chave de acesso que se deseja

atualizar, identificando o endereço do registro na área de dados. Após esta operação, ele atualiza os dados do registro no endereço correspondente.

Exclusão - O Sistema Operacional remove fisicamente da área de índices a chave de acesso do registro que será deletado e realiza uma exclusão lógica do registro na área de dados.

• Seqüencial Indexado

É o tipo de organização em que, semelhante à organização indexada, os registros são identificados por um índice, chamado de chave do acesso ou chave do registro ou chave principal, que representa o valor com que as demais informações do registro será identificado no arquivo. Esta chave de acesso deverá ser única para cada registro, ou seja, não é possível ter dois registros com o mesmo valor de chave.

ESTRUTURA INTERNA

O arquivo de organização seqüencial-indexada possui duas áreas:

- 1) Área de Índices - Contém o valor da maior chave de acesso existente em cada bloco de registros da área de dados e o endereço deste bloco nesta área.
- 2) Área de Dados - Contém, agrupados em blocos, os registros que compõem o arquivo, sendo mantidos ordenados dentro de cada um desses blocos.

MODOS DE ACESSO NA ORGANIZAÇÃO SEQÜENCIAL INDEXADA

- Seqüencial - É feito através da leitura seqüencial dos dados na área de dados.
- Aleatório ou Randômico - É feito a localização, através do valor da chave de acesso na área de índices, do bloco onde deverá estar armazenado o registro desejado, identificando o seu endereço na área de dados. Após a localização do bloco nesta área, é feita uma procura seqüencial nos registros nele existentes até a localização do registro desejado.

PRINCIPAIS OPERAÇÕES PARA ATUALIZAÇÃO DOS DADOS

Inclusão - O Sistema Operacional localiza na área de índice o bloco onde será armazenado a chave a ser incluída. Após esta operação ele localiza na área de dados o bloco desejado e inclui o registro na posição correta, isto é, ordenado.

Alteração - O Sistema Operacional localiza na área de índices o endereço do bloco onde poderá estar armazenado a chave desejada. Localizando-o, na área de dados, é feita a procura seqüencial do registro a ser atualizado.

Exclusão - O Sistema Operacional remove fisicamente da área de dados o registro desejado, após a localização do bloco na área de índices.

• Relativa ou Direta

É o tipo de organização que contém, apenas, uma área de dados para armazenamento dos registros. Estes registros são identificados por uma chave principal, cujo endereço físico para armazenamento é dado através do valor desta chave ou de um valor calculado a partir dela. Um arquivo de organização relativa tem que possuir chave de conteúdo numérico, para possibilitar o cálculo do endereço e localização do registro na área de dados.

Para encontrar o endereço de um registro o Sistema Operacional utiliza uma função que, através de um algoritmo qualquer, transforma o valor da chave em um endereço físico, onde está armazenado o registro solicitado. Eventualmente, esta função gera para dois ou mais valores de chave diferentes o mesmo endereço físico. Neste caso, o primeiro registro incluído ficará na área de dados e os demais em uma área auxiliar, chamada área de colisão, interligados através de uma lista simples. A área de colisão é uma área de dados auxiliar, onde fica armazenado todos os registros cujo cálculo do endereço é igual a de outros já armazenados.

MODOS DE ACESSO NA ORGANIZAÇÃO RELATIVA

- Sequencial - É feito através da leitura sequencial dos dados na área de dados, obedecendo a seqüência das chaves.
- Aleatório - É feito a localização do registro através do endereço físico calculado através de uma função a partir do valor da chave de acesso. Caso haja mais de um registro para o mesmo endereço, o Sistema Operacional fará pesquisa sequencial dos registro na área de colisão do arquivo.

PRINCIPAIS OPERAÇÕES PARA ATUALIZAÇÃO DOS DADOS

Inclusão - O Sistema Operacional calcula o endereço do registro, através do valor de sua chave, incluindo-o no local especificado, caso já exista registros neste endereço ele fará a inclusão na área de colisão.

Alteração - O Sistema Operacional calcula o endereço do registro, através do valor de sua chave, atualizando-o no local especificado ou na área de colisão, dependendo de onde ele esteja armazenado.

Exclusão - O Sistema Operacional calcula o endereço do registro, através do valor de sua chave, removendo-o logicamente da área de dados ou da área de colisão, dependendo de onde ele esteja armazenado.

• Invertido

É o tipo de organização que permite o acesso direto aos registros de dados através de chaves secundária (chave que não é primária), de um arquivo de dados. Esta arquivo é formado pelo conteúdo das chaves secundária e da relação das chaves primárias dos registros que a contém.

Exemplo:

Arquivo Principal: Funcionário

Chave primária ou principal - Cod. funcionário

COD. FUNCIONÁRIO	NOME	SALÁRIO
10	A	110.000
15	D	150.000
20	E	180.000
23	C	110.000
28	F	180.000
32	B	150.000
35	G	180.000

Arquivo invertido por salário:

SALÁRIO	CHAVES PRIMÁRIA
110.000	10, 23
150.000	15, 32
180.000	20, 28, 35

Este tipo de organização auxilia a recuperação de registros através de uma consulta, sendo bastante utilizado pelos Sistemas de Gerenciamento de Banco de Dados (SGBD) existentes. Ele está sempre associado a um arquivo principal, onde está armazenado o registro completo.

O processo de atualização de um arquivo invertido ocorre simultaneamente à do arquivo principal, devendo o mesmo ser feito pela própria aplicação ou pelo SGBD.

ORGANIZAÇÃO DE REGISTROS EM BLOCOS

Um arquivo pode ser visto como um coleção de registros. No entanto, uma vez que dados são transferidos entre o disco e a memória principal em unidades de um bloco, é conveniente atribuir registros a blocos de modo que um bloco único contenha registros relacionados. Se designarmos registros para blocos aleatoriamente, será comum que se faça acesso a um bloco diferente para cada registro. Por outro lado, se pudermos fazer o acesso a diversos registros desejados usando apenas um acesso a um bloco, economizaremos alguns acessos ao disco.

Uma cadeia consiste em tantos blocos quantos necessários para representar os dados, mas duas cadeias diferentes nunca repartem blocos.

GERENCIAMENTO DE BUFFER

O Buffer é aquela parte da memória principal disponível para o armazenamento de cópias de blocos do disco. O gerenciador de buffer intercepta todas as requisições de blocos do banco de dados feitas pelo restante do sistema. Se o bloco já estiver no buffer, é passado ao processo requisitante o endereço do bloco na memória principal. Se o bloco não estiver no buffer, o gerenciador lê o bloco do disco para o buffer e passa o endereço do bloco na memória ao requisitante.

Técnicas de gerência de buffer:

- Estratégia de substituição: Quando não há mais espaço disponível no buffer, um bloco precisa ser removido do buffer antes que um novo possa ser lido. Um sistema típico utiliza o esquema “menos recentemente utilizado” (LRU – Least Recently Used)¹
- Blocos Presos (“Pinned”): A fim de habilitar o sistema a recuperar-se de panes, é necessário restringir as ocasiões em que um bloco pode ser regravado no disco. Um bloco que não pode ser regravado no disco é chamado de preso. Este sistema é comum em bancos imunes a panes.
- Saídas forçadas de blocos: Existem situações em que é necessário regravar o bloco no disco mesmo que o espaço de buffer que ele ocupa não seja necessário.

INDEXAÇÃO

Um índice é uma espécie de arquivo armazenado. Para ser mais específico, é um arquivo no qual cada entrada compões-se precisamente de dois valores, um valor de dados e um ponteiro. O valor de dados é um valor para um determinado campo do arquivo indexado, e o ponteiro identifica o registro daquele arquivos que tenha o valor daquele campo.

A vantagem fundamental do índice é que ele acelera a recuperação. Entretanto há também a desvantagem - reduz a velocidade das atualizações.

O armazenamento no índice pode ser denso e não denso. A principal diferença é que, na indexação densa o índice possui a referência a todos os registros do arquivo de dados, enquanto que na indexação não densa o índice possui referência apenas para o primeiro e o último registro do bloco.

Árvore B

Um tipo de índice particularmente comum e importante é a árvoreB. Embora seja verdade de que não existe uma única estrutura de armazenamento ótima para todas as aplicações, não há dúvidas de que, se é necessário escolher uma única estrutura, então a árvore B, de uma variedade ou de outra, será a escolhida. Assim, a maioria dos sistemas relacionais suporta as árvores B como a principal forma de estrutura de armazenamento, e outros não suportam absolutamente nada, a não ser estas.

Antes que possamos explicar o que é uma árvore B, precisamos discutir uma noção preliminar, a saber, o índice multinível (ou estruturado em forma de árvore)

A solução para este problema é a mesma: ou seja, tratamos o índice simplesmente como um arquivo armazenado/regulador e construímos um índice para o mesmo.

Esta idéia pode ser colocada em prática em tantos níveis quanto necessário (em geral três níveis na prática); um arquivo teria de ser muito grande para necessitar de mais do que três níveis de indexação. Cada nível de índice atua com um índice não denso para o nível inferior. Como mencionado, as árvores B, de um ou outro modo, são agora provavelmente a estrutura de armazenamento mais utilizada nos sistemas modernos de bancos de dados (relacionais ou não)

As estruturas de árvore em geral tem problema, ou seja, tais inserções e eliminações podem causar desbalanceamento da árvore. Uma árvore é desbalanceada quando todas as folhas de nós não se encontram no mesmo nível

HASH

O acesso hash é a técnica que proporciona um rápido acesso direto ao registro armazenado, baseado um determinado valor de um certo campo. O campo em questão, é em geral, mas não necessariamente, a chave primária. Essa técnica funciona como segue:

Cada registro armazenado é colocado no banco de dados em uma localização, cujo endereço (RID, ou talvez ou número do bloco) é computado como função (a função Hash). O endereço computado é denominado endereço Hash.

Deve ter ficado claro que, pela descrição anterior, que o acesso hash difere da indexação, uma vez que um determinado arquivo armazenado poderá ter qualquer quantidade de índices, mas só pode ter um estrutura Hash. Uma outra desvantagem é a possibilidade de colisões. Uma outra abordagem ao problema da colisão, talvez mais encontrada na prática, seria de tratar o resultado a partir da função hash, digamos "a" como endereço de armazenamento, não do registro de dados, mas do "ponto âncora". O ponto âncora no endereço de armazenamento "a" é então considerado a cabeça de uma cadeia de ponteiros (uma cadeia de colisões), unindo todos os registros que colidem a um "a". As colisões dentro de uma determinada cadeia de colisões serão mantidas em um sequência de campo hash, a fim de simplificarem as buscas subsequentes.