Centro de Ciências Exatas, Ambientais e de Tecnologias

Curso de Especialização em Análise de Sistemas com Ênfase em Arquitetura Cliente-Servidor

Tecnologia de Banco de Dados para Aplicações Cliente/Servidor

Modelagem de Dados

1º Semestre de 2003

Prof. André Luís dos R.G. de Carvalho

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS
CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGIAS
PLANO DE ENSINO

Curso:	Disciplina:	Docente:	C.H. Semanal	Período Letivo:
Especialização em Análise de Sistemas c/ Ênfase em Arquitetura Cliente-Servidor	Tec.de BD p/ Apl. Cliente/Servidor	André Luís dos R.G. de Carvalho	04 (quatro horas aula)	1º Semestre de 2003

Objetivo Geral da Disciplina:

- Estudar conceitos e técnicas para o suporte ao desenvolvimento de aplicações distribuídas em ambiente com banco de dados e arquitetura cliente-servidor.

Avaliação:

- Haverá uma série de atividades que serão desenvolvidas ao longo do semestre (A_i).
- A média final será dada pela média das notas das atividades A_i, ponderadas com pesos compatíveis com sua complexidade.

Freqüência:

- Nenhum abono de falta será concedido pelo professor. Problemas neste sentido deverão ser encaminhados à secretaria que tomará todas as providências no sentido de encaminhar a solução dos mesmos.

Bibliografia Utilizada :

- Sistemas de Bancos de Dados
  Korth, H.F.; e Silberschatz
- Introdução a Sistemas de Bancos de Dados
  Date, C.J.
- Projeto Lógico e Projeto Físico de Bancos de Dados
  Setzer, V.W.
- Banco de Dados: Fundamentos, Projeto e Implementação
  Kroenke, D.M.
- The Entity-Relationship Approach to Logical Database Design
  Chen, P.
- Computer Database Organization
  Martin, J.
- Fundamental Concepts of Information Modeling
  Flavin, M.

Núm. de Aulas	Conteúdos Selecionados	Estratégias/ Técnicas de Ensino	Forma de Avaliação
20	• Banco de dados: Revisão de conceitos; • Motivação e fundamentos; • Modelo relacional; • Modelagem de dados.	• Aulas expositivas • Aulas práticas • Estudos de caso • Estudo dirigido • Trabalhos práticos em campo	• Trabalhos práticos
8	• Aplicações cliente-servidor: revisão de conceitos; • Motivação da arquitetura cliente-servidor; • Principais desafios no desenvolvimento de aplicações cliente-servidor; • Distribuição de dados e de processos; • Ambientes de desenvolvimento; • Técnicas de projeto de sistemas distribuídos; • Confiabilidade e eficiência em sistemas distribuídos.	• Estudo dirigido • Trabalhos práticos	• Trabalhos práticos
20	• Linguagens de consulta: revisão de conceitos; • SQL (estrura básica da linguagem, tratamento de tuplas duplicadas, operações de conjunto, predicados e junções, testes de relações vazias, comparação de conjuntos, apresentação ordenada de tuplas, funções de agregação, tratamento de valores nulos, manutenção de tabelas e de dados).	• Aulas expositivas • Aulas práticas • Estudo dirigido • Trabalhos práticos	• Trabalhos práticos
8	• Controle de concorrência; • Schedules; • Serialização; • Protocolos baseados em bloqueio; • Protocolos baseados em timestamp; • Técnicas de validação; • Granularidade de bloqueio.	• Estudo dirigido • Trabalhos práticos	• Trabalhos práticos
8	• Processamento de transações; • Modelo de memória; • Recuperação de falhas em transações; • Tratamento de deadlock; • Transações de longa duração; • Definição de transações em SQL.	• Estudo dirigido • Trabalhos práticos	• Trabalhos práticos
8	• Recuperação de falhas; • Classificação de falhas; • Modelo de transações; • Recuperação baseada em log; • Checkpoints; • Shadow paging.	• Estudo dirigido • Trabalhos práticos	• Trabalhos práticos

Índice

Informação, Dado e Elemento de Dado

Quando nos propomos a falar de bancos de dados, sem dúvida não poderemos deixar de esbarrar no conceito de **informação**. Dizemos **informação** qualquer **conhecimento** de que não disponhamos, mas de que tenhamos necessidade.

Em sendo conhecimento, a informação só existe na mente das pessoas, constituindo uma parte de um quadro mental de referência. A **informação** está sempre associada a um ente específico concreto ou abstrato, que poderíamos chamar de ente de referência.

Hoje em dia, mais do que nunca, **informação** constitui um recurso básico e imprescindível para toda e qualquer atividade humana. Informações completas e oportunas acabam por diferenciar uma companhia de seus concorrentes, vindo até mesmo a ter implicações não desprezíveis no sucesso ou insucesso de uma empresa.

Dizemos **dado** qualquer conjunto de símbolos organizados com intencionalidade para representar a informação fora da mente humana, possibilitando seu armazenamento e transferência.

No contexto da computação, quando falamos a respeito de armazenar informações do mundo real em arquivos de dados, nos referimos à armazenagem de coisas que se conheça a cerca das entidades que povoam o mundo real, visando construir uma representação destas mesmas entidades através de seus atributos.

Neste processo de representação, assumimos de maneira implícita que todos os entes do mundo real sejam idênticos a um ente de referência o que nos possibilita construir uma representação padrão.

Um dado representa primariamente informações sobre uma entidade em uma realidade, e secundariamente um próprio pedaço da realidade.

Quando extraímos informações dos entes do mundo real e as representamos através de dados em um banco de dados, acabamos por restringir a informação real que virtualmente poderia requerer uma quantidade infinita de dados para representá-la com propriedade.

Isto sem falar a respeito das coisas que são deixadas de lado a fim de conseguir maior semelhança com o ente de referência e em última instância a construção de uma representação padrão.

Desta maneira, **dados** não são mais que uma imagem pálida do que é a informação viva do mundo real.

Assim, quando obtemos de um sistema de informação dados a respeito de algum ente do mundo real, deve-se realizar sobre eles um processo inverso de enriquecimento semântico, de maneira que se consiga associar os dados obtidos a partir de um banco de dados às entidades do mundo real que lhes deram origem.

Não sendo mais que uma representação das entidades do mundo real, podemos dizer que podem existir inúmeras representações possíveis, sendo cada uma mais ou menos adequada a cada situação.

Chamamos **elemento de dado** a qualquer conjunto de símbolos com um significado específico que compõe um dado, mas que não constitui informação completa.

A Abordagem certa para cada Organização

Estudos a respeito da evolução das empresas, no que diz respeito ao uso do processamento de dados, apontam seis estágios pelos quais passam as empresas: iniciação, contágio, controle, integração, administração de dados e maturidade.

Segundo estes estudos, esta evolução é gradativa, reflete a progressiva maturação dos profissionais de informática e dos usuários, e não pode ser artificialmente apressada.

Nos primeiros estágios a empresa se preocupa mais com funções, mudando o enfoque, gradativamente, até que, entre o terceiro e o quarto estágio, começa a se preocupar menos com funções e mais com dados.

O estágio em que a empresa se encontra pode sugerir a abordagem mais adequada para suprir suas necessidades.

Desenvolvimento de Programas

Tradicionalmente, existem duas maneiras para se abordar o desenvolvimento de um programa: a partir das funções e a partir dos dados.

Uma boa razão para a abordagem funcional é o fato de um programa ser uma função. Assim, pode-se entender sua análise como sendo a atividade de decompor esta função em suas funções componentes, estas em suas componentes menores e assim, sucessivamente.

Se, por um lado, constatamos que a abordagem funcional parece ser uma abordagem bastante natural para programas, por outro lado, temos o fato de que dados são muito mais estáveis que funções e, assim sendo, a abordagem pelos dados normalmente nos livra de programas isolados e arquivos redundantes, incentivando o compartilhamento de dados.

Numa abordagem moderna, a tarefa de desenvolvimento de sistemas pode ser enxergada em três dimensões: a dimensão das funções, a dimensão dos dados e a dimensão das transições de estado. Recomenda-se a seguinte ordem para a análise de cada uma dessas três dimensões:

- Definição preliminar do modelo de armazenamento de dados;
- Definição do modelo funcional com revisão do modelo de armazenamento de dados;
- Definição do diagrama de estados do sistema, isto é, do diálogo do sistema através de janelas, botões, menus, sub-menus, comandos, teclas, etc.

Como o objetivo deste curso é o projeto de BD, restringiremos nosso estudo somente à questão do armazenamento.

Antes da Tecnologia de BDs

Procedimento

Podemos dizer de maneira simplificada, que o método tradicional de desenvolvimento de sistemas pode ser resumido em entrada, processamento, e saída, sendo a entrada e a saída, representadas por um conjunto de arquivos de dados, cada qual constituído por um conjunto de registros de dados, cada qual constituído por um conjunto de campos que representam elementos de dados.

Entre as características desta abordagem tradicional, podemos ressaltar o incentivo ao desenvolvimento de múltiplas aplicações isoladas, cada qual mantendo seus próprios arquivos de dados.

Problemas Existentes

Fazendo uma análise desta abordagem, podemos identificar nela diversos inconvenientes, a saber:

1. A falta de integração entre os arquivos;
2. O elevado grau de redundância de dados entre os arquivos das diversas aplicações, o que leva (a) a uma ocupação inadequada de espaço; e (b) à possível incompatibilidade entre as diversas ocorrências do mesmo dado;
3. O elevado índice de reorganizações ("sorts" e "merges") para a obtenção de ordenação apropriada para a emissão de relatórios;
4. A duplicidade de procedimentos;
5. A dificuldade em garantir a segurança dos dados; e
6. A dependência dos dados por parte dos sistemas de aplicação.

Sistemas de Banco de Dados

Nem todos os sistemas de computação fazem uso extensivo de dados, mas, para aqueles que fazem, este elemento sistêmico é, na maioria das vezes, um elemento de importância vital, o que torna importante resolver os problemas que existiam antes desta tecnologia.

Um sistema de computação que mantenha os dados de uma organização de acordo com a estrutura e o ambiente real da organização poderia garantir isso. Para permitir esta modelagem adequada, surge a tecnologia de bancos de dados.

O que é afinal um sistema de banco de dados? Poderíamos dizer que um sistema de banco de dados é um sistema de armazenamento de dados em ambiente computacional, cujo objetivo global é registrar e manter informações de maneira a eliminar os problemas que podem ser

identificados em sistemas desenvolvidos segundo a abordagem tradicional, e a conferir aos dados as características apontadas acima como necessárias.

Entendemos por **banco de dados** um depósito de dados armazenados em geral de forma integrada e compartilhada.

Podemos imaginar um banco de dados como sendo a unificação de diversos tabelas que de outra forma seriam distinguíveis, eliminando total ou parcialmente (mas de qualquer forma mantendo sob controle) qualquer redundância entre aqueles tabelas.

Os dados em um banco de dados podem em geral ser compartilhados entre diversas aplicações e usuários diferentes. Na verdade, a possibilidade de compartilhamento poderia ser vista como uma conseqüência da integração do banco de dados.

Desta maneira, informações sobre peças que uma empresa fabrique poderiam ser compartilhadas pelo setor de fabricação, pelo setor de controle de qualidade e pelo setor de vendas.

O termo **compartilhado** é em geral estendido para descrever compartilhamento no tempo, isto é, a possibilidade de diversos usuários ou aplicações acessarem em conjunto o banco de dados ao mesmo tempo.

Um sistema de banco de dados envolve quatro componentes maiores: hardware, software, usuários e os dados propriamente ditos.

Para que seja possível a integração e o compartilhamento de informações em um banco de dados, faz-se necessário que os dados residam em dispositivos de acesso direto como discos ou fitas.

Em geral, para controlar o acesso e o uso das informações em um banco de dados, entre o banco de dados físico e os sistemas aplicativos que os manipulam, encontra-se um software normalmente chamado de sistema gerenciador de banco de dados ou simplesmente SGBD.

Este sistema manipula todas as solicitações dos usuários para acesso e/ou utilização dos dados do BD, isolando os sistemas aplicativos dos detalhes de hardware, garantindo a integridade de atualizações aos dados e promovendo o acesso controlado (proteção) aos dados. Em outras

palavras, o SGBD fornece uma visão do BD a um bom nível de abstração, suporta operações de usuários, garante a integridade dos dados, bem como a segurança deles.

Para alcançar a independência de dados, o SGBD mantém o BD e provê comandos para acessá-lo, sendo este acesso sempre feito através do SGBD. O SGBD mantém uma visão estável dos dados, que não é afetada por mudanças na organização. Para tal, o SGBD deve distinguir entre a organização lógica e a organização física dos dados.

Entendemos por organização lógica dos dados, a maneira pela qual os dados são **vistos** pelos programas aplicativos; e por organização física dos dados, o modo como estes se encontram armazenados em dispositivos físicos.

Enquanto os programas aplicativos **devem** conhecer a organização lógica dos dados, **é melhor que ignorem** a organização física dos dados.

Como vantagens do controle centralizado dos dados por parte do SGBD poderíamos citar:

- Reduzir e/ou manter a redundância de informações sob controle (redundâncias por um lado provocam desperdício de espaço de armazenamento e podem causar inconsistências, mas por outro podem vir a melhorar a eficiência de um sistema de banco de dados);
- Promover o compartilhamento dos dados (não somente para as aplicações existentes BD, mas também para novas aplicações);
- Manter restrições de segurança (a centralização dos dados permite um maior controle destes, e os SGBD em geral permitem a manutenção de diversas permissões de acesso - acréscimo, remoção, modificação, recuperação, etc - e a diversos níveis);
- Garantir a integridade (restrições de integridade ou procedimentos de validação podem ser definidos e serão executados pelo SGBD sempre que for tentada uma atualização do BD; é importante chamar a atenção para o fato de que a integridade dos dados é ainda mais importante em sistemas de banco de dados do que em sistemas convencionais, pois uma vez que os dados são compartilhados, torna-se possível que o mal uso que alguma aplicação possa fazer dos dados do banco de dados tenha um efeito não somente local à aplicação, mas possivelmente afetar diversas outras aplicações);

- Obter independência dos dados (uma aplicação enxerga os dados do banco de dados da maneira que lhe for conveniente, sem preocupações com o formato, localização ou método de acesso aos dados. Aplicações diferentes poderão ter visões diferentes do BD permitindo alterações no BD com menor impacto sobre as aplicações, tornando possíveis modificações no software ou no hardware sem necessidade de reprogramações e facilitando o uso compartilhado dos dados uma vez que permite uma visão adequada às necessidades das diversas aplicações sobre o mesmo conjunto de dados).

Administração de Dados

Aceita-se, com naturalidade, o fato de que uma empresa, para alcançar seus objetivos, deve usar de maneira eficaz e racional seus recursos.

Estes recursos podem ser classificados didaticamente em: recursos humanos, recursos financeiros, máquinas e equipamentos, e tecnologia.

Nas empresas, os órgãos encarregados de administrar estes recursos têm funções como:

1. Recrutar, selecionar e captar novos recursos;
2. Garantir a plena disponibilidade dos recursos administrados;
3. Evitar o uso indevido dos recursos da empresa; e
4. Promover o uso eficaz dos recursos.

Se observarmos o crescente nível de dependência das empresas com relação aos dados, fica evidente que a cada dia se torna mais necessário acrescentar a esta lista os dados, já que, sendo assim tão vitais, é fundamental que os dados se caracterizem por sua disponibilidade, facilidade de interpretação, atualidade, exatidão e precisão.

Para gerir os dados, enquanto recursos, torna-se necessário um profissional específico: o Administrador de Dados. Dentre suas principais funções, podemos destacar:

1. Construir e manter o modelo conceitual dos dados da empresa;
2. Manter um registro com a descrição de todos os dados existentes na empresa;
3. Divulgar aos usuários e profissionais de informática as descrições dos dados da empresa;

4. Promover a integração entre os diversos sistemas de aplicação; e
5. Facilitar o desenvolvimento de aplicações para as quais já existem dados.

Esquemas

A palavra **esquema**, no contexto de bancos de dados, deve ser entendida como sinônima de **descrição**. Desta maneira, pode-se entender que esquema lógico é uma descrição da organização lógica dos dados e um esquema físico é uma descrição da organização física dos dados.

Em uma arquitetura de dois esquemas, cada esquema lógico é definido em termos de esquema físicos existentes. Esta estrutura permite adicionar ou mudar esquemas lógicos facilmente, embora cause uma grande dependência entre os esquemas lógico e físico, i.e., mudar um esquema físico pode ser muito complicado e causar grandes mudanças nos esquemas lógicos existentes.

Em uma arquitetura de três esquemas, para isolar os esquemas lógico e físico, introduz-se um esquema intermediário. Assim tem-se:

- Um esquema externo ou lógico, que consiste dos dados tais como estes são vistos pelos programas aplicativos;
- Um esquema intermediário ou conceitual, que consiste de uma descrição global e não redundante do BD a nível conceitual;
- Um esquema interno ou físico, que consiste dos dados tais como estes são armazenados em dispositivos físicos.

Cada esquema lógico é definido em termos de esquemas conceituais e estes, por sua vez, são definidos em termos de esquemas físicos.

Uma vez que o esquema conceitual é não redundante, mudanças no esquema físico só afetam uma pequena parte do esquema conceitual, levando a uma maior simplicidade e facilidade na realização de mudanças no esquema físico nesta arquitetura do que na de dois esquemas.

Com isso, poderemos distinguir dois tipos de independência de dados: a independência lógica dos dados (mudanças na visão do BD de um programa aplicativo não afetam as visões que outros programas aplicativos possam ter do BD) e a independência física dos dados (mudanças na estrutura física do BD deixam intacto o esquema lógico, isto é, nenhum programa aplicativo tem sua visão do BD afetada).

Modelos de Dados

Entendemos por Modelo de Dados, a forma, a estrutura, que se concebe ser a organização dos dados no BD. Os três modelos mais consagrados são os modelos de rede, hierárquico e o relacional, embora já se fale muito de um quarto modelo, o modelo orientado a objetos.

O Modelo Relacional é, de longe, o modelo mais comum e amplamente empregado pelos SGBD em uso nos dias atuais, por esta razão, receberá, nesta disciplina, uma atenção especial.

O Modelo Relacional

Segundo o modelo relacional, imagina-se que os dados se organizem em tabelas, ou relações, cada qual constituída por uma série de linhas e colunas.

Cada elemento em um conjunto de dados é representado por uma linha na tabela, cada atributo dos elementos de um conjunto é representado por uma coluna na tabela e cada item de dado deste elemento é representado pelo valor da célula que ocupa na tabela.

Ex.:

Nomes dos casais que tiveram filhos no século passado

Pai	*Mae*
João Antônio Salgado Júnior	Anna Francisca de Oliveira Salgado
Francisco Monteiro de Carvalho e Silva	*Escolástica Pires Monteiro*
Antônio de Abreu Sampaio	*Maria das Dores de Carvalho Sampaio*
Antônio Monteiro de Carvalho e Silva	*Theolinda de Abreu Sampaio*

Seja t uma tabela qualquer, l uma linha qualquer de t e c uma coluna qualquer de l. Podemos, considerando o modelo relacional, fazer as seguintes afirmações:

1. l é sempre única em t, isto é, não é possível que haja em t uma outra linha exatamente idêntica a l;

2. l possui uma quantidade fixa de colunas;

3. l possui exatamente as mesmas colunas que as demais linhas de t e estas se apresentam exatamente na mesma ordem;

4. Toda linha de uma tabela, seja ela qual for, possui pelo menos uma coluna, cujo valor a identifique dentre as demais linhas da mesma tabela; (5) toda colunas atributos têm valores sempre atômicos (não há grupos que se repetem) e assumem valores em domínios de valores possíveis; o mesmo domínio pode ser usado por diversos atributos; as linhas e colunas podem ser consideradas em qualquer seqüência sem que a mudança de ordem afete as informações em qualquer momento.

Para estabelecer relações entre os dados, são empregadas Chaves Estrangeiras, ou seja, Chaves Primárias de uma tabela quando ocorrem em outra tabela.

Observe, no exemplo abaixo, como as colunas Esposo e Esposa estabelecem uma relação entre Pessoas, dependendo do seu Numero.

Ex.:

Numero e Nome das Pessoas que se casaram de 1989 a 2001

Numero	Nome
4	Adriana Rodrigues Tufaile
13	André Luís dos Reis Gomes de Carvalho
21	Antônio Oscar da Silva
32	Cleusa de Carvalho e Silva Rodrigues F
39	Deodato Eleutério Rodrigues Neto
44	Edi Aparecida Cervan
59	Giulliano Humberto Capana
63	Haroldo Lisboa Rodrigues Júnior
87	Juliana Rodrigues Tufaile
91	Lília de Carvalho e Silva Rodrigues
97	Luiz Paiva Pereira
103	Márcia Regina Santana

Data, Local e Cônjuges dos Casamentos que ocorreram de 1989 a 2001

Data	Local	Esposo	Esposa
1989-03-18 00:00:00.000	Campinas	13	91
1989-11-04 00:00:00.000	São José do Rio Preto	21	32
1989-12-23 00:00:00.000	São José do Rio Preto	39	103
1991-12-20 00:00:00.000	Araraquara	63	44
1992-03-21 00:00:00.000	São Paulo	97	4
1997-11-22 00:00:00.000	São Paulo	59	87

Mais adiante voltaremos a falar do modelo relacional, uma vez que estudaremos com mais profundidade.

Normalização

Manter os dados organizados de uma forma eficiente e estável é imprescindível. Antes da tecnologia de BDs, esta técnica representava a ferramenta mais útil que um desenvolvedor poderia lançar mão para organizar os dados de suas aplicações.

Trata-se de uma técnica que estabelece critérios para verificar se tabelas estão estruturadas convenientemente, i.e., se possuem uma estrutura estável e eficiente e, em caso negativo, indica transformações que devem ser realizadas sobre as tabelas, a fim de corrigir sua estrutura.

Dependência Funcional

O conceito de Dependência Funcional é um conceito que serve de base para todo o processo de normalização.

Podemos entendê-la como uma relação que se estabelece ou não entre conjuntos de colunas de uma tabela.

Sejam C_1, C_2 e C_3 conjuntos distintos e disjuntos de colunas de uma tabela T.

Podemos dizer que C_2 depende funcionalmente de C_1 se, e somente se, cada valor de C_1 estiver sempre associado precisamente um valor de C_2.

Em outras palavras, podemos dizer que C_2 é dependente funcionalmente de C_1 se, e somente se, para toda linha, todo valor de C_1 determinar um único valor de C_2, ou ainda, se, e somente se, todo valor de C_2 puder ser identificado de forma única dentre os demais valores de C_2 por um valor de C_1.

Podemos dizer que C_3 depende funcionalmente de forma transitiva do atributo C_1, se C_3 depender funcionalmente de C_2 e C_2 depender funcionalmente de C1.

Em outras palavras, podemos dizer que C_3 depende funcionalmente de forma transitiva do atributo C_1 quando C_3 depende indiretamente de C_1.

As Formas Normais

O processo de verificação e adequação da estrutura dos tabelas de dados se dá em fases as quais chamamos de formas normais, a saber: a Primeira Forma Normal (ou 1FN), a Segunda Forma Normal (ou 2FN) e a Terceira Forma Normal (ou 3FN).

- **Primeira Forma Normal (1FN):**

  Dizemos que um tabela T está na 1FN se, e somente se, T não possuir colunas multivaloradas, i.e., campos capazes de armazenar mais de um valor em uma mesma linha.

  Para transformar um tabela T que não se encontra na 1FN em tabelas que se encontrem na 1FN devemos:

  1. Remover de T a(s) coluna(s) multivalorada(s) nele existentes e a(s) elevar à qualidade de tabela(s); e
  2. Acrescentar a chave primária de A a essas tabelas, eventualmente, na condição de parte da chave primária (sem, por este motivo, removê-la de T).

- **Segunda Forma Normal (2FN):**

  Dizemos que uma tabela T está na 2FN se, e somente se, T estiver na 1FN e todos as suas colunas não chave forem TOTALMENTE dependentes funcionalmente da chave primária.

  Em outras palavras, para que uma tabela T esteja na 2FN, ele não pode possuir colunas não chave que dependam funcionalmente apenas de parte da chave primária.

  Para transformar uma tabela T que não se encontra na 2FN em tabelas que se encontrem na 2FN devemos:

  1. Considerar as possíveis partes P_1, P_2, ..., e P_n da chave primária de T das quais dependam funcionalmente as colunas não chave de T (caso haja alguma coluna não chave em T que seja TOTALMENTE dependente funcionalmente da chave primária de T, deve-se incluir entre estas partes a chave primária de A inteira); e

2. Para todo i, tomar P_i e as colunas não chave de T que dependerem de P_i e, com eles constituir uma nova tabela, tornando, P_i a chave primária da tabela recém constituída (T deve deixar de existir findo este processo).

- **Terceira Forma Normal (3FN):**

  Dizemos que uma tabela T está na 3FN se, e somente se, T estiver na 2FN e todas as suas colunas não chave forem dependentes funcionalmente da chave primária de forma não transitiva.

  Em outras palavras, para que um tabela T esteja na 3FN, todas as suas colunas não chave devem depender funcionalmente da chave primária de forma direta.

  Para transformar um tabela T que não se encontra na 3FN em tabelas que se encontrem na 3FN devemos:

  1. Considerar os possíveis conjuntos de colunas não chave de T, C_1, C_2, ..., e C_n, dos quais dependam outros conjuntos de colunas não chave de T; e

  3. Para todo i, tomar C_i e as colunas não chave de T que dependerem de C_i e, com eles constituir uma nova tabela.

     Nesta nova tabela, caso seja viável, C_i tornar-se-á chave primária da tabela recém constituída, caso não seja viável, poder-se-á criar uma chave artificial para a tabela recém criada.

     C_i deve continuar existindo em T no primeiro caso, já, no segundo caso, não, caso em que, a chave artificial criada deve ser incorporada a T. As colunas não chave que dependiam de C_i devem ser retiradas de T.

A título de exemplo, consideremos a tabela cuja estrutura é apresentada abaixo:

Prod {**#P**, NomeP, CorP, PesoP, FornP {#F, NomeF, CidF, DistF,QdeF}}

Devemos entender que Prod é uma tabela de produtos de uma certa empresa E.

Para cada produto em Prod, as seguintes informações: #P (o número de um produto e chave primária da tabela Prod), NomeP (o nome de um produto), CorP (a cor de um produto), PesoP (o peso de um produto) e FornP (oS fornecedoreS de um produto).

Para cada fornecedor em FornP, teremos as seguintes informações: #F (o número de um fornecedor), NomeF (o nome de um fornecedor), CidF (a cidade de um fornecedor), DistF (a distância da cidade onde fica um fornecedor até a cidade onde fica a empresa E) e QdeF (a quantidade de um produto que um fornecedor é capaz de fornecer por mês).

O tabela Prod, claramente não está na 1FN, já que possui o campo multivalorado FornP.

Passando para a 1FN, retiramos a coluna multivalorada FornP e a elevamos à condição de tabela, levando para compor sua chave primária, juntamente com #F, a chave primária da tabela Prod, #P. Veja abaixo:

```
Prod {#P, NomeP, CorP, PesoP}
Forn {#P, #F, NomeF, CidF, DistF, QdeF}
```

A tabela Prod, naturalmente, está na 2FN, já que se encontra na 1FN e não tem colunas não chave que dependam funcionalmente de parte de sua chave primária, inclusive porque, sua chave primária não é uma chave composta, i.e., não tem partes.

Já o tabela Forn, claramente, não está na 2FN; ela se encontra na 1FN, mas nem todas as colunas não chave de Forn dependem funcionalmente TOTALMENTE da chave primária; as colunas NomeF, CidF e DistF não dependem de #F, enquanto QdeF depende de #P+#F.

Assim (1) tomamos as colunas #F, NomeF, CidF e DistF e, com elas, constituímos uma outra tabela que batizamos de Forn (#F fica sendo a chave primária de Forn); e (2) tomamos as colunas #P, #F e QdeF e, com elas, constituímos uma tabela que batizamos de PF (#P+#F fica sendo a chave primária composta de PF). A tabela Forn original deixa de existir. Veja abaixo:

```
Prod {#P, NomeP, CorP, PesoP}
Forn {#F, NomeF, CidF, DistF}
PF {#P, #F, QdeF}
```

O tabela PF, naturalmente, está na 3FN, já que se encontra na 2FN e não tem colunas não chave que dependam funcionalmente de outras colunas não chave, inclusive porque, possui uma única coluna não chave.

O tabela Prod, também está na 3FN, já que se encontra na 2FN e não tem colunas não chave que dependam funcionalmente de outras colunas não chave.

Já o tabela Forn, claramente, não está na 3FN; ela se encontra na 2FN, mas nem todas as colunas não chave de Forn dependem funcionalmente DIRETAMENTE da chave primária; a coluna DistF depende funcionalmente de CidF que, por sua vez, depende funcionalmente de #F.

Assim tomamos as colunas CidF e DistF, e, com elas, constituímos uma outra tabela que batizamos de DistCids (CidF fica sendo a chave primária de DistCids). A tabela Forn original continua existindo e, nela, a coluna Cid continua existindo. Veja abaixo:

```
Prod {#P, NomeP, CorP, PesoP}
Forn {#F, NomeF, CidF}
PF {#P, #F, QdeF}
DistCids {Cid, Dist}
```

Modelagem de Dados

No processo de modelar dados, segue-se uma rotina de vários passos que conduzem do mundo real à criação de um BD.

1. O mundo real do ponto de vista formal é muito nebuloso. Este mundo é povoado por seres, fatos, coisas, etc.

   Faz-se necessária uma organização deste mundo, o que, normalmente, se dá através do levantamento de informações a respeito da porção do mundo real que é de interesse no escopo computacional em questão.

   Estas informações, organizadas ainda de uma forma informal, são constituídas por relatórios escritos em uma linguagem natural (português, inglês, etc.) e podem conter um simbolismo difícil de ser entendido.

   Este é um nível dito descritivo, pois procura-se descrever o mundo real por meio de frases.

   Não existem regras formais para desenvolver este modelo, pois tanto o mundo real quanto o modelo descritivo não são formais.

2. Vencidas todas estas dificuldades e de posse de um modelo descritivo do mundo real, poderemos passar à descrição das estruturas e transações envolvidas no modelo descritivo, construindo o modelo conceitual do BD.

O modelo aqui desenvolvido deve ser estritamente formal, o mais próximo possível dos formalismos matemáticos. Este modelo em geral é feito baseado em símbolos para os quais deve haver uma conceituação rigorosa.

Sem dúvida podemos dizer que o modelo descritivo também é um modelo conceitual, mas sempre que nos referirmos ao modelo conceitual da base de dados, estaremos nos referindo a este modelo conceitual formal.

Em geral, a construção do modelo conceitual de um BD se processa em duas fases: a fase dos modelos conceituais parciais e a do modelo conceitual global. Exploraremos com mais riqueza de detalhes esta modelagem a medida que prosseguirmos nosso estudo.

3. Desta fase para frente, a tarefa modelar um BD se torna bastante mais amena, uma vez que não mais lidamos com informações informais, mas sim com um modelo conceitual bem e formalmente definido.

   Nosso objetivo nesta fase é a construção do modelo interno do BD, ou seja, definir as representações internas dos dados e dos programas.

   Este modelo, em geral, permanece sempre desconhecido para os usuários do BD que ignoram (uma vez que não lhes interessa) como seus dados estão descritos internamente, isto é, como as estruturas de dados fornecidas por eles são gravadas internamente no BD.

   Esta fase também é feita em duas etapas: a construção de um modelo interno preliminar e a otimização deste. Voltaremos a abordar este assunto mais para frente neste texto.

4. Por fim, vem a fase da implementação dos programas que manipularão os dados a fim de realizar os objetivos funcionais inicialmente estabelecidos.

   No caso de realizarmos a implementação do BD fazendo uso de um SGBD, nós também desconheceremos a maneira pela qual os dados serão internamente armazenados e interagiremos com o SGBD para solicitar o acesso ao BD.

   No caso da implementação ser feita com a utilização de uma linguagem tradicional de programação, teremos que nos preocupar mais ou menos com este tipo de coisa, dependendo dos recursos que a linguagem de programação coloque à disposição.

A Construção do Modelo Descritivo

Esta fase se caracteriza por ser extremamente subjetiva e de vital importância para o sucesso do sistema de computação que se tenciona desenvolver. Diversos fatores contribuem grandemente para a dificuldade da execução desta fase, a saber:

- Dificuldades de ordem técnica (estas dificuldades de certa forma são até esperadas, tendo em vista a inexistência de métodos precisos para a realização das tarefas desta fase);
- Dificuldades de natureza política (quando os grupos de usuários possuem interesses conflitantes e não se conseguem chegar a uma solução consensual);
- Problemas de comunicação (sabemos que a linguagem que utilizamos para nos comunicarmos é uma linguagem informal e imprecisa, gerando a possibilidade de múltiplas interpretações; sabemos ainda que outra grande fonte de problemas reside na diferença de contextos e vocabulários que existe entre profissionais de informática e usuários de uma maneira geral);
- Omissões voluntárias ou involuntárias de informação (as omissões voluntárias são, vias de regra, causadas pela insegurança dos usuários. Muitos, por questão de ignorância ou desinformação, temem perder seus lugares com o advento do sistema. Por outro lado, as omissões involuntárias em geral são causadas pelo fato de que para os usuários, muitas das coisas que fazem parte das suas atividades são tão obvias que eles acham irrelevantes, acabando por omitir informações importantes);
- Ignorância e mistificação a cerca do que seja e do que seja capaz um computador (ainda nos dias de hoje, há quem pense que basta por no computador para ter tudo resolvido, há quem ainda veja o computador como uma caixinha de milagres);
- Tabu (muitos usuários vêem tudo que se relaciona com informática como sendo algo hermético, e inacessível.

  Muitos ainda acham tudo relacionado com informática extremamente complicado e nutrem uma espécie de aversão preconceituosa a tudo relacionado à computação.

  Neste contexto, não tentam entender nossos "feed backs", acham de devemos saber o que estamos fazendo, para que se intrometer no assunto?);

- Visualização global (muitas vezes, principalmente quando ainda nos falta experiência, podemos encontrar dificuldade em enxergar o problema a enfrentar como um todo, nos sentindo perdidos em meio aos inúmeros detalhes envolvidos na situação).

Já mencionamos anteriormente que a tecnologia de BD vem para aglutinar dados de aplicações tradicionalmente esparsos, redundantes, incoerentes e incompletos.

Desta maneira, todas aplicações deixariam de manter seus próprios tabelas e passariam a armazenar seus dados no BD.

Naturalmente, neste processo de aglutinação, acabaremos angariando para o BD dados que são de interesse de certas aplicações, mas não de outras.

Uma vez que cada aplicação manipula do BD a parte que lhe é devida e desconhece todo o resto dos dados que por ventura lá existam armazenados, podemos dizer que as aplicações não possuem uma visão global do BD; cada aplicação vê o BD como sendo o subconjunto dos dados do BD que utiliza para a realização de suas funções.

Assim, existirão várias visões do BD, uma para cada aplicação.

O produto desta fase deverá ser uma especificação, aplicação por aplicação, dos dados e operações envolvidas na porção do mundo real de interesse da aplicação.

A Construção do Modelo Conceitual

Poderíamos definir esta fase como sendo a construção de uma representação do conhecimento de uma organização orientada para a representação dos dados envolvidos e das relações dos dados entre si.

Tem-se por objetivo captar o mais fielmente possível a realidade da organização a fim de prover base para a implantação física de um BD.

Como dissemos anteriormente, a construção do modelo conceitual é feita normalmente em duas fases: a construção dos modelos conceituais parciais (um para cada aplicação identificada na fase anterior) e a integração deles na formação do modelo conceitual global.

Para ambas, lançaremos mão de uma técnica de modelagem conhecida pelo nome de MER.

O MER

Dentre as técnicas para a modelagem de dados mais amplamente usadas, podemos destacar o Modelo Entidade Relacionamento que, doravante, chamaremos apenas de MER.

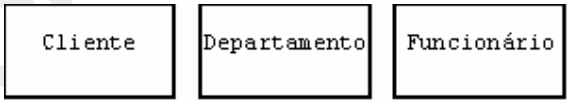
O MER, conforme foi proposto por seu autor, Peter Chen, apresentava algumas restrições e inadequações, que foram sanadas através de extensões feitas ao modelo. Em nosso estudo, abordaremos o modelo estendido, embora ainda façamos referência a ele simplesmente por MER.

Daremos início ao estudo do MER através de algumas definições:

1. Entenderemos por entidade qualquer coisa que exista no mundo real da qual se tenha necessidade de ter informações;
2. Entenderemos por conjunto de entidades um conjunto constituído por entidades de um mesmo tipo;
3. Entenderemos por relacionamento qualquer vínculo entre entidades sobre o qual se tenha necessidade de ter informações;
4. Entenderemos por conjunto de relacionamentos um conjunto constituído por relacionamentos de um mesmo tipo;
5. Entenderemos por atributo qualquer informação necessária a respeito de uma entidade ou de um relacionamento;
6. Por fim, entenderemos por restrição qualquer exigência que se faça a respeito de qualquer dos itens que definimos acima.

O MER é um modelo que expressa, cada um dos conceitos acima, através de um simbolismo gráfico.

Conjuntos de entidades são representadas por um retângulo com uma inscrição interna representando o nome da entidade. Os nomes são no singular por convenção. Considere os exemplos de entidades abaixo.



Conjuntos de relacionamentos são representados por um losango com uma inscrição interna representando o nome do conjunto de relacionamentos. Os nomes são no singular por convenção.

Como vista que todo conjunto de relacionamentos vincula sempre duas entidades, temos que sempre existem dois sentidos associados a ele, e em última instância a um conjunto de relacionamentos.

Acrescentamos, por esta razão, ao losango duas setas com sentidos opostos, representando com elas os dois sentidos dos relacionamentos do conjunto de relacionamentos. Sobre estas setas, inscrevem-se frases especificando cada um dos dois sentidos.

O exemplo abaixo simboliza um conjunto de relacionamentos que contêm relacionamentos que vinculam entidades que representam empregados no mundo real com entidades que representam departamentos de uma empresa no mundo real.

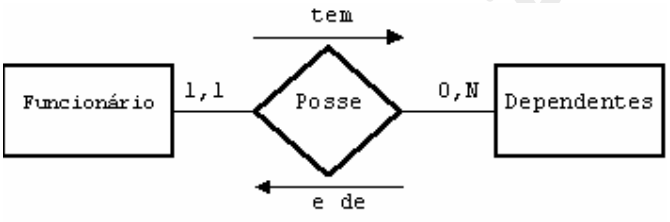
Conjuntos de relacionamentos têm sempre duas cardinalidades (uma para cada sentido dos relacionamentos) que expressam restrições com respeito àquele conjunto de relacionamentos.

Uma cardinalidade é escrita como um par de valores numéricos separados por uma vírgula. N representa um valor desconhecido, podendo ser arbitrariamente grande.

As cardinalidades de um conjunto de relacionamentos costumam ser posicionadas em lados opostos do losango, no sentido das setas.

O exemplo abaixo já contém uma carga semântica um pouco mais pesada que os anteriores. Queremos dizer com ele é que funcionários se vinculam com dependentes pelo conjunto de relacionamentos Posse. As setas nos dizem que funcionários têm dependentes e que dependentes são de funcionários.

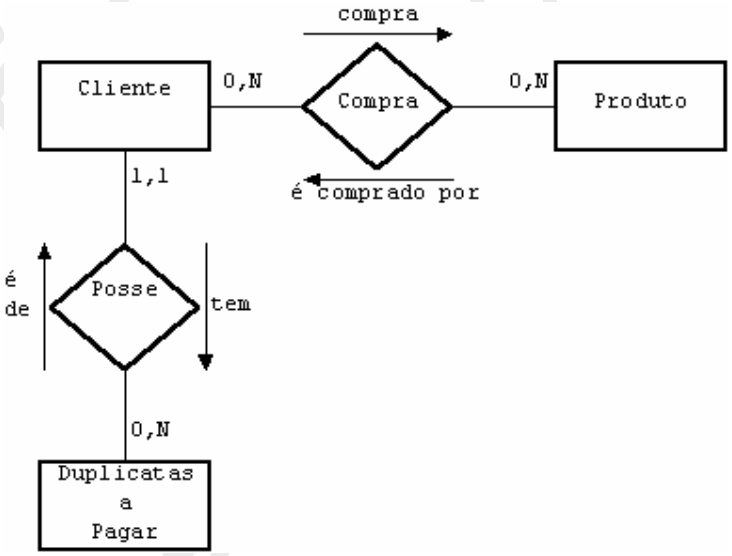
Lemos o modelo abaixo da seguinte maneira: um funcionário tem de 0 até N dependentes, e um dependente é de 1 e sempre um funcionário.



Como no mundo real cada conjunto de entidades pode associar-se a outros conjuntos de entudades, também, no modelo, poderemos expressar este fato.

Considere o exemplo abaixo. Nele expressamos que clientes se relacionam pelo conjunto de relacionamentos Compra com Produtos e também com duplicatas a pagar pelo conjunto de relacionamentos Posse.

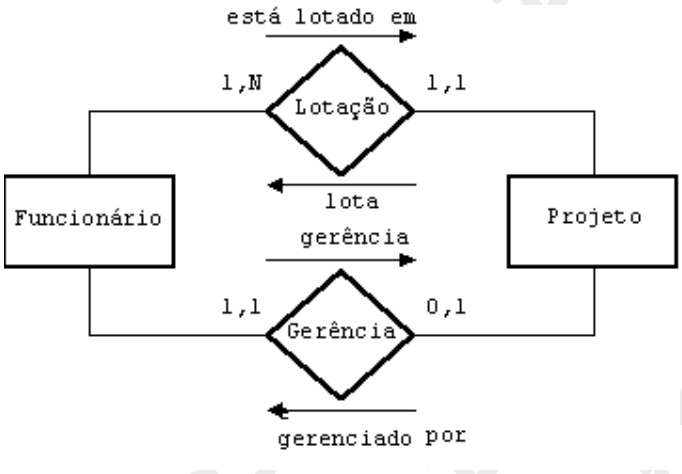
Cada Cliente compra de 0 até N Produtos e cada Produto é comprado por de 0 até N Clientes. Cada Cliente tem de 0 até N Duplicatas a pagar e cada Duplicata a pagar e de 1 e sempre 1 Cliente.



Também à semelhança do mundo real, dois conjuntos de entidades podem estar associados através de mais de um conjunto de relacionamentos.

Veja o exemplo abaixo. Nele, Funcionários se relacionam com Projetos através dos conjuntos de relacionamentos Lotação e Gerência. Cada Funcionário está lotado em 1 e sempre 1 Projeto

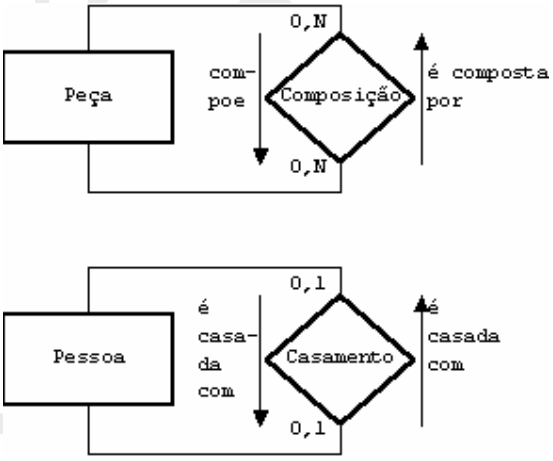
e cada Projeto lota de 1 até N Funcionários. Além disto, cada Funcionário pode gerenciar ou não um Projeto e cada Projeto é gerenciado por 1 e sempre 1 Funcionário.



Conjuntos de auto relacionamentos expressam relacionamentos entre entidades de um mesmo conjunto de entidades. No exemplo abaixo temos dois casos de conjunto de auto relacionamentos.

No primeiro, peças se relacionam com peças pelo conjunto de auto relacionamentos Composição. Cada peça compõe de 0 até N peças e, por outro lado, cada peça pode compor de 0 até N peças.

No segundo, pessoas se relacionam com pessoas pelo relacionamento casamento. Cada pessoa pode ser casada ou não com outra pessoa.

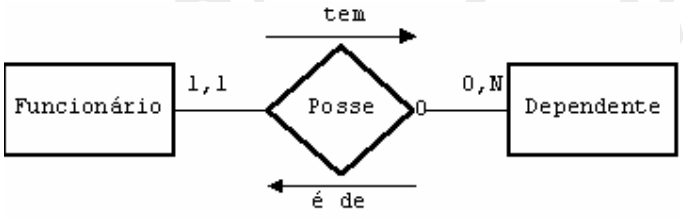


Conjuntos de entidades podem ser fracos com relação a outros. Isto significa que não fazem sentido se o conjunto de entidades forte.

Esta relação de fraqueza é expressa no modelo como um conjunto de relacionamentos com um pequeno círculo de um dos lados. O conjunto de entidades do lado do círculo é o conjunto de entidades fraco.

Conjuntos de entidades fracos podem ter, como parte de sua chave primária, a chave primária da entidade forte.

No exemplo abaixo, mostramos que o conjunto de entidades Dependente, associado através do conjunto de relacionamentos Posse com o conjunto de entidades Funcionário, é fraco, isto é, não tem sentido sem o conjunto de entidades Funcionário.

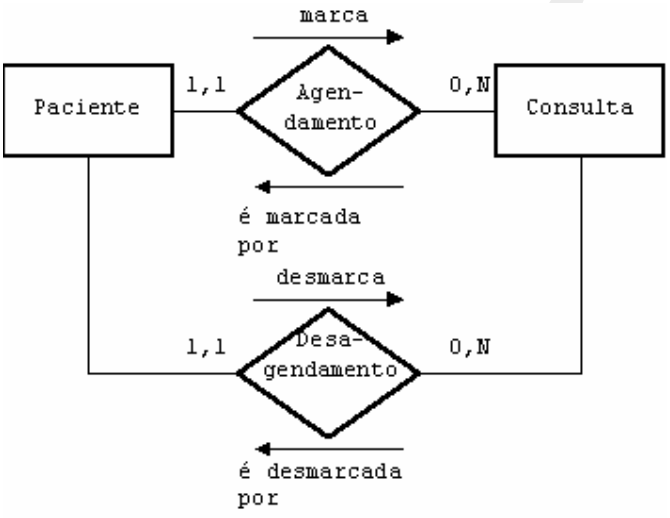


O MER é um modelo para expressar dados e seus vínculos que se tem interesse de manter. Por esta razão, não expressamos em um MER nem operações (uma vez que não são dados nem vínculos entre dados) nem dados sobre os quais não se tenha interesse.

Assim, no exemplo abaixo, mostramos o conjunto de entidades Paciente relacionada com a entidade consulta pelos conjuntos de relacionamentos Agendamento e Desagendamento.

A menos que se tenha uma boa razão para se guardar desagendamentos, desagendamento é uma operação que se faz sobre o conjunto de relacionamentos agendamento.

Quando desejamos agendar uma consulta, geramos um relacionamento no conjunto de relacionamentos Agendamento, quando desejamos desagendar uma consulta, simplesmente retiramos um relacionamento do conjunto de relacionamentos Agendamento, não precisamos de um conjunto de relacionamentos Desagendamento.



Atributos representam as informações que se deseja manter a respeito de um conjunto de entidades ou relacionamentos.

Segundo suas funções, os atributos se classificam em chave primária, chave adicional e atributo comum (chaves estrangeiras não são permitidas neste modelo).

Conforme veremos adiante, existe ainda um outro tipo de atributo, mas deixaremos para comentar a seu respeito mais para frente.

Chave primária é o menor conjunto de atributos que identifica unicamente uma única entidade em um conjunto de entidades ou um único relacionamento em um conjunto de relacionamentos.

Todo conjunto de entidades e todo conjunto de relacionamentos possuem chave primária, embora ela, em geral, não seja sempre explicitada.

Às vezes, esta combinação de chaves não é o suficiente para identificar unicamente um relacionamento em um conjunto de relacionamentos (pela possibilidade da existência de mais de um relacionamento entre os mesmos duas entidades dos mesmos dois conjuntos de entidades em um conjunto de relacionamentos).

Neste caso, existem duas opções: colocar uma chave adicional no conjunto de relacionamentos (que em conjunto com a chave primária implícita identificaria unicamente um relacionamento) ou indicar, explicitamente, uma chave primária para o conjunto de relacionamentos.

Por fim, atributos comuns são atributos sem nenhuma função especial, representam tão somente informações que se deseje manter sobre entidades relacionamentos.

Vale a pena ressaltar o fato de que não se pode expressar com o MER atributos iterativos (repetitivos), por esta razão não se pode ter atributos com nome no plural.

Aconselha-se que em paralelo à construção do modelo, vá se construindo também um *dicionário de dados* contendo informações sobre os dados do BD (nome, tipo, tamanho, sistemas em que está envolvido e em que operações, etc.).

Qualquer que seja o tipo do atributo, eles sempre são representados por uma haste com um símbolo em uma das extremidades que indica o tipo do atributo seguido do nome do atributo. Veja abaixo o símbolo de cada um dos atributos acima mencionados.

———*[i] Nome
———+[i] Nome
———O Nome

O símbolo de chave primária é um asterisco seguido opcionalmente por um número. Este número serve para numerar os componentes da chave primária em caso de chave composta.

O símbolo de chave adicional é um sinal de mais seguido opcionalmente por um número. Este número serve para os mesmos fins do número que segue opcionalmente o símbolo de chave primária.

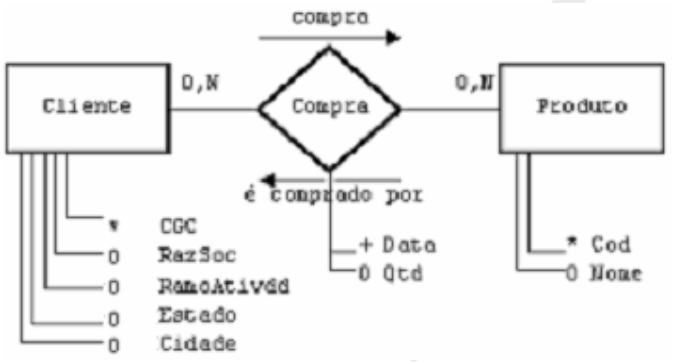
O símbolo de atributo comum é a letra O.

Veja no exemplo abaixo atributos apropriadamente dispostos em conjuntos de entidades e de relacionamentos.

O que se pretende expressar é que o conjunto de entidades Cliente tem com chave primária o atributo CGC e, como atributos comuns, RazSoc, RamAtivdd, Cidade e Estado.

O conjunto de entidades Produto tem como chave primária o atributo Cod e, como atributo comum, Nome.

Por fim, o conjunto de relacionamentos Compra tem como chave adicional o atributo Data e, como atributo comum, Qtd.



Às vezes, conjuntos de entidades se classificam em tipos, cada qual tendo atributos próprios e diferenciados. Surge para isto o conceito de hierarquia.

Existem dois tipos de hierarquias conforme se pode observar no exemplo abaixo.

O primeiro desenho representa uma hierarquia sem interseção e o segundo uma com interseção.

Qualquer que seja o tipo de uma hierarquia, o conjunto de entidades superior detém os atributos comuns a todos os membros da hierarquia e, os inferiores, os atributos próprios a cada uma delas.

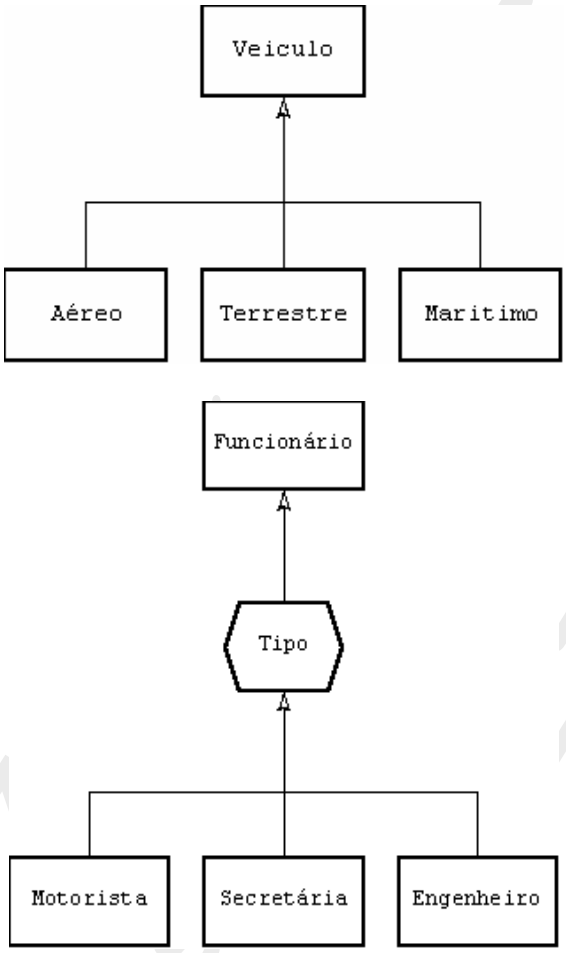
É perfeitamente possível se ter árvores de hierarquia mais profundas, isto é, hierarquias sob hierarquias. Também é possível se ter mais de uma hierarquia sob um mesmo conjunto de entidades.

Conjuntos de entidades que participam de uma hierarquia são conjuntos de entidades como outros quaisquer, portanto podem ter seus próprios atributos e podem se associar a outros conjuntos de entidades através de conjuntos de relacionamentos.

Não faz sentido existir uma hierarquia em que não haja atributos nos conjuntos de entidades inferiores.

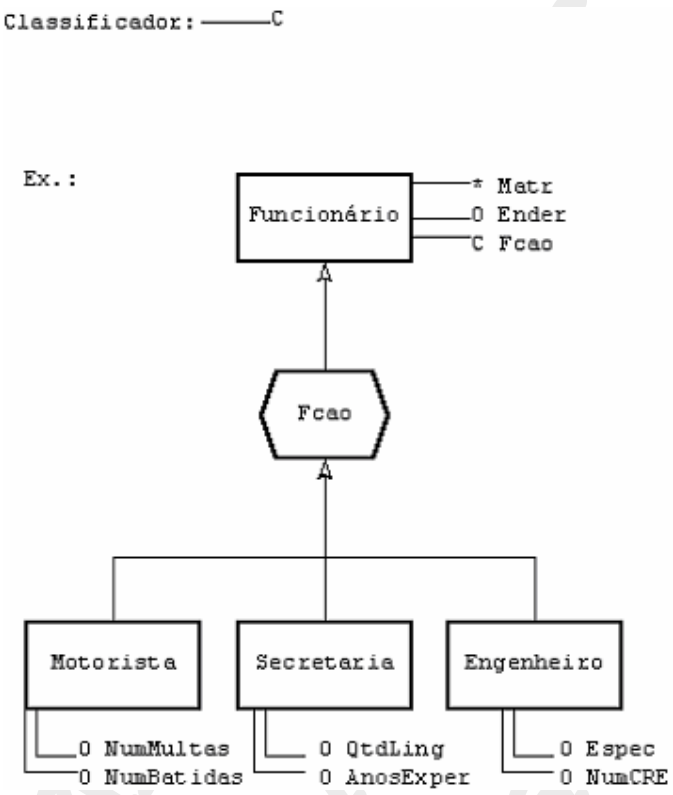
No primeiro desenho, vemos que veículos se classificam em aéreo, terrestre e marítimo, podendo um veículo ser de um, dois ou dos três tipos concomitantemente.

No segundo desenho, vemos que funcionários se classificam em motorista, secretária e engenheiro, não podendo alguém que ocupa uma função ocupar também outra.



Surge com as hierarquias sem interseção aquele último tipo de atributo que havíamos mencionado acima. Trata-se do atributo classificador, que serve para dizer de que tipo (onde se encontra nos conjuntos de entidades inferiores) é cada entidade do conjunto de entidades superior.

Este tipo de atributo é representado por uma haste seguida por uma letra C e pelo nome do atributo. Note no exemplo abaixo, que além da novidade do atributo classificador, não se usa colocar chave primária nos conjuntos de entidades inferiores de uma hierarquia uma vez que estas têm como chave primária a chave primária do conjunto de entidades superior.

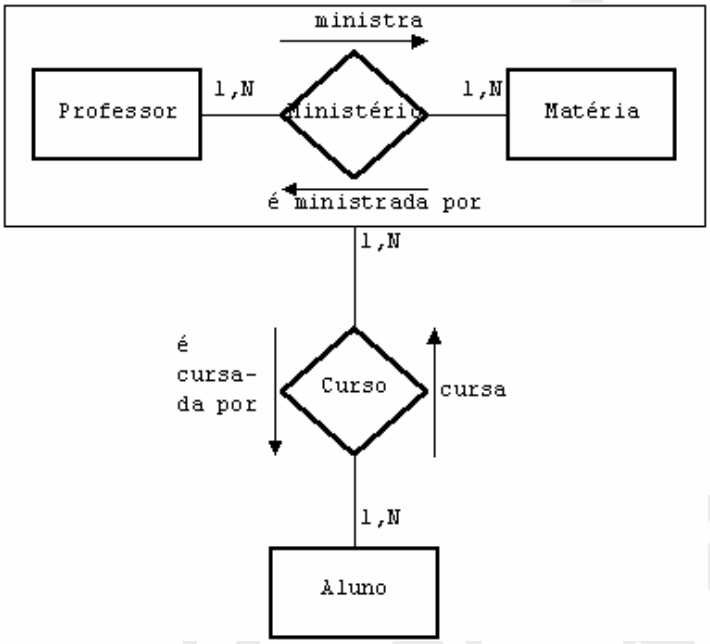


O último conceito que introduziremos é o conceito de agregação.

Usa-se agregação quando se tem dois conjuntos de entidades associados através de um conjunto de relacionamentos, e se deseja relacionar, através de outro conjunto de relacionamentos, uma terceira entidade não com a primeira entidade, nem com a segunda, mas com o conjunto de relacionamentos que existe entre eles.

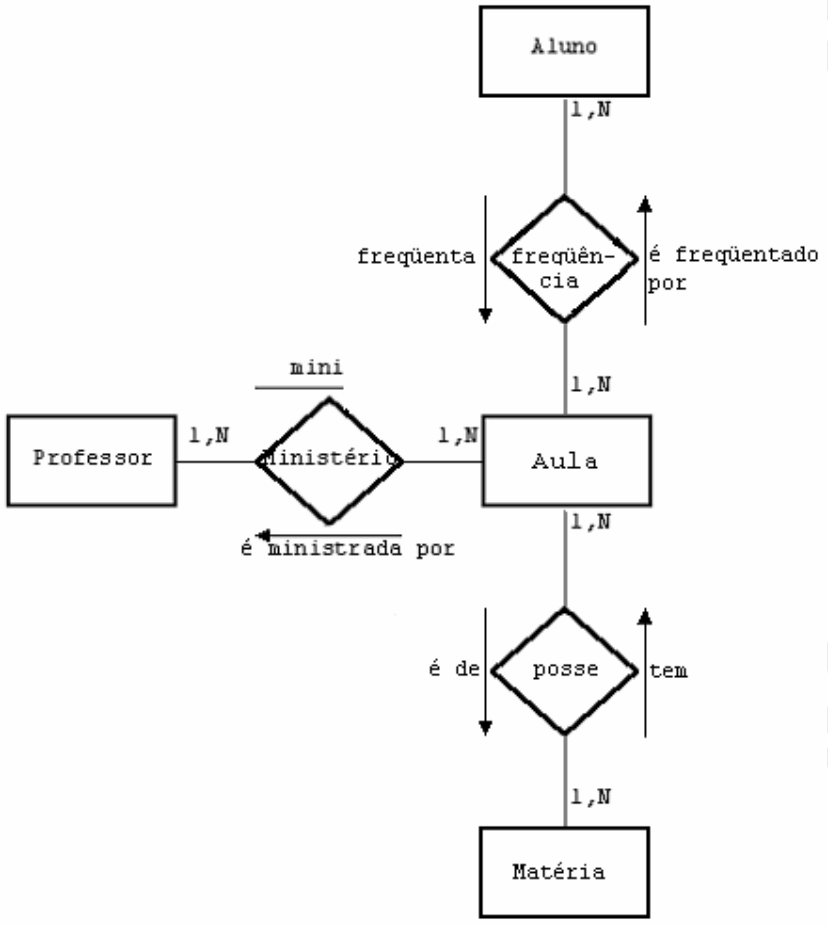
Agregados têm como chave primária a chave do seu conjunto de relacionamentos interno.

No exemplo abaixo, temos o conjunto de entidades Aluno relacionado não com o conjunto de entidades Professor, não com o conjunto de entidades Matéria, mas sim com o conjunto de relacionamentos Ministério.



Um agregado pode ser evitado se introduzirmos, no lugar do conjunto de relacionamentos que define o agregado, um conjunto de entidades (que chamamos de conjunto associativo de entidades) associado, não apenas aos conjuntos de entidades que definiam o agregado, mas também ao conjunto de entidades que se relaciona com o agregado.

No exemplo abaixo, temos o conjunto de entidades Aula relacionado (1) pelo conjunto de relacionamentos Freqüência, com o conjunto de entidades Aluno; (2) pelo conjunto de relacionamentos Ministério, com o conjunto de entidades Professor; e (3) pelo conjunto de relacionamentos Posse, com o conjunto de entidades Matéria.



Segue agora um exemplo mais complexo que envolve muitos dos conceitos que estudamos até o momento. Considere a situação descrita abaixo e observe seu MER logo em seguida.

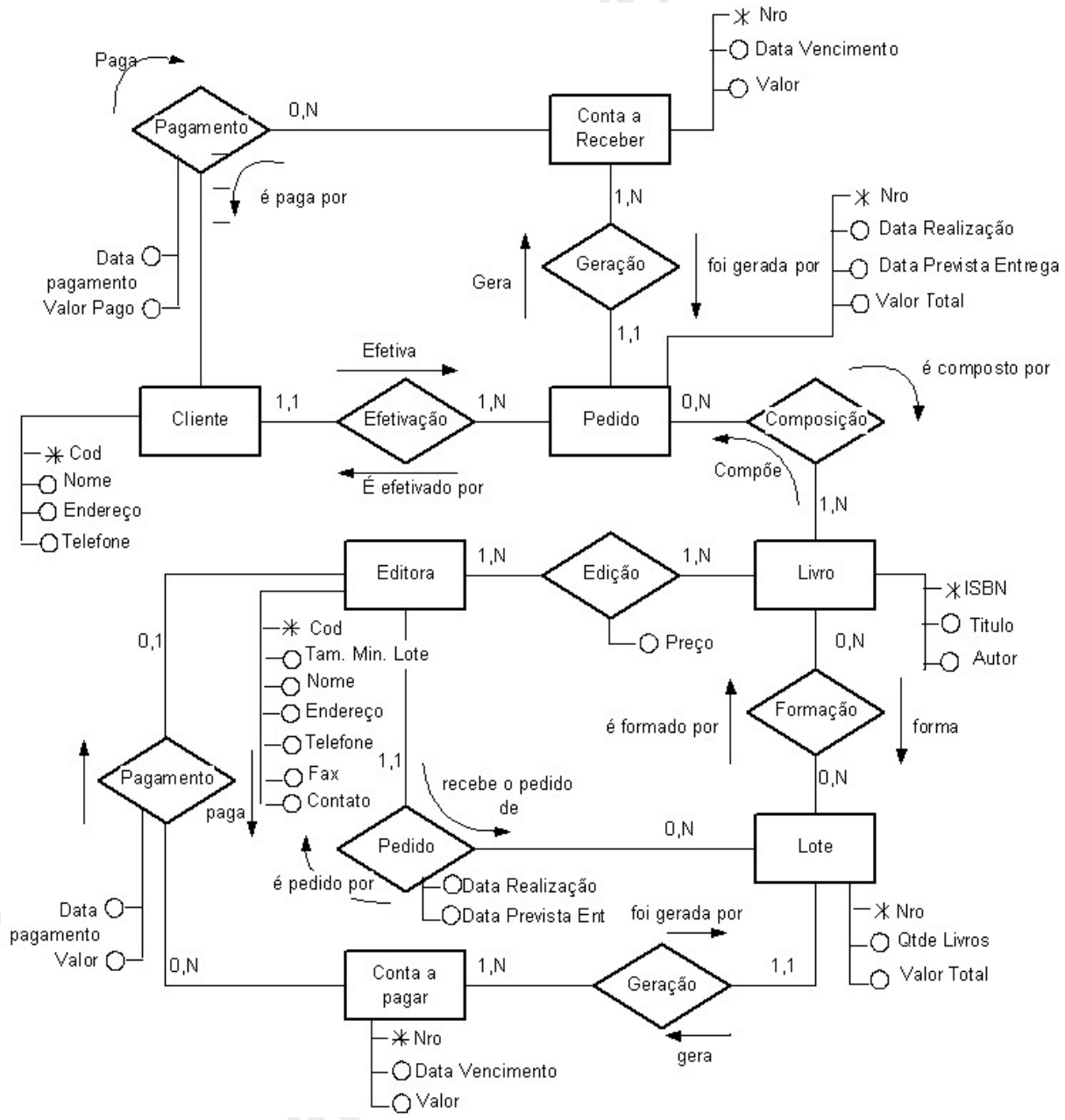
Uma companhia de comércio de livros trabalha recebendo pedidos de livros dos clientes, encomendando-os para editoras e remetendo-os aos clientes assim que disponíveis.

Os pedidos de compra a editoras são feitos em lotes para que a companhia possa desfrutar de descontos por fazer encomendas maiores.

As remessas aos clientes são feitas assim que o pedido estiver completamente atendido.

Quando da entrega, o cliente recebe também um aviso de cobrança para que possa efetuar o pagamento (o cliente paga somente após ter recebido os livros encomendados).

As editoras, por sua vez, de posse dos pedidos, enviam à companhia juntamente com os livros, uma guia de remessa (para possibilitar a companhia conferir os produtos enviados com os realmente recebidos) e uma fatura que deverá ser liquidada pela companhia.



A Construção dos Modelos Conceituais Parciais

Esta parte do modelo conceitual tem duas componentes: uma representação gráfica e uma linguagem descritiva.

Representaremos graficamente os dados e suas relações e utilizaremos a linguagem descritiva a fim de descrever as operações que são realizadas sobre os dados. Desta maneira, teremos expressado no modelo, tanto os aspectos estáticos (dados), como dinâmicos (operações).

Deve-se ainda expressar neste modelo restrições semânticas e de integridade.

Para representar graficamente os dados, suas relações, e algumas restrições semânticas e de integridade, sugere-se que se use o MER.

Para descrever as operações que são realizadas sobre os dados e outras restrições semânticas e de integridade que não puderam ser representadas graficamente através do MER, sugere-se que se use algo como português estruturado.

É imprescindível atentar para o fato de que, durante a construção dos MER, deve-se evitar ao máximo pensar em processos, tabelas e fluxos de informação. Em um modelo de dados, só devem aparecer os dados que são usados na aplicação modelada, e da maneira como são usados.

Assim, é perfeitamente possível haver um conjunto de entidades ou de relacionamentos em um modelo, apresentando um conjunto de atributos diferentes daquele que apresenta em um outro modelo de uma outra aplicação.

A chave primária de um conjunto de entidades ou de relacionamentos deve ser sempre a mesma, em todos os modelos parciais onde aparecerem.

A cardinalidade de um conjunto de relacionamentos também deve ser sempre a mesma, em todas os modelos onde o mesmo aparecer.

É possível que um conjunto de entidades apareça (1) como parte de uma hierarquia; (2) mantendo determinados conjuntos de relacionamentos; ou (3) como parte de um agregado em um modelo e em outro modelo não.

A Construção do Modelo Conceitual Global

O modelo conceitual global é construído a partir dos modelos conceituais parciais de uma maneira incremental.

Basicamente, tudo que deve ser feito é transportar, para um o papel em branco onde será construído o modelo conceitual global, todos os modelos conceituais parciais, sempre tendo o cuidado de fazer com que elementos do modelo parcial (conjuntos de entidades e de relacionamentos, atributos, agregados e hierarquias) que estão sendo transportados se sobreponham, de forma incremental, às suas versões já presentes no modelo global.

A Construção do Modelo Lógico

Como já havíamos comentado anteriormente, esta fase se compõe de duas subfases (a construção do modelo lógico preliminar e a otimização deste) que passaremos a detalhar em seguida.

A Construção do Modelo Lógico Preliminar

Podemos pensar que o objetivo desta fase é trabalhar o modelo conceitual global a fim de eliminar, ou melhor, transformar em conjuntos de entidades, qualquer coisa presente no modelo conceitual global que não seja um conjunto de entidades. O modelo conceitual assim trabalhado será o modelo interno preliminar.

Como perceberemos a seguir, os casos que teremos que analisar são, virtualmente, infinitos e, desta maneira, nosso objetivo nesta parte do estudo não é examinar cada caso, mas sim desenvolver uma linha de raciocínio que possa ser utilizada quando em face de uma situação real nova.

Por simplicidade, inicialmente deixaremos de lado qualquer tipo de MER diferente de dois conjuntos de entidades relacionados por um conjunto de relacionamentos.

Qualquer que seja a cardinalidade do conjunto de relacionamentos, sempre é possível reduzir o MER a três conjuntos de entidades. Em alguns casos é possível a redução a duas e até mesmo a uma.

Seremos sempre ambiciosos, desejando a máxima redução possível. Consideraremos a seguir algumas possibilidades para a cardinalidade do conjunto de relacionamentos.

- **Caso 1,1 - 1,1**

Nesta situação, cada entidade de A se relaciona com exatamente uma entidade de B (uma entidade de A possui exatamente um relacionamento no conjunto de relacionamentos R) e vice versa.

Desta maneira é possível a redução do MER a um único conjunto de entidades ARB que aglutinaria os atributos de A, de R e de B, naturalmente eliminando os possíveis atributos comuns.

- **Caso 1,1 - 1,N**

Neste caso, cada entidade de A pode estar relacionada com de 1a N entidades de B (uma entidade de A pode possuir de 1 a N relacionamentos no conjunto de relacionamentos R). Cada entidade de B, por sua vez, sempre se relaciona com exatamente uma entidade de A (uma entidade de B pode possuir um e somente um relacionamento no conjunto de relacionamentos R).

Como não somos capazes de precisar a quantidade de relacionamentos que uma entidade de A poderá vir a manter com uma entidade de B, não poderemos aglutinar o MER em uma única entidade onde cada entidade representasse uma entidade de A, todos seus relacionamentos com entidades de B, e as próprias entidades de B relacionadas.

Para alcançarmos a solução mais ambiciosa (uma entidade ARB), existe ainda uma possibilidade: aglutinar o MER em um único conjunto de entidades onde, cada entidade representasse uma entidade de A, um dos relacionamentos que esta entidade mantém com uma entidade de B, e a própria entidade de B em questão.

Neste caso, como as entidades de A podem ter muitos relacionamentos com entidades de B, teremos muitas entidades, no conjunto de entidades aglutinado, associadas a uma única entidade de A.

Isto, naturalmente, caracteriza redundância, mas, se soubermos de antemão que, na prática, a probabilidade de entidades de A terem muitos relacionamentos com entidades de B é baixa, poderemos optar por esta solução.

Existem ainda duas possibilidades mais modestas (formar dois conjuntos de entidades): aglutinar A com R em um conjunto de entidades AR deixando B separado; e aglutinar R com B em uma entidade RB, deixando A separado.

A primeira possibilidade está sujeita às mesmas considerações que tecemos sobre a aglutinação em uma única entidade ARB.

A segunda possibilidade será eleita sempre que não dispusermos das informações estatísticas necessárias a uma aglutinação ou quando estas estatísticas não forem favoráveis.

- **Caso 1,N - 1,N**

Neste caso, cada entidade de A pode estar relacionada com de 1a N entidades de B (podendo uma entidade de A possuir de 1 a N relacionamentos no conjunto de relacionamentos R) e vice versa.

Podemos raciocinar de maneira semelhante à que utilizamos no caso anterior.

Para aglutinarmos o MER em uma única entidade ARB, seria preciso garantir que a probabilidade das entidades de A manterem muitos relacionamentos com entidades de B é baixa e também que a probabilidade das entidades de B manterem muitos relacionamentos como entidades de A é baixa.

Para aglutinarmos o MER em duas entidade AR e B, seria preciso garantir que a probabilidade das entidades de A manterem muitos relacionamentos com entidades de B é baixa.

Para aglutinarmos o MER em duas entidade A e RB, seria preciso garantir que a probabilidade das entidades de B manterem muitos relacionamentos com entidades de A é baixa.

Optaremos por deixar três conjuntos de entidades separados, A, R, e B, quando não dispusermos das informações estatísticas necessárias a uma aglutinação ou quando estas estatísticas não forem favoráveis.

Vale a pena lembrar, que a maneira de julgar se uma probabilidade é baixa ou não, varia de caso para caso (depende fundamentalmente do número de entidades e relacionamentos que se espera ter nos conjuntos de entidades e de relacionamento respectivamente).

O raciocínio para o caso dos conjuntos de auto-relacionamentos é idêntico ao que tivemos quando tratamos dos conjuntos de relacionamentos comuns.

Ampliemos agora nosso universo, deixando de lado somente os agregados e as hierarquias e passando a trabalhar com MER com diversos conjuntos de entidades e de relacionamentos.

O processo é simples: tomamos sempre um conjunto de relacionamentos (e os conjuntos de entidades por ele relacionados, talvez um só, no caso de conjuntos de auto-relacionamentos) serem analisados conforme raciocínio que tecemos acima.

Se algum dos conjuntos de entidades primitivo mantinha algum conjunto de relacionamentos com outro conjunto de entidades não considerada na operação, este conjunto de relacionamentos será transferido para o conjunto de entidades que aglutinou o conjunto de entidades primitivo (no caso de ter havido aglutinação).

Repete-se este processo, até que não se existirem mais conjuntos de relacionamentos no MER. Sempre serão considerados primeiro neste processo os conjuntos de relacionamentos com cardinalidade mais baixa.

O caso das hierarquias é o mais simples de todos, em geral o processo se resume em eliminar a hierarquia, deixando apenas os conjuntos de entidades que faziam parte da hierarquia, todos identificados pela chave primária da entidade genérica da hierarquia.

Existem, no entanto, algumas outras possibilidades que passaremos a considerar.

No caso das hierarquias sem interseção cujo conjunto genérico de entidades não mantenha conjuntos de relacionamentos, poderemos fazer uma espécie de implosão, trazendo as entidades apropriadas da entidade genérica para cada entidade especializada.

No caso das hierarquias com interseção cujo conjunto genérico de entidades não mantenha conjuntos de relacionamento, poderemos fazer a mesma implosão acima considerada, mas, neste caso, haverá redundância e, por isso, teremos que considerar probabilisticamente a viabilidade de mantê-la.

No caso dos agregados, procedemos com segue: (1) tratamos o conjunto de relacionamentos e a(s) entidade(s) interna(s) que definem o agregado; (2) eliminamos o agregado, deixando somente os conjuntos de entidades possivelmente aglutinados que estavam em seu interior; e (3) transferimos todos os conjuntos de relacionamentos que ser relacionavam com o agregado para o conjunto de entidades no qual foi aglutinado o conjunto de relacionamentos que definia o antigo agregado.

Antes de qualquer coisa, sempre se trata primeiro os agregados e hierarquias, para então, e só então, tratar o restante do MER.

Tendo transformado em conjuntos de entidades tudo quanto não era um conjunto de entidades no modelo conceitual global, poderemos considerar todas as entidades a que chegamos como tabelas preliminares.

A Construção do Modelo Lógico Otimizado

Quando falamos em otimização do modelo lógico, falamos em partições horizontais ou verticais de tabelas.

Entendemos por partição horizontal de uma tabela, a separação de grupos de linhas desta tabela em tabelas separadas. Entendemos por partição vertical de uma tabela, a separação de colunas desta tabela em tabelas separadas.

É importante notar, que se pode ter interseções, isto é, em uma partição horizontal podemos levar certos grupos de linhas para mais de uma tabela e, em uma partição vertical, podemos levar certos grupos de colunas para mais de uma tabela.

Vale a pena ressaltar, que uma tabela pode ser partida, tanto horizontal, quanto verticalmente, em qualquer ordem. As tabelas resultantes de uma partição também podem ser partidas em mais tabelas.

Partições têm por objetivo isolar em tabelas separadas e menores, informações que possuem uma grande taxa de acesso, aumentando assim a eficiência do acesso ao BD.

A Implementação dos Programas Aplicativos

Agora que temos as tabelas otimizadas resultantes da modelagem, poderemos retornar aos modelos conceituais parciais para a implementação dos algoritmos em português estruturado associados a cada operação de cada aplicação.

Para tanto, escolheremos uma linguagem de programação adequada e desenvolveremos as aplicações identificadas na modelagem. Isto poderá ser feito sob um SGBD ou não.

Apêndice A:
Exercícios de Normalização

Considere as tabelas abaixo e produza tabelas equivalentes que estejam na 1FN, na 2FN e na 3FN. Explique as transformações realizadas e as razões pelas quais decidiu realiza-las:

1. Alunos {**RAalu**, NomAlu, FoneAlu, DiscsAlu {CodDis, NomDis, QtdHs}}

   Trata-se de uma tabela de alunos. Para cada aluno, guarda-se RAalu (o RA do aluno), NomAlu (o nome do aluno), FoneAlu (o telefone do aluno) e DiscsAlu (as disciplinas nas quais o aluno está matriculado).

   Para cada disciplina na qual um aluno está matriculado, guarda-se CodDis (o código da disciplina), NomDis (o nome da disciplina) e QtdHs (a quantidade de horas semanais que o aluno dedica à disciplina).

2. Médicos {**NroMed**, NomMed, PacsMed {NroPac, NomPac, TpoPac}}

   Trata-se de uma tabela de médicos. Para cada médico, guarda-se NroMed (o número do médico), NomMed (o nome do médico) e PacsMed (os pacientes do médico).

   Para cada paciente de um médico, guarda-se NroPac (o número do paciente), NomPac (o nome do paciente) e TpoPac (a quantidade de tempo expresso em meses que o médico tem aquele paciente).

3. Laboratorios {**NroLab**, NomLab, ExmsLab {CodExm, NomExm, CustoExm}}

   Trata-se de uma tabela de laboratórios de análises clínicas. Para cada laboratório, guarda-se NroLab (o número do laboratório), NomLab (o nome do laboratório) e ExmsLab (os exames que o laboratório é capaz de realizar).

   Para cada exame de um laboratório, guarda-se CodExm (o código do exame), NomExm (o nome do exame) e CustoExm (quanto o exame custa para aquele laboratório).

Apêndice B:
Exercícios de Modelagem de Dados

1. A TELEFONICA está querendo automatizar suas operações, e após algumas conversas com esta, conseguimos a seguinte descrição.

   Existem duas maneiras para se adquirir um telefone: através de um particular e através da própria TELEFONICA.

   Ao adquirir um telefone de um particular, deve-se preencher um formulário apropriado que é conseguido na própria TELEFONICA onde constam os dados pessoais de ambas as partes e também suas assinaturas. Entregue este formulário na TELEFONICA, é feita então a transferência de titularidade do proprietário antigo para o novo proprietário.

   Para se adquirir um telefone da própria TELEFONICA, deve-se dirigir a uma de suas lojas e preencher um formulário de inscrição com os dados pessoais do pretendente. Verifica-se então, com base no endereço do pretendente, mais especificamente com base no bairro em que resido o mesmo, os prefixos das linhas que poderiam atender a sua residência. No caso de haver alguma linha da TELEFONICA com o prefixo apropriado e livre, esta é vendida ao interessado; caso contrário, a TELEFONICA lhe vende um plano de expansão. Em qualquer caso, o pagamento poderá ser feito de diversas maneiras, de acordo com os planos de pagamento vigentes (estes planos são mutáveis com o tempo, naturalmente somente para compradores novos).

   Quando a TELEFONICA instala novas centrais na rede telefônica (e portanto passa a dispor de novas linhas), contata-se os proprietários de planos de expansão da linha (prefixo)correspondente à central instalada e faz-se então a instalação da linha se o proprietário ainda morar em um local adequado a linha e cancela-se o plano de expansão correspondente, transferindo-se a titularidade da nova linha da TELEFONICA para o novo assinante.

2. A indústria de autopeças AUTOPARTS fornece seus produtos a três classes básicas de clientes: Indústrias Automobilísticas, Revendedores Autorizados e Público em Geral. A cada reajuste de preços a AUTOPARTS emite listas de preços, que contém, para cada produto, o preço que será cobrado pela AUTOPARTS de seus clientes, a partir da data de vigência da lista. Note que os preços são diferentes para cada classe de cliente. Uma lista de preços é válida desde sua data de vigência, até que outra lista com outra data de vigência seja emitida.

   As listas de preços são base para o Sistema de Faturamento, que, a partir da identidade do cliente e do produto, aplica o preço correspondente e o desconto instituído para obter o preço de venda.

   O desconto na AUTOPARTS é algo bastante flexível: ele pode ser dado para um produto, para uma categoria de produtos, para um cliente, para uma classe de clientes ou para qualquer combinação destas coisas. No entanto, o faturamento só considerará o desconto se ele tiver sido cadastrado previamente.

   O Departamento de Administração de Vendas possui um órgão coordenador centralizado na SEDE e órgãos operacionais distribuídos pelas FILIAIS.

   É atribuição da coordenação centralizada na SEDE operar a política de reajustes de preços que podem ser motivados por:

   - Reajuste nos formadores de preços dos produtos (matéria prima, salários, custos operacionais, etc);
   - Índices de reajustes autorizados pelo governo;
   - Preços praticados pela concorrência.

   Assim, a política de preços da AUTOPARTS varia entre "estabelecer o melhor preço para a AUTOPARTS" e "praticar aquilo que o governo e a concorrência permitem", dependendo do momento.

   Para operar essa "política de preços", a coordenação precisa de apoio informatizado para:

Estabelecer índices de reajustes variados, de acordo com as várias situações (por produtos, por classes de produto, por cliente, por classes de cliente, etc);

Simular as listas de preços resultantes e seu impacto no faturamento futuro;

Optar por uma determinada situação de simulação e efetivá-la;

Estabelecer índices de desconto de acordo com as várias possibilidades;

Simular o impacto desses descontos no faturamento, de acordo com uma determinada lista de preços e uma determinada projeção de vendas;

Optar por uma determinada simulação de descontos e efetivá-la;

Manter histórico de listas de preços praticadas e de índices de desconto praticados.

Os órgãos operacionais distribuídos pelas FILIAIS têm por objetivo controlar a aplicação da política de preços da AUTOPARTS.

O sistema de faturamento tem a opção de atribuir o preço e o desconto automaticamente ou, por comando do operador do faturamento, permitir a entrada manual de preço ou de um desconto quando a combinação preço/desconto não satisfizer a condição de venda. É necessário acompanhar o preço efetivamente praticado para, com isso, subsidiar a geração das novas tabelas de preços e de descontos, de forma a minimizar o preço dado manualmente no futuro.