PROJETO DE BANCO DE DADOS

Uma Visão Prática

Felipe Machado
Mauricio Abreu

11ª EDIÇÃO

Projeto: Conceitual, Lógico e Físico

Construção de Banco de Dados Físico
(Microsoft® SQL Server™ 7.0 e Oracle® 8.xx)

Consultas e Comandos em
SQL ANSI e Microsoft® SQL Server™ 7.0

ÉRICA

Felipe Nery Rodrigues Machado
Maurício Pereira de Abreu

Projeto de Banco de Dados
Uma Visão Prática

Editora Érica Ltda.
2004 - 11ª Edição

658.4
M 149 p
11. ed
ex. 2

Conselho Editorial:

Diretor Editorial:	Antonio Marco Vicari Cipelli
Diretor Comercial:	Paulo Roberto Alves
Diretor de Publicidade:	Waldir João Sandrini
Finalização de Capa:	Maurício S. de França
Editoração:	Rosana Ap. A. dos Santos
Desenhos:	Pedro Paulo Vieira Herruzo
Revisão Gramatical:	Marlene Teresa Santin Alves
Coordenação:	Rosana Arruda da Silva

Copyright © 1996 da Editora Érica Ltda.

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Machado, Felipe Nery Rodrigues
Projeto de Banco de Dados: uma visão prática / Felipe Nery Rodrigues Machado, Mauricio Pereira de Abreu. -- São Paulo : Érica, 1996.

Bibliografia.
ISBN 85.7194.312-5

1. Banco de dados: I. Abreu, Maurício Pereira de. - II. Título

96.0014 CDD-005.75

Índices para catálogo sistemático

1. Banco de Dados: Sistemas: Processamento de dados 005.75

Todos os direitos reservados. Proibida a reprodução total ou parcial, por qualquer meio ou processo, especialmente por sistemas gráficos, microfílmicos, fotográficos, reprográficos, fonográficos, videográficos, internet, e-books. Vedada a memorização e/ou recuperação total ou parcial em qualquer sistema de processamento de dados e a inclusão de qualquer parte da obra em qualquer programa juscibernético. Essas proibições aplicam-se também às características gráficas da obra e à sua editoração. A violação dos direitos autorais é punível como crime (art. 184 e parágrafos, do Código Penal, cf. Lei nº 6.895, de 17.12.80) com pena de prisão e multa, conjuntamente com busca e apreensão e indenizações diversas (artigos 102, 103 parágrafo único, 104, 105, 106 e 107 itens 1, 2 e 3 da Lei nº 9.610, de 19/06/98, Lei dos Direitos Autorais).

Os Autores e a Editora acreditam que todas as informações aqui apresentadas estão corretas e podem ser utilizadas para qualquer fim legal. Entretanto, não existe qualquer garantia, explícita ou implícita, de que o uso de tais informações conduzirá sempre ao resultado desejado. Os nomes de sites e empresas, porventura mencionados, foram utilizados apenas para ilustrar os exemplos, não tendo vínculo nenhum com o livro, não garantindo a sua existência nem divulgação. Eventuais erratas estarão disponíveis no site da Editora Érica para download.

"Algumas imagens utilizadas neste livro foram obtidas a partir do CorelDRAW 7, 8 e 9 e da Coleção do MasterClips/MasterPhotos© da IMSI, 1985 Francisco Blvd. East, San Rafael, CA 94901-5506, USA."

Editora Érica Ltda.
Rua São Gil, 159 - Tatuapé
CEP: 03401-030 - São Paulo - SP
Fone: (11) 295-3066 - Fax: (11) 6197-4060
www.editoraerica.com.br

96504
Sociedade de Ensino Superior Estácio de Sá
Campus [illegible]
N.º 25759 105 EX.: 2
Em: 20.05.2005 Aquisição C

Prefácio

Desde a Antiguidade, o homem tem procurado transmitir e documentar seu conhecimento, objetos e fatos da vida real. Nas cavernas pré-históricas, foram encontrados desenhos de animais, caçadas e cenas do cotidiano. Por meio de símbolos que representavam objetos e animais, os habitantes daquelas cavernas eternizavam a sua realidade. O homem evoluiu. Sua técnica de representar a realidade por intermédio de modelos também mudou.

O final do milênio nos surpreendeu com mudanças radicais em todos os níveis da atividade humana. Numa quantidade e velocidade nunca antes experimentadas. A era da informação não só já bateu à nossa porta, como ocupa nosso escritório, sala de aula e muitas vezes reparte conosco nossa própria sala de estar. Nem todos a perceberam, mas assim como a revolução industrial mudou o perfil da indústria mundial, a revolução da informação está modificando o perfil comportamental das pessoas e das organizações. Nunca, como nos últimos anos, a visão do negócio esteve tão próxima da visão do projeto dos sistemas de informação.

Com tudo isto, as técnicas, métodos e ferramentas de desenvolvimento de sistemas aplicativos mudaram e evoluíram de forma incrível. A Metodologia da Engenharia da Informação, por exemplo, trouxe-nos uma série enorme de ferramentas para o desenvolvimento eficaz de sistemas de informação, entre elas as técnicas formais da modelagem de dados.

No passado, o processo era o centro de tudo nos projetos de desenvolvimento de aplicativos. Estes eram os proprietários dos dados. A modelagem de dados era então simplesmente uma atividade paralela, quase desconhecida e muitas vezes desprezada. Quando utilizada, seu objetivo era meramente documentacional.

Com o reconhecimento de serem os dados um dos recursos mais importantes das corporações, ultrapassando as próprias fronteiras dos sistemas aplicativos, a modelagem de dados se tornou, com justa razão, a mais importante técnica utilizada na produção dos resultados das fases de planejamento e análise dos projetos de sistemas de informação.

Com o surgimento da Reengenharia de Negócios e o advento do Business Process Reengineering (BPR), volta-se a colocar nova ênfase nos processos. Procura-se agora um balanceamento perfeito entre a análise dos processos e dos

dados envolvidos, ambos centrados porém, na visão do negócio em que a informação é o objetivo final. Neste contexto, a modelagem de dados passa a ter uma importância maior ainda.

Os princípios básicos da modelagem de dados são amplamente conhecidos, ensinados e divulgados pelos mais diversos meios. Na vida real, porém, muitas vezes nos defrontamos com situações inusitadas, desconhecidas e que nunca foram cobertas ou sequer imaginadas nesses cursos e publicações. Nós, como projetistas de sistemas de informações, precisamos então acumular uma vasta experiência na área para fazer frente a esses eventos, ou procurar auxílio externo. E experiência é o que não falta aos autores deste livro que, de uma maneira clara, simples e agradável, transporta-nos dos primeiros passos da arte de modelar dados até suas últimas conseqüências. Com uma série de Estudos de Casos, eles nos transmitem toda a sua vivência, apresentando soluções para uma série de situações as quais certamente, se você ainda não enfrentou, mais cedo ou mais tarde irá encontrar.

O valor intrínseco desta obra, praticamente uma enciclopédia sobre Modelagem de Dados, reside especialmente na vivência desses dois projetistas de sistemas de informação, que sem academicismos obscuros dão ao assunto o tratamento que todos nós sempre quisemos ver em uma publicação técnica: clareza e objetividade.

Lívio Augusto Taufer
Diretor da RCM Informática

Sumário

1 - INTRODUÇÃO 01

1.1 - O Ciclo de Vida Tradicional ou em Cascata 04
1.2 - O Ciclo de Vida da Análise Estruturada 05
1.3 - O Ciclo de Vida da Engenharia de Software 07
1.4 - O Ciclo de Vida da Engenharia da Informação 08
1.5 - Informação - Um Recurso Empresarial 12
1.6 - Uma Nova Ordem - Começar por Dados 14
1.7 - Objetivos do Livro 15

2 - O FUTURO 17

2.1 - A Motivação 17

3 - MODELAGEM CONCEITUAL 21

3.1 - O Projeto de Banco de Dados 25
3.2 - O "Meta-Modelo" Entidade-Relacionamento (E-R) 27

4 - O MODELO ENTIDADE-RELACIONAMENTO 29

4.1 - Modelagem Conceitual de Dados 29
4.2 - Objetos Conceituais 29
4.3 - Entidades 32
4.4 - Atributos 35
4.5 - Enxergando Entidades 37
4.6 - Generalização e Especialização 40

5 - RELACIONAMENTOS 45

5.1 - A Existência 45
5.2 - Condicionalidade 48
5.3 - Relacionamentos Condicionais 50
5.4 - Relacionamentos Incondicionais 50
5.5 - A Viagem 51
5.6 - Grau do Relacionamento 54
5.7 - A Modelagem dos Relacionamentos 59

6 - EFETIVAÇÃO LÓGICA DOS RELACIONAMENTOS 67

6.1 - A Expressão do Relacionamento 67
6.2 - Quando os Fatos podem Confundir-nos 69
6.3 - Valor Nulo 73
6.4 - Como se Efetiva este Relacionamento? 76

7 - RELACIONAMENTOS ESPECIAIS 81

7.1 - Relacionamentos entre Múltiplas Entidades 81
7.2 - Relacionamentos Binários de Um-para-Um 87
7.3 - Usar N:N ou Construir 2 vezes 1:N - A escolha 96

8 - AGREGAÇÃO 95

8.1 - A Decomposição de um Relacionamento 95
8.2 - Agregação e Cardinalidade 100
8.3 - Restrições para Uso de Agregação 104
8.4 - Como Utilizar Agregação com Entidades Associativas 106
8.5 - Relacionamentos entre Blocos do Modelo 109

9 - AUTO-RELACIONAMENTO 113

9.1 - Introdução 113
9.2 - Auto-Relacionamento Um-para-Muitos 113
9.3 - Auto-Relacionamento Muitos-para-Muitos 116
9.4 - Auto-Relacionamento e Papel em um Relacionamento 118

10 - POR ONDE COMEÇAR? 121

10.1 - Iniciando um Modelo 122

11 - ESTUDOS DE CASOS 133

11.1 - Introdução 133
11.2 - Estudo de Caso 1 134
11.3 - Estudo de Caso 2 146
11.4 - Estudo de Caso 3 151

12 - NORMALIZAÇÃO 155

12.1 - Anomalias de Atualização 156
12.2 - Primeira Forma Normal (1FN) 158
12.3 - Variação Temporal e a Necessidade de Histórico 161
12.4 - Dependência Funcional 162
12.5 - Dependência Funcional Total (Completa) e Parcial 162
12.6 - Dependência Funcional Transitiva 163
12.7 - Segunda Forma Normal (2FN) 163
12.8 - Terceira Forma Normal (3FN) 165
12.9 - Forma Normal de BOYCE/CODD (FNBC) 167
12.10 - Quarta Forma Normal (4FN) 169
12.11 - Quinta Forma Normal (5FN) 171
12.12 - Roteiro de Aplicação da Normalização 175
12.13 - Desnormalização 177
12.14 - Considerações Finais Sobre Normalização 178

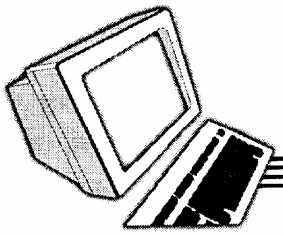
13 - O MODELO LÓGICO RELACIONAL 181

13.1 - Principais Vantagens da Abordagem Relacional 182
13.2 - As 12 Regras de Codd 183
13.3 - Chaves e Índices 184
13.4 - O Conceito de Chave no Modelo Relacional 184
13.5 - Regras de Integridade do Modelo Relacional 186
13.6 - Características do Modelo Relacional 186
13.7 - Derivação do Modelo E-R para o Modelo Relacional 187

14 - SQL **195**

14.1 - A Importância da Linguagem SQL 195
14.2 - A Linguagem SQL 196
14.3 - Vantagens e Desvantagens da Linguagem SQL 199
14.4 - O Exemplo 200
14.5 – Estudo de Caso 1 277
14.6 - Estudo de Caso 2 282
14.7 - Estudo de Caso 3 292

Capítulo



Introdução

1

Temos observado ao longo dos nossos anos de experiência no desenvolvimento de sistemas de informação, que existe uma grande dificuldade dos analistas e programadores em entenderem a diferença entre **INFORMAÇÃO** e **DADO**. Esta dificuldade traz, como conseqüência direta, problemas na especificação e modelagem de um sistema.

A INFORMAÇÃO acrescenta algo ao conhecimento da realidade a ser analisada. Por exemplo, a dosagem que um paciente precisa receber de um determinado remédio, é uma INFORMAÇÃO. Este conhecimento pode ser (ou não) modelado (registrado).

O DADO é uma representação, um registro de uma informação. Este DADO pode ser registrado fisicamente através de um papel (receita médica), um disco de computador ou um impulso elétrico, etc. Este registro pode ser o originador de uma série de processos que influenciam na realidade observada (salvar a vida de um paciente, tocar um alarme, etc.).

O tratamento das INFORMAÇÕES dá origem a vários tipos de DADOS, porém o DADO deve registrar apenas os aspectos realmente relevantes da INFORMAÇÃO, ou seja, o endereço do fabricante do remédio não tem nenhum interesse para um sistema de controle que mantém a vida dos pacientes em um CTI.

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

Para que possamos chegar à automação de uma realidade (CTI computadorizado de um hospital), é necessário que possamos entender esta mesma realidade. Fica evidente a impossibilidade de um analista ou programador conhecer todas as possíveis realidades que, ao longo de sua vida profissional aparecem para serem modeladas e automatizadas. Na verdade, o desenvolvimento do conhecimento humano, além de exigir uma contínua especialização, acaba por provocar também uma crescente necessidade de gente capacitada relacionar as partes com o todo: "projetista de sistemas", o qual deve ter a capacidade de sintetizar complexidades.

Tendo em mente a natural "restrição" humana de manipular grandes quantidades de informação ao mesmo tempo, foram criadas técnicas para se modelar os diversos problemas que existem. Estas técnicas, juntas, formam um conjunto conhecido como METODOLOGIA de produção de sistemas de informação.

A construção de sistemas pode ser encarada como a construção de um edifício (James Kowal [15]), a qual obedece a fases e procedimentos bem definidos. Na **figura 1.1,** é apresentada uma comparação entre a construção de sistemas e edifícios.

Construção de um edifício	Desenvolvimento de sistemas
criatividade no projeto	criatividade na análise e projeto
construção de modelos	construção dos modelos
introdução das últimas alterações	introdução dos últimos ajustes
várias recursões são efetuadas até o projeto final	várias iterações são efetuadas durante os estágios de análise e projeto
existência de padrões para aferição da qualidade do projeto	existência de padrões e regras para medição da qualidade do projeto

Figura 1.1

Construção de um edifício	Desenvolvimento de sistemas
detalhamento do projeto em vários níveis	o projeto consiste de vários níveis de detalhamento
o projeto é constituído por várias partes	os modelos são constituídos de várias partes
minimização da interface entre as partes do projeto	minimização da interface entre as partes do projeto
pouca criatividade na construção física	pouca criatividade durante a programação
especialistas manipulam diferentes partes do projeto	especialistas manipulam áreas específicas
aferição do progresso do projeto	gerentes de projeto avaliam a evolução do desenvolvimento
variação dos materiais e equipamentos	variação do hardware e software
componentes pré-moldados são usados	reutilização de código para aumentar a produtividade

Figura 1.1 (continuação)

Uma METODOLOGIA estabelece um caminho único no desenvolvimento de sistemas novos ou na evolução de sistemas já existentes. Ela introduz uma consistência ao longo do desenvolvimento de vários projetos de sistemas, provendo uma lista de todas as atividades a serem realizadas, estabelecendo pontos de checagem para auditoria e controle do projeto.

Desde 1950, várias metodologias estão sendo colocadas em prática. Estas metodologias definem o CICLO DE VIDA do desenvolvimento, no qual estão mostradas as fases que compõem o caminho a ser seguido pelos analistas e programadores, até a produção do sistema de informações na sua

versão operacional (figura 1.2). Cada fase pode ser vista como o refinamento da etapa anterior.

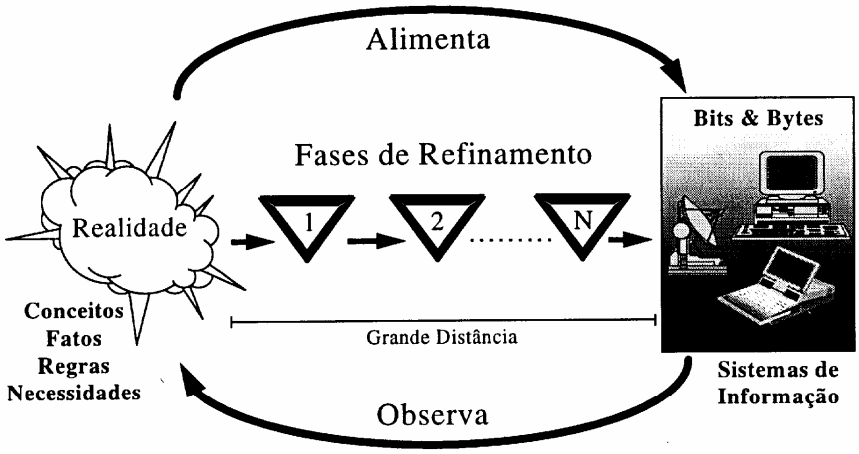


Figura 1.2

A seguir, serão apresentadas algumas metodologias que são bastante utilizadas no processo de desenvolvimento de sistemas de informação. Muitas delas já não apresentam o mesmo vigor de utilização de quando foram desenvolvidas, e praticamente formam um conjunto evolutivo no processo de captar as reais necessidades que um usuário possui em termos de automação de sua realidade.

1.1 - O Ciclo de Vida Tradicional ou em Cascata

Este ciclo de vida (figura 1.3) apresenta como principal característica a baixa interação dos usuários do sistema com o pessoal de desenvolvimento. Durante as etapas de Levantamento e Análise, o usuário tenta passar para o analista tudo que sabe sobre o problema e o que ele deseja para solucionar o mesmo. Após a definição do problema, é criado um documento, contendo os requisitos do futuro sistema, que é então congelado e utilizado durante todas as fases de desenvolvimento.

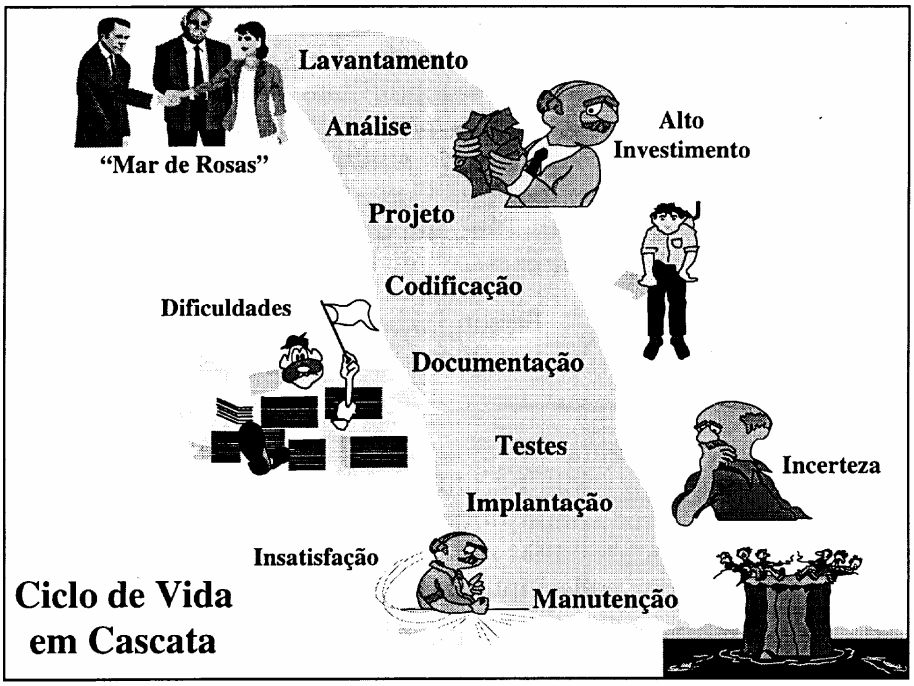


Figura 1.3

Neste ciclo de vida, não é criado nenhum tipo de modelo, não são utilizadas técnicas de estruturação e quase não existe oportunidade para o usuário realizar alguma alteração em pontos dos requisitos congelados. As atividades são realizadas em seqüência e não existem retornos entre as atividades. Toda a documentação é produzida após o término do projeto.

Fica evidente que os projetos realizados com este ciclo de vida se caracterizam pela alta incidência de manutenção, pois estão sujeitos a poucas alterações durante o desenvolvimento.

1.2 - O Ciclo de Vida da Análise Estruturada

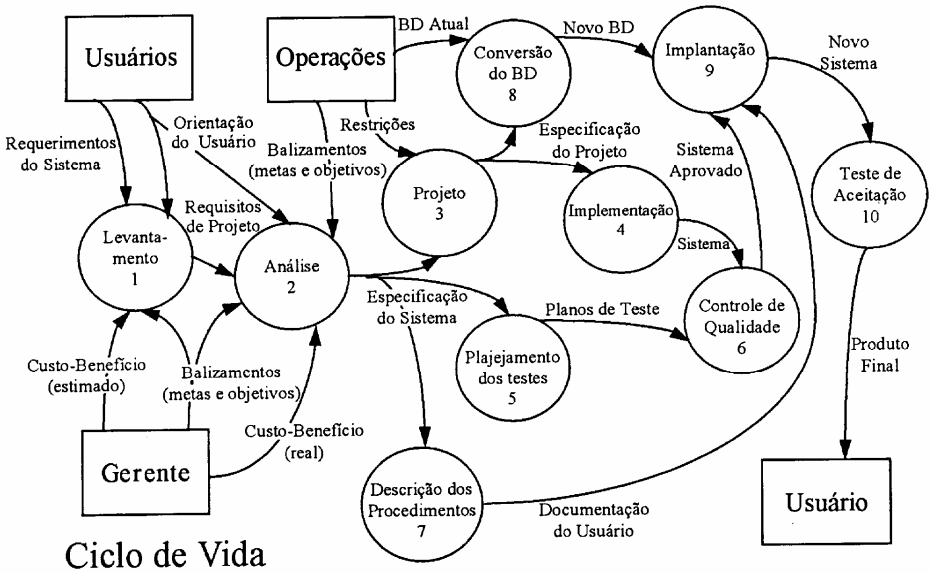
O conceito de programação estruturada foi introduzido em 1962, através de artigos escritos por E.W. Dijkstra [9] e C. Bohm/G. Jacopini [3]. Os dois últimos afirmavam que era possível escrever qualquer programa utilizando os três construtores básicos: seqüência, repetição e decisão. Eles afirmavam que utilizando estes construtores, a programação se tornar-se-ia mais fácil de entender e manter.

A partir destas idéias, no início dos anos 70, foram surgindo os conceitos do projeto estruturado (W. Stevens. - G. Myers - L. Constantine[21]), no qual se organizavam as funções de um programa de forma hierárquica, sobre a qual estão presentes dois conceitos fundamentais: **Acoplamento** - que retrata a comunicação entre os módulos do sistema, e **Coesão** - que fala a respeito das relações internas dos módulos. O produto final só estaria em um nível aceitável de qualidade para ser colocado em produção, quando possuísse **baixo acoplamento** e **alta coesão**.

Mais tarde, Cris Gane e T. Sarson [12], Tom DeMarco [8] e Edward Yourdon [24] publicaram livros descrevendo um método estruturado de analisar sistemas.

Este ciclo de vida (figura 1.4) é caracterizado pelo uso das técnicas estruturadas, incluindo as revisões estruturadas. Muitas das atividades são realizadas em paralelo, produzindo documentação nos vários estágios do desenvolvimento. Revisões periódicas são realizadas para se detectar o mais cedo possível problemas que podem influenciar no produto final.

Neste ciclo de vida, o envolvimento do usuário é bastante significativo. Sua participação na maioria das revisões traz novas sugestões e correções dos aspectos não compatíveis com suas necessidades.

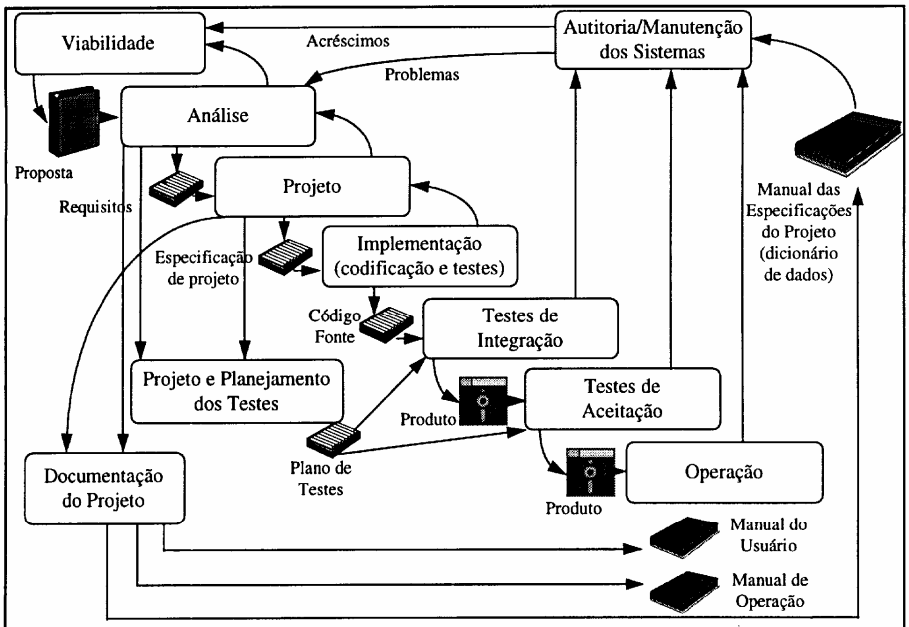


Ciclo de Vida Estruturado

Figura 1.4

1.3 - O Ciclo de Vida da Engenharia de Software

Durante os anos 70 com a utilização da análise estruturada, o desenvolvimento de sistemas ganhou um impulso muito grande. Em decorrência deste impulso, a necessidade de novos sistemas cresceu rapidamente e com isso as manutenções também começaram a ter um papel proeminente no ciclo de vida dos sistemas. Isto tudo levou a um aumento natural do custo de desenvolvimento e manutenção. Começou então a surgir uma preocupação maior relativa à produtividade dos analistas e programadores, com a qualidade dos produtos e com os aspectos de segurança de programas.



Ciclo de vida da Engenharia de Software

Figura 1.5

Com base nestas preocupações, foi criado o ciclo de vida da engenharia de software (figura 1.5), o qual veio para preencher certas lacunas deixadas pelo ciclo de vida da análise estruturada. Na engenharia de software se busca uma maior disciplina em termos de desenvolvimento de sistemas. Ela é caracterizada pela forte orientação por processos, pela determinação bem acentuada de cada fase, enfatiza a reutilização de código de programa, provê revisões e pontos de checagem bem determinados e define métricas

bem fundamentadas para o gerente realizar o controle da produtividade, a qualidade e o custo do produto final. Algumas métricas da engenharia de software foram apresentadas por Bany Boehm [2] e por Arndt Von Staa [20].

A engenharia de software é fundamentada em sete fases: viabilidade, análise, projeto, implementação, teste do sistema, teste do usuário e produção. Quando algum problema ocorre em uma das fases, retorna-se a fase imediatamente anterior para se rever os passos que levaram ao desenvolvimento daquela onde ocorreu o problema.

1.4 - O Ciclo de Vida da Engenharia da Informação

No decorrer de 20 anos de uso de técnicas para o desenvolvimento de sistemas, no qual a idéia central era analisar com base nos processos atuais apresentados no ambiente do usuário e nos propostos para se chegar ao sistema final, começou-se a notar que os processos dentro de uma empresa, corporação, repartição, etc. eram fortemente influenciados pelo meio ambiente externo aos locais de utilização destes processos (figura 1.6).

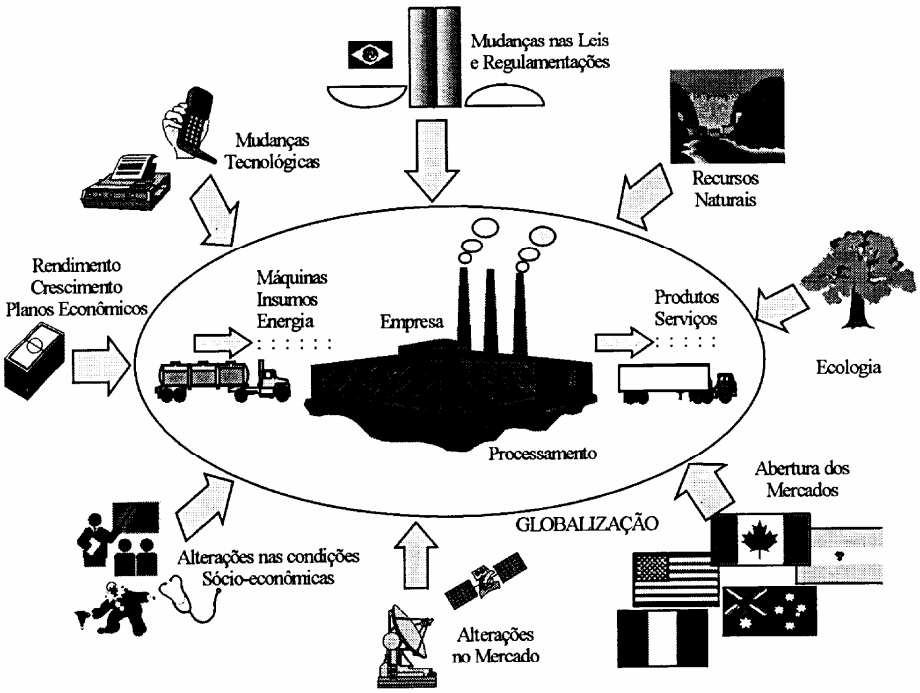


Figura 1.6

Com estas observações concluiu-se que os dados envolvidos em cada processo eram extremamente estáveis, se comparados com os processos. Esta estabilidade era devido ao fato de que os dados só sofrem algum tipo de mudança no momento em que o negócio também muda/evolui.

Através destas conclusões, em 1981, Matt Flavin [11], James Martin e Clive Finkelstein [17] introduziram o conceito de engenharia da informação. O princípio fundamental baseava-se no fato de que o dado existe e é descrito, independentemente dos processos que podem utilizá-lo.

Como o centro desta metodologia é o DADO, a idéia principal é levantar as estruturas de dados que vão dar origem aos bancos de dados, provendo um fácil acesso aos mesmos.

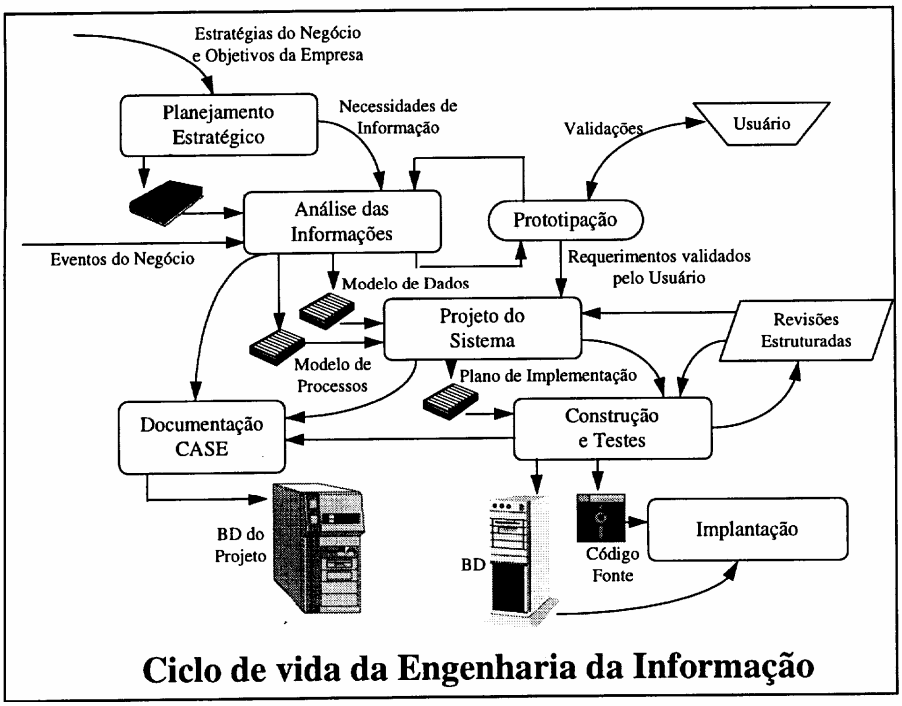


Figura 1.7

A engenharia da informação (figura 1.7) é um conjunto integrado de técnicas que organiza os dados de um determinado negócio e determina um acesso fácil, por parte do usuário final, a estes dados. Esta metodologia pode ser detalhada nas seguintes fases: planejamento estratégico das informações, análise da informação, modelagem de dados, formação dos procedimentos,

análise do uso dos dados, análise da distribuição dos dados, projeto físico da base de dados e especificação dos programas.

O suporte desta metodologia está baseado na técnica de modelagem de dados e seus relacionamentos, desenvolvida inicialmente por Peter Chen [5] em 1976, chegando à modelagem da informação através de Flavin [11] em 1981, e finalmente à modelagem semântica dos dados através de Shlaer e Mellor [19] em 1988.

Devido ao crescimento natural dos negócios (Veja "O Armazém" - quadro abaixo), das necessidades de informação e do aprimoramento dos sistemas de gerenciamendo de banco de dados (SGBD), que passaram a ser cada vez mais utilizados pelas empresas, a modelagem de dados passou a ser um fator fundamental no desenvolvimento de sistemas de informação.

O Armazém

Para ilustrarmos melhor a evolução das necessidades de informação, vamos analisar o crescimento das empresas no país. A grande maioria das empresas nacionais teve seu início de atividade caracterizado por um negócio de pequeno porte. Para ilustrar essa figura típica, poderíamos utilizar a imagem de uma pequena quitanda, um armazém de bairro. Ali o proprietário (um "futuro" grande empresário) administra seu negócio junto com seus familiares. Observa-se que existe um domínio completo sobre o negócio e sobre suas informações. O proprietário obtém dados rapidamente sobre seus estoques, simplesmente olhando para os seus itens nas prateleiras. Conhece pessoalmente seus clientes, normalmente pessoas do bairro, permitindo que comprem a crédito através de um caderno de "pinduras" por cliente, e também conhece seus fornecedores pessoalmente. O controle do fluxo de caixa resume a simples operações de pagar, receber e sobrar, diretamente na caixa registradora. Existe então a informação completamente acessível, disponível e de qualidade suficiente para o negócio.

Mas como este é um país de iniciativas (???é mesmo???), o bom atendimento prestado pelo proprietário do armazém faz com que a clientela aumente, solicite produtos melhores, mais variedade e quantidade, além de formas de pagamento diferenciadas, etc., e o negócio vai crescendo, até que encontramos o velho armazém transformado em um supermercado.

Com o crescimento, o volume de informações cresce de forma quase que exponencial, qualifica-se e especializa-se. O proprietário daquele antigo armazém, agora um promissor supermercado, conta com profissionais especializados que o auxiliam no atendimento de atividades específicas, tanto junto aos clientes quanto no controle do negócio, e

principalmente, para controlar as informações agora setorizadas e especializadas. Existem, na empresa, atividades profissionais específicas tais como: contador, comprador, almoxarife, etc.

Este é um primeiro instante da "departamentalização" da empresa. Mas o que nos interessa salientar é que se observa uma grande dificuldade do ser humano em lidar com grandes massas de informação. Já não é mais possível, ao proprietário, manipular e ter o domínio completo das informações de seu negócio. Isto caracteriza o que denominamos de aumento do domínio do problema.

Neste instante, pode ser estratégico e crítico para o negócio a necessidade de informatização do supermercado.

Mas a questão é: O que estamos procurando solucionar? Qual é o problema?

A resposta correta é, e sempre será, a necessidade de informação.

Os Dados

O desenvolvimento de aplicações computadorizadas, orientadas pelas necessidades setoriais de cada área, utilizando equipamentos e softwares variados não irá resolver o problema, mas paliativamente passar a sensação de que as coisas estão andando rapidamente, e que todos têm controle sobre suas situações e necessidades. Mas e qual é a realidade afinal?

Em toda e qualquer empresa que se analise, constataremos que as informações se interligam constantemente, não permanecendo isoladas e estanques em um determinado setor de uma empresa. Logo, dados que são gerados por Vendas, são necessários a Compras, para que esta disponibilize a entrega das vendas, estabeleça os pontos de ressuprimento dos produtos considerando e calculando o giro de estoque. Da mesma forma os dados de Compras interessam para o Financeiro, já que devemos providenciar uma previsão de desembolso. Os dados de Compras são importantes para Vendas para estabelecimento de preços. Poderíamos continuar todo este livro fazendo ligações entre departamentos, mas não é este o objetivo. O que queremos é salientar a importância das informações no negócio da empresa, e que elas não produzem resultados se isoladas em órgãos.

Existe a clara necessidade de interligação de todas as informações, para que se venha a obter uma visão corporativa do negócio, e que seja explícita o suficiente para nos mostrar a empresa como ela realmente é.

Um sistema de informações realizado de forma estanque e departamentalizada, vai ter como característica um fluxo de dados extremamente complexo entre os departamentos, mas não representando uma realidade clara e límpida do negócio.

As áreas operacionais da empresa acabam possuindo bases de dados próprias para sua utilização exclusiva, não importando a sua utilidade e importância no resto da empresa.

A cada área "informatizada" correspondem processos de atualização distintos e particulares (feudos de PD), que consomem recursos inversamente proporcionais aos resultados que produzem.

Logo observa-se na empresa, um nível de inconsistência de informações muito alto, criando dificuldades sensíveis no processo gerencial da mesma, com um custo significativo. E de nada adiantam as soluções paliativas de atualizações simultâneas entre os diversos sistemas e bases de informação, buscadas num último instante como solução de processamento de dados. É típico observarmos em várias empresas, a disputa pela posse da informação mais confiável, em reuniões gerenciais para análise e definição de estratégias, que em nada contribuem para processo como um todo.

Frase típica: "Os dados do "meu" sistema são os certos!" ou então a clássica: "O CPD não funciona, gastamos um dinheirão à toa". Enquanto se briga por autoria e buscam-se culpados, a empresa está parada, sem informações que possibilitem a sua administração eficiente. A busca de culpados em nada resolve o problema do dono do negócio, somente com a adoção de uma metodologia correta e um envolvimento corporativo poderemos obter o caminho correto.

1.5 - Informação - Um Recurso Empresarial

Devemos encarar as informações como um bem patrimonial de uma empresa, como são cadeiras, mesas, prédios, máquinas, etc.

A informação deve ser utilizada de uma maneira estratégica, para que possa atender e atingir muito rapidamente os objetivos, metas e desafios traçados pela alta gerência de um negócio. Esta velocidade de mudança faz com que qualquer negócio possa aproveitar uma oportunidade de competição de mercado, sabendo que as informações estratégicas, táticas e operacionais estão disponíveis a qualquer momento para tomada de decisões.

Fica evidente que a utilização deste enfoque passa necessariamente por uma nova realidade tecnológica em termos de hardware e sofztware. Felizmente, esta nova realidade já está disponível, com equipamentos mais compactos e mais velozes e com SGBD's mais poderosos (ex.: ORACLE®, ZIM®, SYBASE®) que podem incorporar esta forma de enfocar o desenvolvimento de sistemas de informação.

Devido ao aumento da complexidade e mutabilidade dos sistemas, é bastante necessária a utilização de uma ferramenta CASE - Computer Aided Software Engineering - integrada e eficiente (ex.: Key for Enterprise®, Answer:Arquitect®, ORACLE CASE®), que nos auxilie a gerenciar toda gama de informações que é passada pelos usuários e que ainda forneça uma ligação com o SGBD que for utilizado pela empresa.

Com um ambiente de desenvolvimento utilizando estes enfo-ques poderemos garantir que as improvisações e incertezas vão se tornar coisas do passado e que os sistemas vão se tornar mais eficientes e abrangentes.

Devemos direcionar nossos trabalhos de análise para o objetivo maior da organização que é a qualidade da informação.

A orientação a procedimentos nos leva a um oceano de métodos e estruturações que, em nada ou muito pouco contribui para obtermos a compreensão do negócio da organização, para a qual desejamos desenvolver um sistema de informações.

A questão então torna-se simples: Como fazer sistemas de qualidade se não temos o domínio do problema (o negócio da organização) e principalmente de seus dados, significados, e importância dos mesmos.

Quando se fala em desenvolver um sistema de informações, é necessário que se defina qual o primeiro passo nas atividades de análise. Torna-se cada dia mais óbvio que é de extrema necessidade que obtenhamos o conhecimento sobre o que são e como são as informações, os dados do negócio, pois eles refletem o próprio negócio da organização. Com base neste enfoque, podemos definir como primeiro passo nas atividades de análise a modelagem de dados.

Por exemplo, um analista que não conhece o setor de produção de uma indústria. Se seu trabalho estiver orientado para procedimentos, evidentemente ele irá consumir uma grande parcela de seu tempo no detalhamento dos fluxos operacionais indicados pelos usuários, e após conviver algum tempo com o setor, poderá ter obtido ou não conhecimento sobre o negócio para o qual ele desenvolve um sistema. O resultado de seu trabalho de análise é uma massa de fluxos e procedimentos para um sistema, mas seguramente não terá domínio completo do problema.

Absolutamente não queremos afirmar que o sistema não irá funcionar, mas podemos dizer com segurança que a margem de erros certamente será

alta, e que o nível de qualidade será provavelmente baixo. Destacamos que não existiu um trabalho de busca das informações, logo não se buscou o conhecimento do problema e sim uma solução para algo momentâneo e provavelmente passageiro.

1.6 - Uma Nova Ordem - Começar por Dados

O que pretendemos ao lançarmos este trabalho, é apresentar um enfoque metodológico baseado inteiramente na definição e modelagem dos dados da organização, para a construção dos sistemas de aplicação, e que possuam em primeiro lugar, qualidade e confiabilidade de informações. Para isto vamos orientar nossos trabalhos de análise no levantamento e definição do aspecto fundamental de um sistema, seus dados.

Os dados não possuem características tão voláteis que sua existência assuma caracterização temporal acentuada, pois nos refletem não um momento a ser modificado, mas sim uma realidade a ser automatizada.

A modelagem de dados (ou de informações) está baseada no princípio de que, comprovadamente, os dados são estáveis no decorrer da vida de uma empresa ou organização, não tendo volatilidade dependente de fatores pessoais, governamentais e temporais. Já os procedimentos possuem esta característica de volatilidade, pois sofrem constantes alterações, seja por fatores pessoais, quando existe troca de pessoas e métodos, por decisões governamentais e legislativas, por fatores de calamidade, e outros, externos às atividades normais da empresa.

Para que uma empresa mude seus dados, torna-se necessário que a mesma mude sua atividade-fim, por exemplo: em uma indústria metalúrgica, seus dados somente irão mudar quando a mesma se transformar em uma empresa agrícola ou fechar e abrir como ferro-velho.

Logo podemos concluir que os dados, na realidade, são imutáveis durante o ciclo de vida de uma empresa, mudando somente os valores presentes nos dados, mas não o conceito do dado em si.

Quando passamos a adotar uma metodologia baseada em modelo de dados, com trabalhos iniciais e extensos de modelagem, conseguimos obter um domínio do negócio da empresa, temos uma fotografia global da mesma.

Mas isto nos dá algum ganho no processo de desenvolvimento? Se considerarmos que os ganhos do processo de desenvolvimento são uma relação direta entre tempo, qualidade e custo, torna-se óbvio que a análise de um sistema com técnicas que permitam conhecimento perfeito e compreensão ampla do negócio em um espaço de tempo reduzido, nos permitirá desenvolver e obter qualidade na aplicação, principalmente sobre a informação, e conseqüentemente um retorno positivo na relação de custo x benefício para o projeto de desenvolvimento do sistema de informações.

1.7 - Objetivos do Livro

A finalidade deste livro é apresentar de forma didática e prática o desenvolvimento da modelagem de dados. Serão abordados aspectos lógicos e físicos, mostrados através de exemplos desenvolvidos em SQL-ORACLE®.

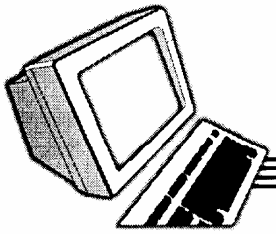
Como vimos no início deste capítulo, existem várias abordagens (ciclos de vida) para o desenvolvimento de sistemas de informação. Neste livro, estaremos preocupados em construir bases de dados que atendam às reais necessidades de informação de um determinado ambiente.

A construção de bases de dados pode ser dividida em três fases bem distintas: modelagem conceitual, projeto lógico e projeto físico, que podem estar presentes em qualquer ciclo de vida e foram desenvolvidas no final dos anos 70.

Iremos, ao longo dos capítulos, detalhar a construção do Modelo Conceitual, o qual apresenta as necessidades de informação captadas. Nesta fase, são registrados os fatos que ocorrem na realidade e que de certa forma influenciam no funcionamento da mesma.

O projeto de um banco de dados é um processo complexo e que geralmente necessita de muitas decisões em diferentes níveis. Esta complexidade é melhor gerenciada quando o problema é subdividido em vários subproblemas independentes ("dividir para conquistar"). O fato central nas três fases do projeto de bases de dados é a modelagem do dado e suas propriedades.

Capítulo



O Futuro

2

Vocês, leitores, devem estar achando no mínimo estranho, o capítulo 2 falar sobre **O Futuro**. Normalmente, um tema como este só aparece no fim do livro. Nós sentimos que será muito importante você saber que aquilo que o espera no futuro tem como base os ensinamentos presentes nos próximos capítulos. Esta percepção lhes dará mais motivação para estudar de forma profunda e consciente.

2.1 - A Motivação

Todos já devem ter notado na introdução, que a grande preocupação em qualquer levantamento feito é com o domínio do problema e as responsabilidades do sistema. Durante muitos anos o desenvolvimento de sistemas de informação era dividido entre o levantamento dos processos (prioridade) e o levantamento dos dados. Esta forma de análise trouxe inúmeros problemas para o usuário, pois o mesmo não via o produto como uma solução para suas necessidades.

Como vimos no capítulo 1, trabalhar com os dados foi um fator vantajoso no desenvolvimento de sistemas de informação. Mesmo com a modelagem da informação mapeando diretamente o domínio do problema, ainda havia a necessidade de um detalhamento maior. Os cientistas [25] da área de sistemas de informação começaram então a juntar dados e processos em um único elemento. Esta união deu origem ao conceito de modelagem orientada a objetos (OO), a qual passou a fornecer um elo de ligação mais estável entre o mundo real e os sistemas de informação. Podemos dizer que OO é um novo pensamento sobre os problemas, organizando modelos cada vez mais próximos dos conceitos do mundo real.

As técnicas de orientação a objetos apresentam vários benefícios:

1 - Utilizam um conceito mais especializado de detalhamento da realidade (herança);

2 - Trabalham com o conceito de reutilização, que permite uma maior produtividade;

3 - Aumentam a consistência dos resultados da análise;

4 - Produzem uma melhor ligação entre o analista e o usuário;

5 - Suportam melhor alterações na realidade;

6 - Podem enfrentar, de forma mais direta, domínios mais complexos na realidade;

7 - Possuem uma maior continuidade em todas as fases do ciclo de vida do projeto.

Esta nova forma de exploração da realidade utiliza, diretamente, os princípios básicos da administração da complexidade enunciados por Coad e Yourdon [25]:

1 - abstração (dados e procedimentos): foco no essencial;

2 - encapsulamento (*information hiding*): invisibilidade dos aspectos internos de um objeto, quando observado por outro objeto;

3 - herança: objetos podem herdar características (dados e processos) de outros objetos;

4 - comunicação entre os objetos através de mensagens;

5 - polimorfismo: características diferentes para um mesmo objeto ao mesmo tempo;

6 - métodos de organização (objetos e atributos, todo e partes, classes e membros com distinção entre eles);

Basicamente, a modelagem da informação atende aos princípios 1 e 6, porém com relação aos demais deixa muito a desejar.

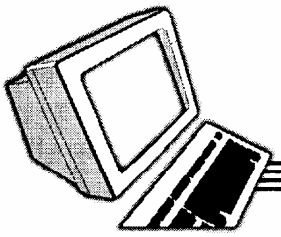
Os sistemas elaborados hoje em dia, são diferentes dos que eram desenvolvidos há dez ou vinte anos. São maiores, mais complexos, mais voláteis e sofrem alterações constantes. E principalmente, hoje se dá uma grande ênfase à interface homem-máquina, a qual se tornou bastante

poderosa e complexa, levando a ter um papel preponderante no código final (aproximadamente 75%).

A OO nasceu com a finalidade de dar maior poder de produtividade e aumentar a qualidade dos produtos desenvolvidos.

Mesmo com a entrada firme da modelagem de dados na maioria das empresas, já se fala em análise, projeto e programação orientada a objetos. A modelagem orientada a objetos tem seu fundamento na modelagem da informação. Este livro tem como preocupação central apresentar a técnica de modelagem de dados. Quanto ao futuro (orientação a objetos), será tema para um outro encontro entre vocês, leitores, e nós, autores. Enquanto este novo encontro não acontece, fiquem com a base de toda esta nova tecnologia. **Mãos à obra.**

Capítulo



Modelagem Conceitual

3

Toda a realidade é sempre, em princípio, bastante nebulosa e informal. Através da observação podemos extrair dela (realidade) fatos que nos levam a conhecê-la de uma forma mais organizada.

Em um negócio, existem fatos que, observados e modelados, dizem algo a respeito do funcionamento deste negócio. Estes fatos estão ligados diretamente ao funcionamento da realidade, os quais temos grande interesse em compreender e manter.

Para que possamos retratar estes fatos e que os mesmos possam nos levar a futuras decisões e ações, se faz necessário então registrá-los. Este registro é feito através da criação de um modelo.

É evidente que os fatos ocorrem a todo instante dentro de uma realidade, geralmente ficam registrados em documentos formais, tais como: fichas, memorandos, requerimentos, leis, protocolos, decretos e na maioria dos casos, estão registrados na cabeça das pessoas que de forma direta ou indireta influenciam na realidade a ser modelada. Normalmente, na criação e utilização desses documentos não há nenhuma preocupação em se utilizar, no futuro, um ambiente automatizado.

O analista, durante a modelagem conceitual dos dados, deve se concentrar na observação dos fatos relevantes que ocorrem na realidade, com a finalidade de construir um sistema que possa automatizar as necessidades de informação da mesma. Neste momento, os documentos que registram estes fatos só devem ser utilizados como apoio ao entendimento, e não como base para o desenvolvimento do sistema de informações, ou seja, não devemos ter a preocupação em simular o ambiente atual, seja ele manual ou automatizado.

A preocupação que o analista deve ter é: retratar as necessidades de informação que as pessoas (que agem sobre esta realidade) precisam para alcançar os objetivos desta mesma realidade.

Ao coletar e selecionar os fatos relevantes, o analista deve identificar os elementos geradores de informação, as leis que regem esta realidade, bem como as operações que incidem sobre os elementos básicos (dados).

O que se quer criar é uma abstração (figura 3.1) da realidade, que seja capaz de registrar os acontecimentos da mesma, de tal forma que se possa implementar um sistema automatizado que atenda as reais necessidades de informação

Para registrarmos as necessidades de informação de uma realidade, precisamos fazer uso de um modelo, ou seja, algo que nos mostre como as informações estão relacionadas (fatos). E com base no modelo criado, os analistas podem interagir com os usuários validando seus objetivos e metas, permitindo a construção de um sistema de informações cada vez mais próximo da realidade do usuário.

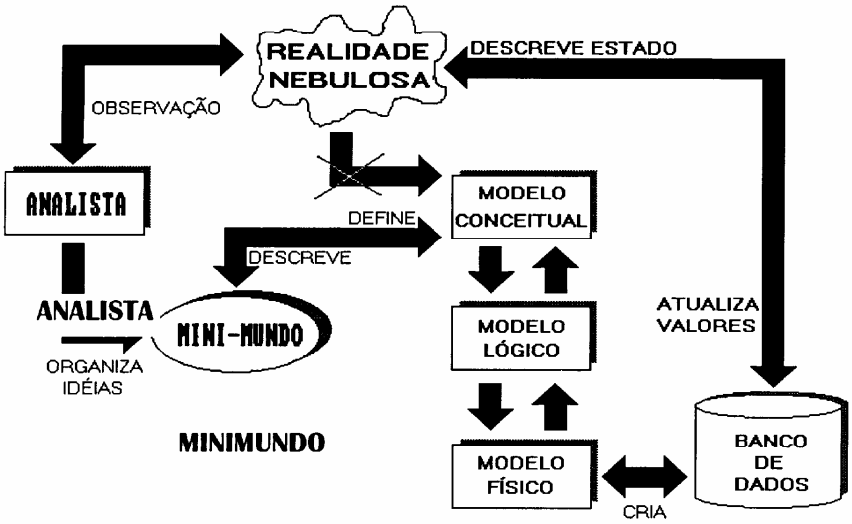
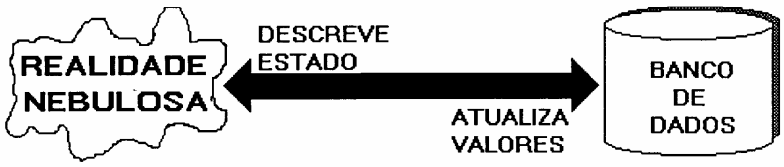


Figura 3.1 - Níveis de Abstração.

Descrição dos elementos da abstração:

- **Minimundo**: Porção da realidade, captada pelo analista, a qual a função gerencial tem forte interesse em observar. A complexidade de se analisar até mesmo um minimundo, pode levar o analista a subdividi-lo em partes menores, às quais damos o nome de visão.

- **Banco de Dados**: É uma coleção de fatos registrados que refletem o estado de certos aspectos de interesse do mundo real. A todo o momento o conteúdo do banco de dados representa uma visão instantânea do estado do mundo real. Cada mudança em algum item do banco de dados reflete uma mudança ocorrida na realidade.

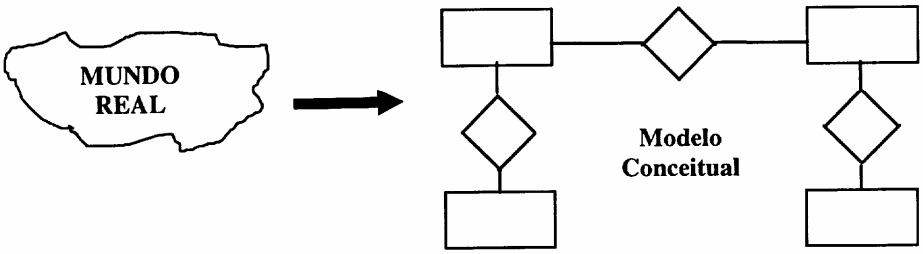


A tecnologia de banco de dados tem como fundamento básico permitir que os dados possam ser definidos e mantidos, independente dos sistemas de aplicação que venham a utilizá-los (independência DADO x PROCESSO).

- **Modelo Conceitual**: Representa e/ou descreve a realidade do ambiente do problema, constituindo-se em uma visão global dos principais dados e relacionamentos (estruturas de informação), independente das restrições de implementação.

Quando se fala em Modelo Conceitual, estamos nos referindo a primeira etapa do projeto de um sistema de aplicação em banco de dados.

O objetivo do Modelo Conceitual é descrever as informações contidas em uma realidade, as quais irão estar armazenadas em um banco de dados. É uma descrição em alto nível (macrodefinição), mas que tem a preocupação de captar e retratar toda a realidade de uma organização, setor, repartição, departamento, etc.

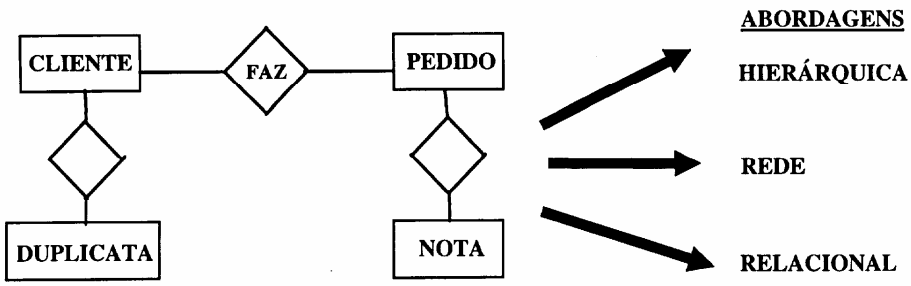


O Modelo Conceitual não retrata os aspectos ligados à abordagem do banco de dados que será utilizado e tão pouco se preocupa com as formas de acesso ou estruturas físicas implementadas por um SGBD específico.

O resultado de um Modelo Conceitual é um esquema que representa a realidade das informações existentes, assim como as estruturas de dados que representam estas informações.

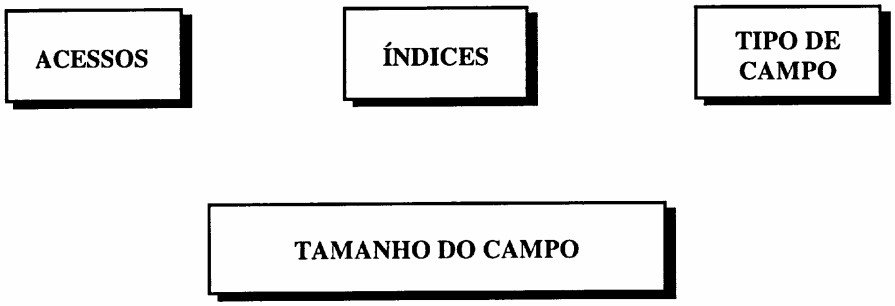
O Modelo Conceitual não é construído com considerações procedurais, não existindo preocupação com as operações de manipulação/manutenção dos dados. É na fase de modelagem conceitual que iremos executar os trabalhos de construção de um modelo de dados.

- **Modelo Lógico**: O Modelo Lógico tem seu início a partir do Modelo Conceitual, levando em consideração uma das três abordagens atualmente possíveis: Relacional, Hierárquica e Rede.



O Modelo Lógico descreve as estruturas que estarão contidas no banco de dados, de acordo com as possibilidades permitidas pela abordagem, mas sem considerar, ainda, nenhuma característica específica de um Sistema Gerenciador de Banco de Dados (SGBD), resultando em um esquema lógico de dados sob a ótica de uma das abordagens citadas.

- **Modelo Físico**: O Modelo Físico irá partir do Modelo Lógico e descreve as estruturas físicas de armazenamento de dados, tais como: tamanho de campos, índices, tipo de preenchimento destes campos, nomenclaturas, etc, projetadas de acordo com os requisitos de processamento e uso mais econômico dos recursos computacionais. Este modelo detalha o estudo dos métodos de acesso do SGBD, para elaboração dos índices de cada informação colocada nos Modelos Conceitual e Lógico.



Esta é a etapa final do projeto de Banco de Dados, na qual será utilizada a Linguagem de Definição de Dados do SGBD (DDL), para a realização da montagem do mesmo no nível do Dicionário de Dados.

3.1 - O Projeto de Banco de Dados

Todo o projeto de um sistema de aplicação para banco de dados necessita de um coração, um centro nervoso do mesmo. A modelagem de um sistema através da abordagem Entidade Relacionamento representa este ponto central no Projeto Conceitual de um Sistema.

O objetivo da Modelagem de Dados é transmitir e apresentar uma representação única, não redundante e resumida, dos dados de uma aplicação. Em projetos conceituais de aplicações em banco de dados o modelo Entidade-Relacionamento é o mais largamente utilizado para a representação e entendimento dos dados que compõem a essência de um sistema.

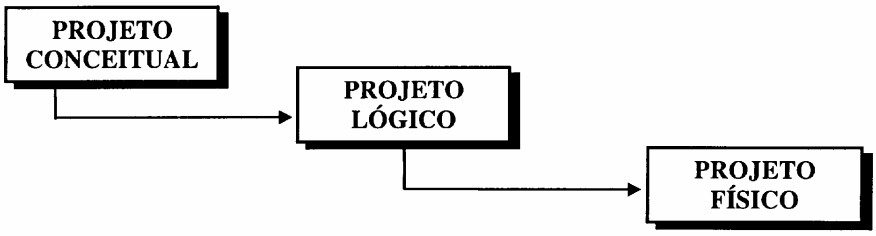
O Projeto de Banco de Dados para sistemas de aplicação hoje não é mais uma tarefa realizada somente por profissionais da área de informática, mas também possível de ser realizada por não especialistas, através de técnicas estruturadas como a Modelagem Conceitual de Dados.

O projeto de um sistema de informações é uma atividade complexa que inclui planejamento, especificações e desenvolvimento de vários componentes.

O que se propõe é situar a seqüência destas atividades em uma ordem que possa resultar em ganhos de produtividade e confiabilidade dos sistemas desenvolvidos, eliminado-se sensivelmente as falhas de sistema após sua implantação.

Desafortunadamente as metodologias de projeto de banco de dados, para sistemas de aplicação, não são ainda muito populares entre a comunidade técnica, sendo que várias organizações e profissionais se utilizam de pequenas técnicas pessoais, ou ainda pior, de uma inexistência completa de metodologia para estes projetos, sendo isto uma das maiores causas de falhas nos sistemas de informação desenvolvidos.

A utilização de uma abordagem correta de metodologia orientada a banco de dados envolve a estruturação nos três níveis de visão de dados vistos anteriormente, ou seja, três etapas na execução de um projeto: conceitual, lógico e físico.



Isola-se desta forma a realidade a ser retratada em dados de suas implicações lógicas e físicas, determinando-se o momento para adequação do modelo a estes fatores.

É evidente e óbvio que a realidade dos negócios de uma empresa é sempre diferente da realidade de outra empresa, mesmo que falem de ambientes similares. Existem particularidades que só dizem respeito ao funcionamento daquele ambiente específico.

Devido a esta não similaridade entre ambientes de mesma natureza, será sempre necessária a criação de um modelo específico para cada nova realidade observada. Fica bem claro então, a necessidade de termos um modelo que nos permita construir vários outros modelos, o qual é chamado de "meta-modelo".

O "meta-modelo" a ser utilizado deve ter a característica de poder modelar qualquer realidade, ter uma forma de trabalho bastante simples e deve ter características gráficas que sejam bastante simples de construir e entender. O "meta-modelo" que iremos utilizar neste livro será o Entidade-Relacionamento (E-R).

3.2 - O "META-MODELO" Entidade-Relacionamento (E-R)

O modelo Entidade-Relacionamento foi definido por Peter Chen em 1976, e teve como base a teoria relacional criada por E. F. Codd (1970).

Ao longo dos anos, vários estudiosos (Theorey, Fry, James Martin entre outros) evoluíram e expandiram este "meta-modelo". No próximo capítulo, iremos apresentar uma visão bastante moderna a respeito dele.

Segundo Chen, a visão de uma dada realidade, baseia-se no relacionamento entre entidades, os quais retratam os fatos que governam esta mesma realidade, e que cada um (entidade ou relacionamento) pode possuir atributos (qualificadores desta realidade).

Grande parte das extensões ao "meta-modelo" baseiam-se em alguns mecanismos de abstração: classificação, generalização e agregação. O conceito de abstração permite ao analista separar da realidade em estudo, as partes que são realmente relevantes para o desenvolvimento do sistema de informações e excluir da modelagem todos os aspectos que não exercem influência sobre o ambiente a ser modelado.

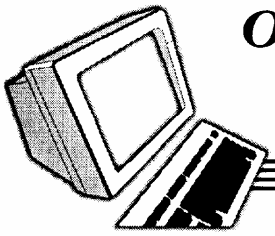
Os conceitos do modelo E-R destinam-se prioritariamente ao projeto de banco de dados, mas podem ser utilizados para o entendimento de um determinado negócio (modelo do negócio) bem como auxiliar o desenvolvimento de estruturas de dados que possam ser implementadas fora de um ambiente de banco de dados, utilizando-se uma linguagem de programação (COBOL, C, PASCAL, etc.). O modelo E-R é, atualmente, a base para o desenvolvimento de sistemas orientados a objetos.

No próximo capitulo, vocês terão a oportunidade de ter contato com este modelo e observar a sua facilidade de uso e sua alta legibilidade.

O objetivo da modelagem de dados é possibilitar a apresentação de uma visão única, não redundante e resumida dos dados de uma aplicação. No desenvolvimento de aplicações em banco de dados, o modelo Entidade-

-Relacionamento (E-R) é o mais largamente utilizado para a representação e entendimento dos dados que compõem a essência de um sistema de informações.

Capítulo



O Modelo Entidade--Relacionamento

4

4.1 - Modelagem Conceitual de Dados

Ao se utilizar a Modelagem Conceitual de Dados com a técnica de Entidades e Relacionamentos, obteremos resultados e esquemas puramente conceituais sobre a essência de um sistema, ou melhor sobre o negócio para o qual estamos desenvolvendo um projeto, não representando-se procedimentos ou fluxo de dados existentes.

A modelagem é como a arte fotográfica, prepara-se a câmera e tira-se a foto, sem se importar com os movimentos. Entretanto, o esquema produzido pelo trabalho de modelagem de dados nos possibilita a visualização das atividades e procedimentos que poderão ser exercidos sobre estas estruturas de dados.

4.2 - Objetos Conceituais

As literaturas existentes nunca deixam claro como podemos entender entidades e relacionamentos. Uma vez que a maioria dos profissionais de análise de sistemas tem sua cultura baseada em sistemas procedurais, onde os dados são o resultado e não o meio, existe a necessidade de que se coloque mais enfoque didático no detalhamento do que são efetivamente entidades e relacionamentos.

As técnicas estruturadas mais avançadas, que neste país alcançaram divulgação profissional a nível prático, baseiam-se na análise dos procedimentos, e com enfoque principalmente direcionado para o Diagrama

de Fluxo de Dados. Estas técnicas estruturadas colocam as informações derivadas dos procedimentos em Depósitos de Dados, os quais finalmente acabam sendo traduzidos em arquivos de um sistema.

Para que se efetue então a migração desta base cultural, torna-se necessário que a regra básica - **procedimentos não nos interessam** - seja atendida nesta abordagem de levantamento.

Vamos estabelecer como preocupação somente a necessidade de retratarmos as informações existentes no negócio. Nosso objetivo primordial é entendermos o negócio, para o qual projetaremos um sistema, através de seus dados.

Quando escrevemos **Objetos Conceituais**, não estamos pretendendo nos inserir na Orientação a Objetos. Apesar de a modelagem conceitual de dados ser a base para o entendimento desta nova abordagem tecnológica, o nosso objetivo é na realidade ir até as raízes da conceituação de modelo Entidade-Relacionamento (figura 4.1).

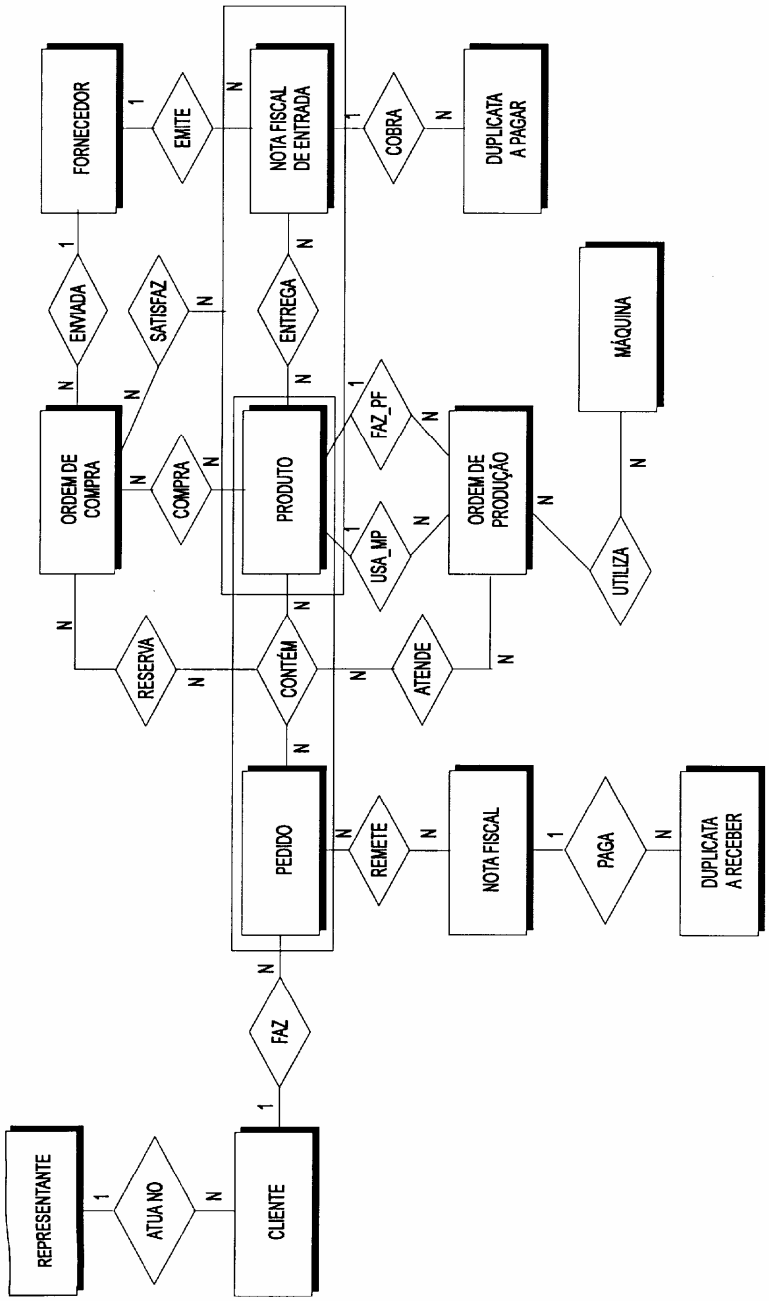


Figura 4.1 - Um Modelo E-R.

Quando Peter Chen formulou a proposta do modelo Entidade-Relacionamento, baseou-se não na visão de um sistema de aplicação como princípio e sim na compreensão da realidade em que se situava o problema. Como iremos projetar um sistema se não entendemos o negócio para o qual será realizado?

Chen dedicou-se a destacar a importância de reconhecer os objetos que compõem este negócio, independentemente de preocupar-se com formas de tratamento das informações, procedimentos, programas, etc. Estes objetos que desejamos conhecer e modelar para um sistema, Chen classificou em dois grupos: Entidades e Relacionamentos.

Na figura 4.2, é apresentado um fato comum que pode acontecer em qualquer realidade. Este fato deve ser retratado através de elementos básicos que compõem o modelo Entidade-Relacionamento.

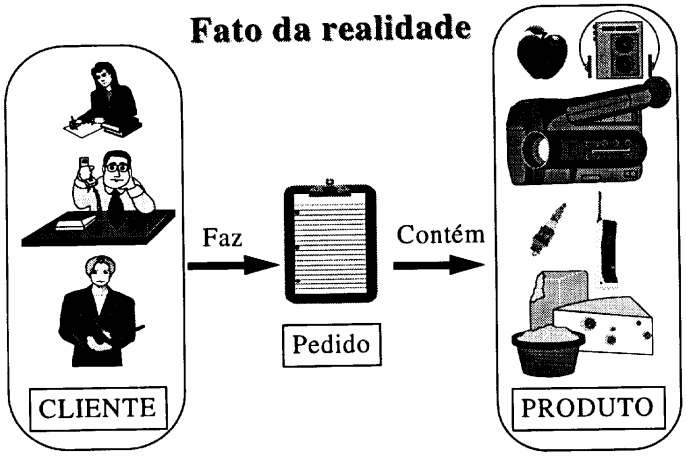


Figura 4.2

4.3 - Entidades

Define-se entidade como aquele objeto que existe no mundo real com uma identificação distinta e com um significado próprio.

São as "coisas" que existem no negócio, ou ainda, descrevem o negócio em si.

Se alguma "coisa" (figura 4.3), existente no negócio nos proporciona algum interesse em mantermos dados. (informações armazenadas sobre ele), isto a caracteriza como uma Entidade do negócio.

Esta entidade será então um conjunto de dados em nosso modelo conceitual.

É importante destacarmos aqui que uma entidade é a representação de uma Classe de dados do negócio, um conjunto de informações de mesmas características, e suas instâncias (ocorrências), são a representação destes dados.

Quando falamos sobre Classe de Dados, estamos na realidade trabalhando mentalmente um nível macro de informações, estamos atuando com abstrações interpretadas de acordo com o meio em que nos localizamos e seus interesses e objetivos organizacionais.

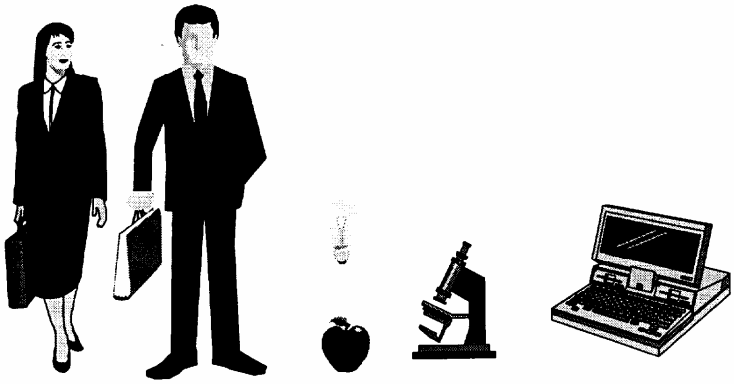


Figura 4.3

Para traçarmos um paralelo inicial com vistas a facilitar a visão do leitor e situá-lo em relação ao seu ambiente cultural normal, diríamos que a entidade é comparável ao arquivo de dados e suas instâncias são os registros deste arquivo. Mas esta é uma comparação que não é de todo real, já que estamos projetando bancos de dados, e logo tratando conjuntos de informações.

A representação de uma Entidade no modelo Entidade-Relacionamento se realiza através de um retângulo, com o nome desta entidade em seu interior, como na figura 4.4.

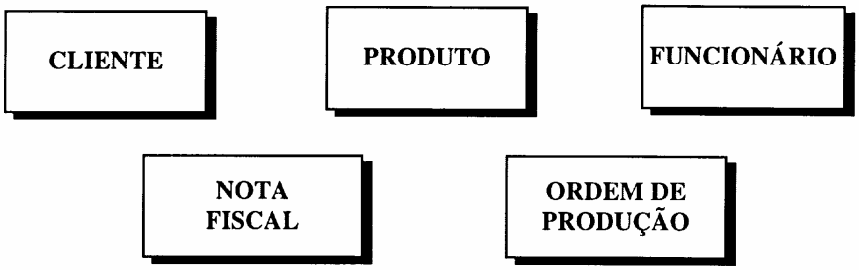


Figura 4.4 – Entidades.

As instâncias de uma entidade não são representadas no diagrama de Entidades e Relacionamentos, mas são semanticamente interpretadas no mesmo.

Devemos, ao visualizar uma entidade, entendê-la como uma tabela de dados, onde cada linha desta tabela representa uma instância da mesma. Mas como vamos detalhar mais este aspecto no tópico de atributos de uma entidade, fixemo-nos neste ponto para com-preendermos as "coisas" de um negócio.

Para que se possa absorver entidades devemos primordialmente nos orientar pelo mundo real, onde acontecem as coisas, buscando obtermos domínio do problema, enxergá-las sem a preocupação da construção de um sistema, sem imaginar programas, e sim exclusivamente preocupados em retratar uma realidade, compreender um negócio por seus dados.

Em nosso dia-a-dia, estamos frente a frente com diversas entidades, as quais processamos, selecionando instâncias que nos interessam.

Por exemplo, quando desejamos uma distração relaxante nos fins de semana, vamos até uma locadora de fitas de videocassete.

Vamos selecionar fitas para assistirmos em casa. Neste momento, estamos consultando a entidade "Filme" da locadora, que possui várias instâncias, ou seja, são as fitas disponíveis na locadora.

Quando desejamos obter o telefone de alguma pessoa, ou empresa, consultamos a entidade "lista telefônica oficial", que possui milhares de instâncias de telefones.

Em uma viagem de férias, utilizamos instâncias da entidade "mala", pois selecionamos algumas das nossas malas disponíveis para o tipo de viagem, quantidade de roupas, clima, etc.

Da mesma forma temos em um ambiente empresarial, diversos objetos setoriais e corporativos que caracterizam-se como entidades, com um número indeterminado de instâncias, como veremos a seguir.

Mas antes vamos conhecer como descrevemos uma entidade, ou melhor, os qualificadores (atributos) de uma entidade.

4.4 - Atributos

Todo objeto para ser uma entidade possui propriedades que são descritas por atributos e valores. Estes atributos e seus valores, juntos, descrevem as instâncias de uma entidade, formando o que no tópico anterior salientamos como registros em um arquivo.

Entidade: Funcionário

Matrícula	Nome	Data de Admissão
4456	João Carlos Silva	29/04/91
6689	Silvia de Oliveira	30/02/92
1203	Carla Martinez	14/04/92
7702	Pedro Guilherme Souza	01/01/92

Figura 4.5 - Entidade Funcionário.

Vamos considerar que em uma empresa, temos uma entidade, um objeto sobre o qual desejamos manter informações armazenadas, chamado Funcionário.

O que descreve Funcionário?

Funcionário é descrito por um número de matrícula, um nome deste funcionário, sua data de admissão, como é representado na tabela da figura 4.5, mas poderíamos ainda descrevê-lo com mais dados, tais como data de nascimento, valor de seu salário, etc. Estes dados que caracterizam o objeto funcionário são os atributos inerentes à entidade Funcionário.

Cada instância de Funcionário, cada existência de um objeto da Classe Funcionário, será formada por valores nestes atributos, sendo que é o

conjunto destes valores representados que devemos visualizar como uma linha de uma tabela de dados.

Os valores de determinados atributos ou de um determinado atributo, nas ocorrências desta entidade, são sempre diferentes para cada instância, caracterizando que não existem objetos repetidos dentro de uma classe de objetos, isto é, dentro de uma entidade.

Este(s) atributo(s) cujos valores nunca se repitem, sempre tem(têm) a função de atuar(em) como identificador(es) único(s) das instâncias da entidade. Dentro da abordagem relacional de banco de dados, denomina-se esta propriedade como Chave Primária de uma tabela, conceito este que iremos utilizar dentro de nosso contexto.

Entidade: Funcionário

Matrícula	Nome	Data de Admissão
4456	João Carlos da Silva	29/04/91
6689	Silvia de Oliveira	30/02/92
1203	Carla Martinez	14/04/92
7702	Pedro Guilherme Souza	01/01/92

Valores de chave primária de tabela:

4456	6689	1203	7702

Chave primária então é o atributo (ou conjunto de atributos concatenados) que identifica uma única ocorrência dentro de uma tabela.

Este conceito não foge de forma alguma ao que acontece no mundo real, já que um objeto sempre possui uma forma de identificação unívoca.

Este objeto não pode, ao modelarmos dados, possuir um identificador com valor nulo, pois um valor desconhecido impediria a identificação (existência) deste objeto.

- Como entender no mundo real um valor nulo?
  - Um valor nulo não deve ser visto de forma física, e sim de forma conceitual, como no mundo real.

    Por exemplo, se nos apresentarmos à frente dos leitores com este livro, em nossas mãos, qual é, neste instante, o valor do conteúdo do mesmo para os leitores que visualizam este fato?

    É desconhecido, pois não o abriram , não leram, logo o valor de seu conteúdo neste instante é nulo. (PS: Se você já leu até este capítulo, logo o valor do conteúdo já não é mais nulo para você, leitor, mesmo que não tenha gostado do que encontrou!)

- Colocam-se neste momento então, as seguintes questões:
  - Em que instante modelamos entidades?
  - Como devemos nos orientar para determinar as entidades?
  - Como ter certeza de que algo é uma entidade?

4.5 - Enxergando Entidades

Quando efetuamos uma atividade de levantamento de dados, estamos efetivamente identificando entidades ou classes de dados. Num primeiro contato com um negócio para o qual se efetuará um sistema de aplicação, podemos não possuir conhecimento especializado no mesmo, logo devemos procurar conhecer seus objetos principais.

A descrição dos objetos ou do objeto central do negócio irá nos apresentar a realidade retratada em diversas entidades. Por exemplo:

Uma clínica médica necessita controlar as consultas médicas realizadas e marcadas pelos médicos a ela vinculados, assim como acompanhar quem são os pacientes atendidos para manter o acompanhamento clínico dos mesmos.

Ao levantarmos os dados para a construção do sistema, nos foi informado que para cada médico a clínica mantém uma ficha com o número de CRM do médico, seu nome, endereço, especialidade etc.

Os pacientes preenchem um cadastro com dados pessoais tais como nome, endereço, data de nascimento, sexo etc. Toda consulta é registrada em fichário próprio com as informações sobre médico e paciente, diagnóstico etc.

Vamos então exercitar neste caso nossa visão de dados, de objetos neste contexto. Quais são os objetos candidatos a entidades?

Ora, o objetivo máximo desta clínica é a administração das consultas médicas. Caracteriza-se neste ponto a existência de um objeto de importância crucial no negócio: **A Consulta Médica**.

Mas a consulta médica possui atributos que a caracterizem efetivamente como uma entidade? Existirão várias ocorrências de consulta médica? Podemos representá-la sob a forma de uma tabela com linhas e colunas?

Obviamente, neste caso, a resposta as estas questões é afirmativa, o que permite-nos dizer que este objeto é efetivamente uma Classe de Dados, ou seja, é uma Entidade.

Mas o que descreve Consulta Médica?

Os atributos de uma consulta médica são aqueles que rapidamente comentamos no início do problema:

- Data de realização da consulta;
- Identificação do médico que realizou a consulta;
- Identificação do paciente que fez a consulta.

Podemos, agora, visualizar a entidade Consulta, que é um conjunto de instâncias que representam o fato, de um paciente se consultar com um médico, na forma de uma tabela de dados. É de extrema importância que se realize a simulação desta tabela (figura 4.6), com atribuição de valores para cada atributo, e com mais de uma ocorrência do objeto consulta médica.

Consulta Médica

Data_da_Consulta	CRM_do_Médico	Identificação_Paciente
22/04/92	21113	João Pedro Lima
22/04/92	21113	Clara Mathias
21/03/91	14442	Luís Alberto Conde
31/03/92	55555	Maria Luiza Andrade

Figura 4.6 - Entidade Consulta Médica.

Devemos continuar nossa análise, pois definimos um *pré-release* da entidade consulta, e ela não é obviamente a única no contexto, portanto consideremos então, que fomos informados de que a clínica mantém informações sobre médicos e também sobre seus pacientes.

Como existe mais de um médico na clínica, fato já colocado, mas sem especificação de um número desta existência, indeterminado, caracteriza-se a existência de uma entidade Médico no contexto, e da mesma forma a entidade Paciente. Já que irão existir muitas ocorrências destes objetos, logo existem as duas entidades (figura 4.7).

Vamos procurar então enxergar as tabelas de instâncias destas entidades para que possamos ter domínio e visão do conjunto de dados, eliminando-se assim o defeito de analisar uma situação com base em um registro.

Lembrem-se, leitores, que vamos utilizar sempre uma visão espacial de dados, vamos conhecer um nível de dado ainda abstrato, logo vamos enxergar o todo e não uma única ocorrência.

MÉDICO

CRM_Médico	Nome_Médico	Especialidade_Médico
21114	Luís Paulo Carvalho	Pediatria
21113	Pedro Estevão Poct	Ginecologia
51024	Maurício Abreu	Neurologia
76004	Simone Almeida	Cardiologia

PACIENTE

Nome_Paciente	Endereço	Sexo	Idade
Júlio Adamastor	R. Silva Sá, 23/11	Masc	33
Carmem Milhor	R. Dias Melhores 334/122	Fem	18
Sandra Chu Li	Av. Arapanés 4487/1915	Fem	44
Álvaro Medeiros Sá	R. Botica do Ouvidor 44/fundos	Masc	29
Paulo Alengui	Trav. das Camélias 661	Masc	38

Figura 4.7 - Entidades Médico e Paciente.

Não devemos considerar como entidade um objeto, se não conseguirmos ter a visão de seu conteúdo em instâncias com valores de atributos.

Isto equivale a afirmar que num primeiro momento, visualizamos sempre o macro objeto, uma abstração, e devemos obrigatoriamente, para nosso entendimento, procurar obter a visão de sua composição de ocorrências, sem a qual não poderemos, com certeza, afirmar sua existência, e fixá-la em nosso modelo.

O exemplo anterior caracteriza que uma entidade pode ser um objeto concreto, como médico e paciente, como também pode ser um objeto abstrato, um fato, um evento que desejamos registrar, e que possui características próprias no caso como a consulta médica (figura 4.8).

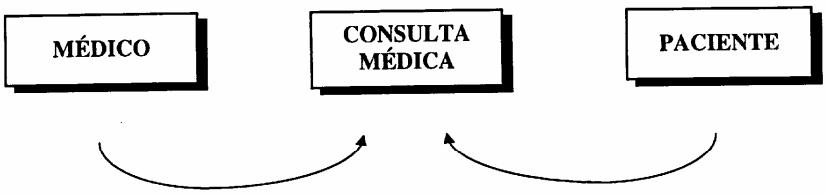


Figura 4.8

4.6 - Generalização e Especialização

É importante que quando estamos na busca da visualização dos dados de um negócio, nos atenhamos ao nível de abstração em que estamos atuando, pois quando definimos uma entidade, estamos com a visão de uma classe genérica de dados, que pode estar incorporando, implicitamente, diversas outras classes de dados.

Ou seja, existe um encapsulamento de informações sob a forma desta entidade genérica, a qual possui subconjuntos de dados que formam classes diferenciadas, mas que possuem características que nos permitem colocá-las sob a visão de uma única entidade.

Por exemplo, o caso que estudávamos da Clínica Médica.

A entidade médico é na realidade uma generalização para diversas classes de dados de médicos, tais como a figura 4.9 representa: Médico Pediatra, Médico Cardiologista e Médico Neurologista.

Se analisarmos superficialmente, poderíamos ter definido entidades para cada uma destas classes. Ou seja, existiriam as entidades específicas de cada um.

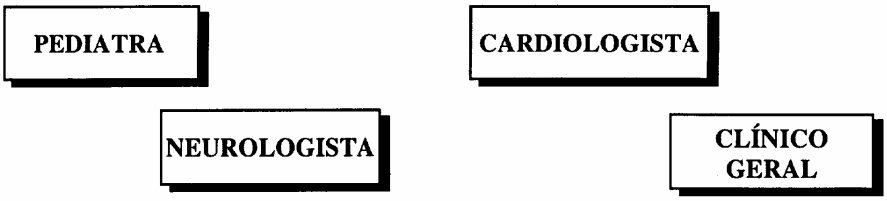


Figura 4.9

O que na realidade fizemos foi, através da colocação de um atributo qualificador, o qual permite a distinção entre cada classe de dados, generalizar todas estas classes em uma única, que denominamos de Médico.

Dentro do contexto da metodologia, alguns autores consideram que a entidade médico é uma entidade supertipo, mas esta nomenclatura não consideramos como ideal para expressar a realidade que estamos analisando.

Temos como regra então que quando encontramos entidades que possuem o mesmo conjunto de atributos para descrevê-las, podemos generalizá-las em uma única entidade, mantendo sua identidade de subconjunto através da inserção de um atributo qualificador para as ocorrências de cada uma.

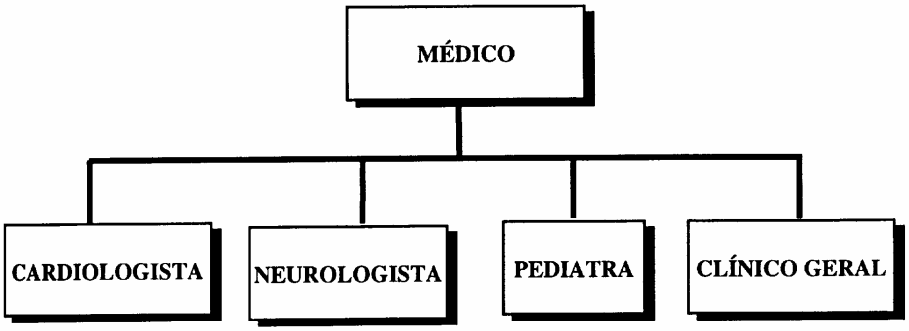


Figura 4.10

A esta qualificação por atributos que nos permitirá identificar um grupo, uma classe dentro da classe genérica, denominamos de Especialização.

Aparece neste instante um dos conceitos de visões de dados.

Visão de dados é a efetivação de um subconjunto de uma entidade (especialização) através da representação efetiva no diagrama de entidades e relacionamentos, de forma a permitir o tratamento e entendimento da formação de dados existente na realidade.

Dentro da abordagem relacional de banco de dados, este é na realidade o conceito de "role", ou "aliás", um indicador de que uma entidade possui entre suas ocorrências, algumas que representam um ou mais conjuntos de dados, com existência real definida.

No nosso exemplo até aqui utilizado, temos na entidade médico, um atributo qualificador que é "Especialidade_do_Médico".

A figura 4.10 apresenta como podemos representar a generalização/especialização num diagrama de entidades e relacionamentos, para o exemplo em questão.

Mas vejamos outros exemplo, para que possamos estender nossa capacidade de análise de entidades. Seja um modelo de dados que possua uma entidade Filme, como a locadora de fitas de videocassete que citamos anteriormente.

Como já salientamos, poderíamos ter a entidade Filme.

Observando a maneira como as pessoas selecionam as fitas que irão assistir, assim como as solicitam e questionam sobre sua existência, podemos observar então que existem dentro da entidade filme, subconjuntos que caracterizam-se como especializações desta entidade.

Nestes subconjuntos podemos visualizar: Filme-Comédia, Filme--Drama e Filme-Policial (figura 4.11).

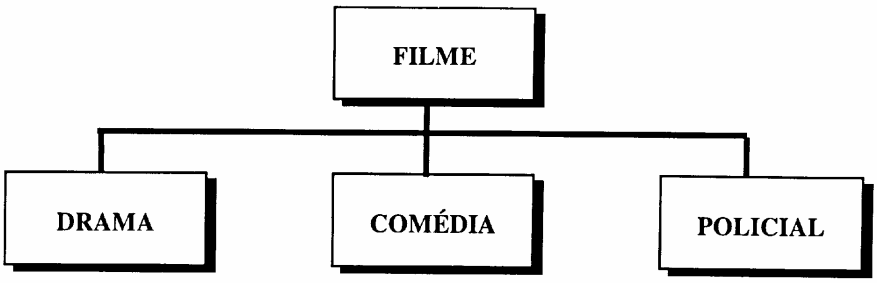


Figura 4.11

Poderíamos, em uma análise mais profunda, descer a um nível mais apurado de observação, em que estes subconjuntos especificados seriam na realidade também compostos por outros subconjuntos, tais como Drama Romântico e Drama Policial, ou ainda Comédia Policial e Comédia Romântica.

Existe, na realidade, uma hierarquia de classes de dados, encapsulada em uma entidade genérica.

- Mas por que a preocupação deste gênero?
- Quando ela se torna importante?

Ela é importante porque podemos vir a ter na análise funcional do sistema, tratamentos procedurais e diferenciados para cada subconjunto, assim como poderemos tratar simultaneamente todos os conjuntos.

Estes subconjuntos de uma entidade podem ser, inclusive, de naturezas completamente diferentes, mas importantes de conhecimento.

Seja uma entidade Pedido em um determinado sistema.

Esta entidade Pedido pode estar generalizando, por exemplo, os seguintes subconjuntos: Pedidos Pendentes, Pedidos Suspensos e Pedidos Atendidos.

Por outro lado os pedidos podem ser Pedidos Nacionais e Pedidos de Exportação.

Neste caso, temos que analisar se existem na entidade pedidos de dois tipos distintos de especialização, um por origem de pedido e outro por situação de pedido, sendo que os dois podem se superpor, tal como Pedido Nacional Pendente, ou Pedido Suspenso de Exportação.

Devemos representá-los de forma que possamos vir a tratá-los como um todo ou como parte do todo.

- Por exemplo:
  - Todos os pedidos suspensos;
  - Todos os pedidos nacionais suspensos;
  - Todos os pedidos nacionais;
  - Todos os pedidos pendentes de exportação.

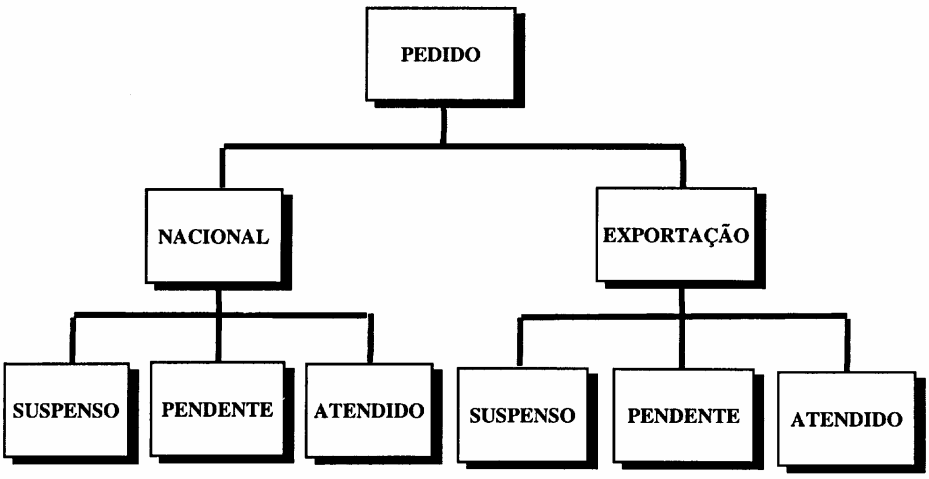
Na realidade, temos os subconjuntos do quadro:

Pedido Nacional	Pedido Exportação	
Pedido Suspenso	Pedido Pendente	Pedido Atendido

- Especialização em três níveis:
  - Pedido Nacional
  - Pedido Nacional Suspenso
  - Pedido Suspenso

A figura 4.12 nos mostra a representação deste caso dentro de um diagrama de Entidade-Relacionamento.

Sempre que existirem topologias em uma entidade, existirá de fato uma generalização/especialização.

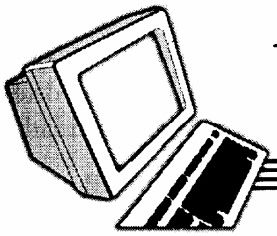


Devemos estar atentos ao executar o projeto conceitual, pois existem casos em que teremos entidades diversas com nomes distintos, mas que na realidade podem ser generalizadas em uma única, já que conceitualmente referem-se a um macro objeto, que por generalização pode absorvê-las integralmente.

Na prática, a generalização é efetivada quando o conjunto de atributos das entidades é comum em sua maior parte.

Concluindo, na generalização estamos colocando todas as instâncias de entidades diversas em uma única entidade, realizando seu tratamento como um todo, ou como parte quando necessário.

Capítulo



Relacionamentos

5

5.1 - A Existência

O entendimento sobre o que são efetivamente relacionamentos e a capacidade de enxergar estes objetos, como participantes do mundo real, são fatores primordiais para que se venha a efetuar trabalhos de modelagem de dados com compreensão do que está sendo realizando.

Não devemos, nunca, colocar temores de complexidade em uma técnica, e sim lembrarmo-nos de que a mesma nada mais é do que uma forma estruturada de representar as coisas que existem e ocorrem no mundo real. Procurando, sempre, retratar com simplicidade os fatos, isso os levará a representar com correção e entendimento.

Temos sentido nas diversas turmas de cursos que ministramos, um certo grau de dificuldade dos alunos em compreender corretamente relacionamentos, fato que leva a dedicar este capítulo para o perfeito entendimento do tema e sua utilização.

Dentro deste enfoque definimos RELACIONAMENTO como o fato, o acontecimento que liga dois objetos, duas "coisas" existentes no mundo real.

Considerando que estamos nos orientando para aplicações que serão desenvolvidas e administradas por um Sistema Gerenciador de Banco de Dados, poderíamos estender o conceito, principalmente para ambientes relacionais, como sendo relacionamento o fato que efetua a junção de duas ou mais tabelas de dados.

Entretanto, como estamos trabalhando em modelagem conceitual de dados, vamos nos privar de preocupações relativas a software ou o hardware, e nos concentrar em obter a visão dos dados de um determinado problema ou área de negócio.

Para um retrato dos objetos e fatos de um problema, os relacionamentos são os elementos que nos dão o sentido da existência destes objetos e suas inter-relações, sem as quais ficaria de extrema complexidade o entendimento e a compreensão do domínio do problema.

A figura 5.1 apresenta apenas dois objetos de um contexto qualquer, sendo: um homem de nome João e uma mulher de nome Maria.

JOÃO

MARIA

Figura 5.1

O que podemos entender deste contexto?

Temos apenas dois objetos (substantivos), soltos no espaço, torna-se evidente, até este instante, a inexistência de qualquer ligação entre os dois objetos.

Você já tentou se comunicar com as pessoas, explicar uma situação, contar uma história sem utilizar VERBOS em seu vocabulário?

É impossível nos fazermos entender, em qualquer situação, sem utilizarmos verbos, sem explicarmos o relacionamento entre as "coisas" sobre às quais nos referimos.

JOÃO CASADO COM MARIA

Figura 5.2

Na figura 5.2 com os mesmos objetos, apenas com a inclusão de um verbo entre eles, passamos a contar com um contexto de mais expressividade. Já podemos dizer que temos maior domínio (conhecimento) do ambiente.

A colocação do verbo, ou seja, do Relacionamento, deu semântica ao todo, as coisas já fazem sentido, não estão mais soltas, já existe um fato entre os objetos realizando uma ligação.

VERBO = A EXPRESSÃO DE UM FATO

No mundo que nos cerca, tanto em nossas atividades profissionais como nas atividade pessoais, convivemos com os mais variados tipos de entidades (objetos reais), que como já vimos, são descritos por uma série de atributos, e que expressam uma realidade de existência.

Estas entidades do dia-a-dia não estão soltas, desligadas umas das outras, e sim relacionadas de forma a mostrar a realidade com um conteúdo lógico.

Diariamente relatamos coisas do mundo real, e quando o fazemos estamos na verdade expressando entidades e relacionamentos, senão vejamos:

- As Pessoas Moram em Apartamentos;
- Os Apartamentos Formam Condomínios;
- Os Condomínios Localizam-se em Ruas, ou Avenidas;
- As Avenidas e Ruas Estão em uma Cidade.

Poderíamos seguir até o infinito do espaço sideral, mas um simples olhar para a figura 5.3 nos dá uma visão clara dos objetos, existentes no

mundo real, assim como as relações entre estes objetos nos dão domínio de conhecimento sobre um contexto específico.

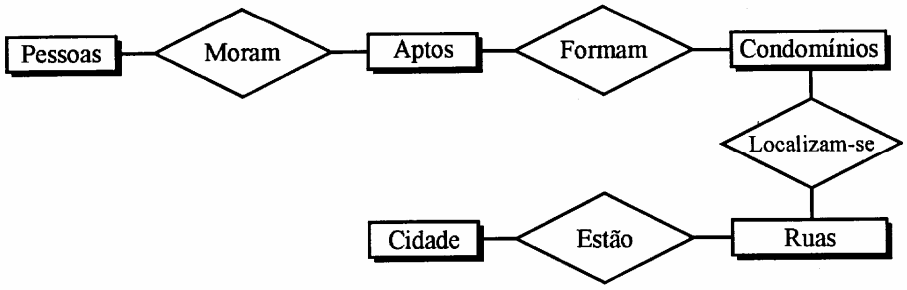


Figura 5.3 - Fatos de uma Realidade.

O que desejamos obter através da modelagem destes objetos, é exatamente a compreensão de contextos através dos dados.

Mas o importante é construir sistemas de aplicação com a certeza de que sabemos para que os construímos e com completo domínio do negócio para o qual o sistema foi projetado. Um sistema é sempre projetado para que venha a resolver um determinado problema de um negócio.

Este negócio, possui, seguramente, um grupo de objetos que representa os interesses da organização. Vamos detalhar então, como estes objetos se relacionam e de que formas estes relacionamentos existem no mundo real.

5.2 - Condicionalidade

Sejam duas entidades conforme a figura 5.4: Homens e Mulheres.

HOMEM
PEDRO LUÍS SÉRGIO CLÓVIS MARCELO CELSO CARLOS

MULHER
SÍLVIA CARLA ANTÔNIA ANDREIA CRISTINA MARIA ANA LÚCIA ANA PAULA ANA FLÁVIA

Figura 5.4

- Um Homem pode estar casado com duas ou mais mulheres ?
  - Depende do país onde estaremos realizando o projeto do sistema.
- Todas as mulheres são casadas?
- Todos os Homens são casados ?

Se estivermos retratando a realidade, as respostas às perguntas "Todos" evidentemente é "Não". Mas então como vamos entender a existência de relacionamentos, se existem elementos que não fazem parte, ocorrências das entidades que não estão se relacionando?

A questão das duas ou mais mulheres, iremos analisar quando tratarmos de cardinalidade de relacionamentos, logo adiante neste livro.

O fato de alguns elementos da entidade não acontecerem no relacionamento, em hipótese alguma indica a inexistência do fato, pois no mundo real o caso de homens não serem casados não elimina a existência do fato, do evento casamento, existindo homens casados e não casados em uma mesma entidade, sendo inclusive a participação no relacionamento (evento Casado), uma qualificação de um conjunto da entidade Homens.

Isto nos leva a dois grandes grupos de Relacionamento:

a) Relacionamentos Condicionais - relacionamentos que possuem uma condição, uma qualificação para ocorrerem, e;

b) Relacionamentos Incondicionais - não possuem esta condição, caracterizam-se por serem obrigatórios.

5.3 - Relacionamentos Condicionais

São efetivamente aqueles relacionamentos em que nem todos os elementos de uma entidade A estão ligados com elementos da entidade B. Dizemos que este tipo de relacionamento possui opcionalidade.

5.4 - Relacionamentos Incondicionais

Todos os elementos de uma entidade estão obrigatoriamente relacionados com um elemento, no mínimo, da outra entidade.

Neste caso, existe a obrigatoriedade do relacionamento entre todos os elementos de uma entidade com os elementos de outra.

Por exemplo, vejamos a figura 5.5 a seguir:

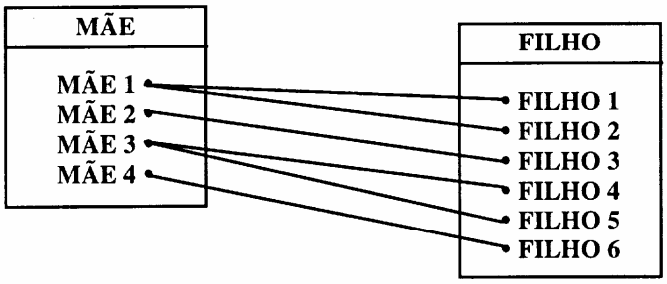


Figura 5.5

Toda ocorrência de mãe está relacionada a um ou mais filhos e; Toda a ocorrência de Filho está obrigatoriamente ligada a uma ocorrência de mãe.

Neste caso, não poderíamos ter em nenhuma hipótese, a possibilidade de existir na entidade filhos ocorrências que não estivessem ligadas a uma ocorrência da entidade mãe. E no sentido inverso também não poderíamos ter uma ocorrência em mãe que não estivesse ligada a um ou mais filhos, pois se permitíssemos, estaríamos distorcendo o mundo real, já que não existe uma mãe se não possuir filhos, e ninguém pode ser filho se não possuir uma mãe.

O relacionamento Tem da figura 5.6 é então um relacionamento Incondicional, pois é obrigatório para todos os elementos de mãe e todos os elementos de Filho.

Estes dois grupos de relacionamentos possuem cada um, vários graus de relacionamentos, que iremos estudar neste livro detalhadamente.

Agora neste ponto já conhecemos os dois elementos básicos do modelo Entidade-Relacionamento, vamos ver então qual a forma de representação gráfica do relacionamento, utilizada por Peter Chen, para que o modelo E-R mantivesse um grau de semântica que espelhasse efetivamente o mundo real.

Os relacionamento são representados por um losango entre as entidades e com arestas ligando as entidades a este losango. No interior do losango, inserimos um verbo que explicite o fato (o evento) que é o relacionamento (figura 5.6).

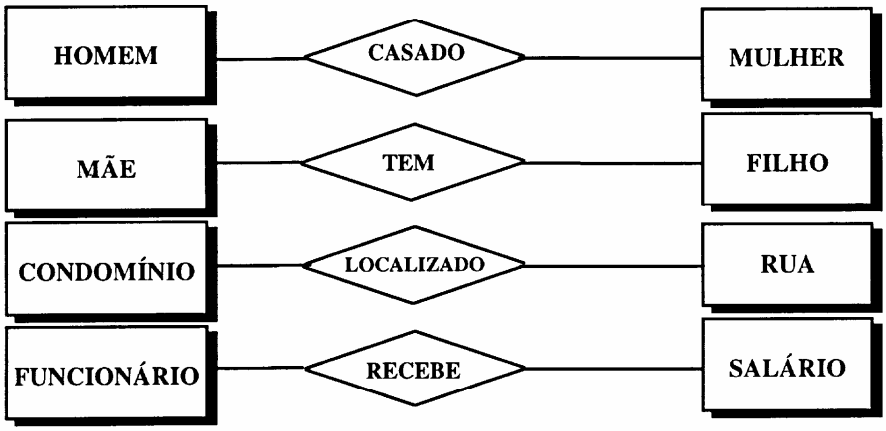


Figura 5.6 - Representação de Relacionamentos.

Na figura 5.6, temos então os relacionamentos:

- Homem Casado com Mulher;
- Mãe Tem Filho;
- Condomínio Localizado em Rua;
- Funcionário Recebe Salário.

5.5 - A Viagem

Mas voltemos ao nosso dia-a-dia, e vamos preparar as malas para uma viagem ao Caribe (quem sabe um dia!!).

Temos em casa um jogo de malas com tamanhos, cores e de materiais diferentes, com rodinhas, sem rodinhas, ou seja, uma infinidade de características para cada uma. Em nossas malas iremos colocar roupas, sapatos, produtos de higiene, etc, tudo aquilo que se leva quando se sai de férias.

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

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

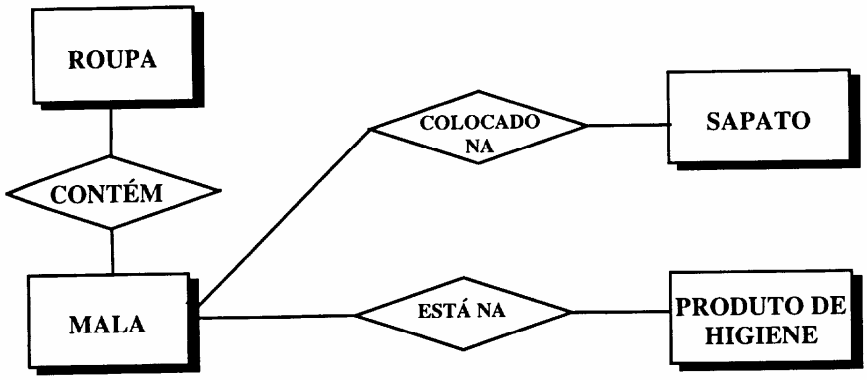
Vamos entender quais são os seus atributos básicos.

ENTIDADE:	ATRIBUTOS
Mala	Cor Volume Material Indicador de rodas Tipo de fechamento
Roupa	Cor da roupa Material (tecido) Descrição da roupa
Sapato	Cor Tipo (Esporte/Social) Marca
Produto de higiene e Beleza	Nome Marca

Se por acaso "um acidente de percurso" acontecer, e uma mala for extraviada ou perdida no trajeto Rio-Miami, e, se não houvesse uma forma de relacionar as coisas, como iremos saber o que estaria faltando de roupas, sapatos e produtos de higiene.

Existem casos em que as pessoas vão lembrar o que perderam ao extraviarem-se suas malas, muitos meses depois.

Logo, temos de criar relacionamentos entre as entidades envolvidas na viagem, que tenham o mesmo efeito de uma lista de coisas colocadas em cada mala, isto é na realidade uma visão dos dados com relacionamentos entre eles. O que iremos fazer é um modelo E-R, que irá representar com a mesma simplicidade da vida real este relacionamento (figura 5.7).



RELACIONAMENTO	LIGA
Contém	Roupa com Mala
Colocado na	Sapato com Mala
Está na	Produto de Higiene com Mala

Figura 5.7

Para a descoberta do relacionamento existe um tipo de, digamos, "Pulo do Gato", para após definidas as entidades de um sistema, descobrirmos onde acontecem relacionamentos, mas isto será estudado mais adiante.

Primeiro é importante estudarmos as chamadas Cardinalidades dos relacionamentos, ou Grau dos Relacionamentos dentro dos dois grupos principais.

Outro aspecto importante a ser considerado é o fato de conseguirmos enxergar em um diagrama de Entidades e Relacionamentos, os fatos que estão ali retratados, com a mesma simplicidade com que estes acontecem no mundo

real, o diagrama de Entidade e Relacionamento deve expressar os dados e eventos, sem teoréticas transcrições ou traduções de códigos, mas sim com expressão pura e simples da realidade dos fatos (figura 5.8).

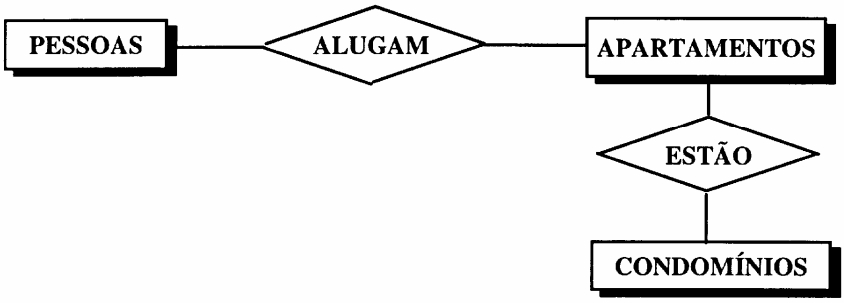


Figura 5.8

5.6 - Grau do Relacionamento

Quando temos um relacionamento entre duas entidades, o número de ocorrências de uma entidade que está associado, com ocorrências de outra entidade, determina o Grau do Relacionamento, ou Cardinalidade deste fato.

Quando questionamos anteriormente se um homem poderia estar casado com mais de uma mulher, estávamos na realidade ques-tionando o grau de relacionamento que existia entre as entidade Homem e Mulher.

O mundo real apresenta-se com três possibilidade de relacionarmos os dados, ou seja, três graus de relacionamento, que são:

5.6.1 - Relacionamento de Um-para-Um

Neste grau de relacionamento, cada elemento de uma entidade relaciona-se com um e somente um elemento de outra entidade.

No caso comentado acima, temos que uma ocorrência da entidade Homem relaciona-se com uma, e somente uma, ocorrência da entidade Mulher, pois o casamento no Brasil ainda é monogâmico (figura 5.9).

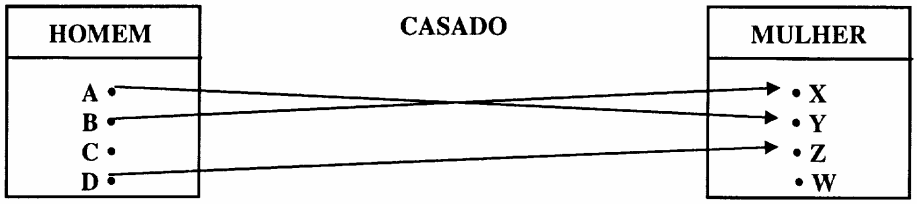


Figura 5.9 - Relacionamento um-para-um.

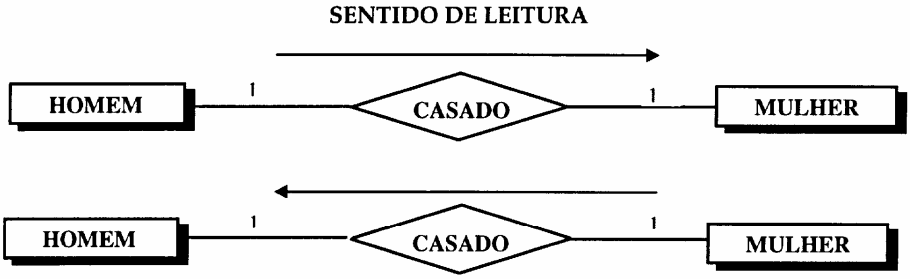
A figura 5.9 mostra um diagrama de instâncias onde dois objetos se relacionam com cardinalidade de um-para-um.

O elemento A da entidade 1 relaciona-se com o elemento Y da entidade 2 e somente com ele, não existindo nenhum outro relacio-namento entre A e qualquer outro elemento da entidade 2, que não o existente com Y.

É importante salientar que lemos o diagrama somente em um sentido, e isto está incorreto dentro do conceito de relacionamento, pois os mesmos não são unidirecionais.

Devemos ler o relacionamento nos dois sentidos em que ele se efetua. Logo leremos no caso da entidade Homem e da entidade Mulher, que um homem está casado somente com uma mulher e uma mulher está casada somente com um homem.

Muitos dos erros na construção de modelo de dados ocorem por serem realizados exames apressados sobre a cardinalidade dos relacionamento, efetuando-se a análise somente no sentido de um interesse específico do negócio, sem que se verifique a cardinalidade no sentido inverso.



IDÊNTICO RESULTADO = 1:1

5.6.2 - Relacionamento de Um-para-Muitos

Este grau de relacionamento é o mais comum no mundo real, sendo o que denominamos de relacionamento básico entre entidades, entretanto possui características específicas, quanto ao sentido de leitura dos fatos e sua interpretação.

Um elemento da entidade 1 relaciona-se com muitos elementos da entidade 2, mas cada elemento da entidade 2 somente pode estar relacionado a um elemento da entidade 1.

Ou seja, o grau de cardinalidade determinante sempre é o maior grau obtido da interpretação dos fatos.

Para ilustrarmos inicialmente este relacionamento, vamos nos deslocar até algum dos países árabes.

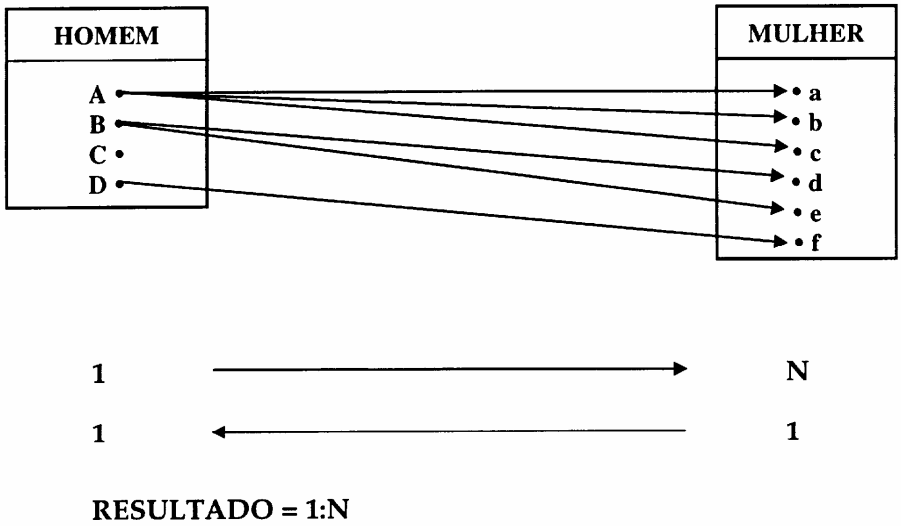


Figura 5.10 - Relacionamento um-para-muitos.

- Um homem é casado com muitas mulheres, mas a recíproca não é verdadeira, pois uma mulher é casada com um só homem (figura 5.10).

O fato de em um dos sentidos de leitura apresentar um grau Um-para--Um, não garante que o grau geral do relacionamento seja este.

Devemos ter como regra geral então, que um relacionamento é do tipo Um-para-Muitos, quando um sentido de leitura dos fatos nos apresenta este grau de Um-para-Muitos e o sentido oposto apresenta obrigatoriamente o grau Um-para-Um.

5.6.3 - Relacionamentos de Muitos-para-Muitos

Vejamos o exemplo da figura 5.11. Fácil visualização e entendimento, apresentando a associação entre a entidade Estudante e a entidade Disciplina (relacionamento Cursa). Nela também são mostradas as instâncias do relacionamento:

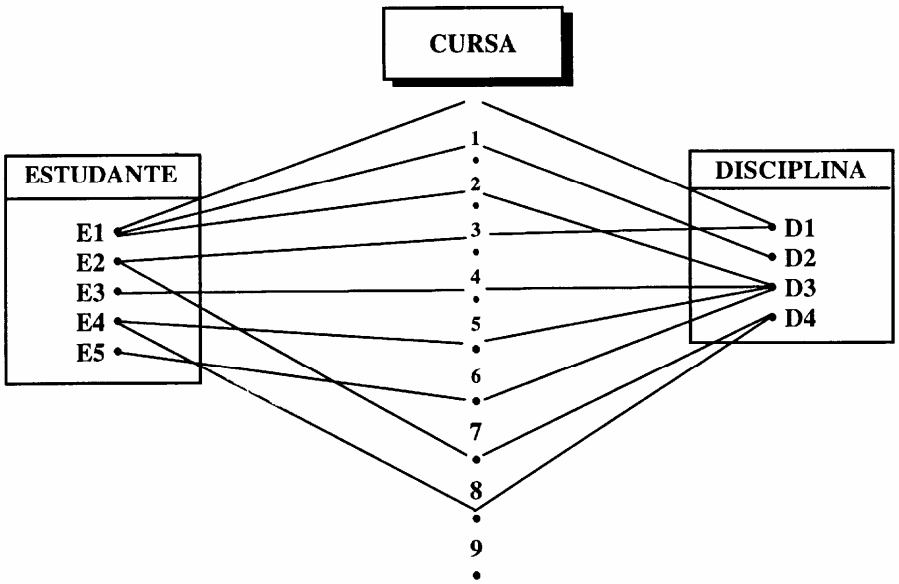


Figura 5.11 - Relacionamento muito-para-muitos.

- Um estudante cursa várias (muitas) disciplinas, mas alguns estudantes temporariamente podem estar cursando somente uma, ou nenhuma disciplina.
- Uma disciplina é cursada por vários (muitos) estudantes, mas eventualmente podemos ter uma disciplina que não possua nenhum estudante cursando-a, ou somente um. Neste caso, por haver opcionalidades, caracteriza um relacionamento condicional.

Identifica-se esta cardinalidade pelo fato de que em ambos os sentidos de leitura encontramos um grau Um-para-Muitos, o que caracteriza ser então um contexto geral de Muitos-para-Muitos.

Este tipo de relacionamento caracteriza-se por um aspecto que lhe é extremamente peculiar, ele possui atributos. Isto quer dizer que este relacionamento possui dados que são inerentes ao fato e não às entidades.

Em nosso exemplo anterior, poderíamos considerar que a data de matrícula, ou a turma onde o aluno cursa a disciplina, são atributos que descrevem ou informam algo sobre a associação das entidades, não sendo um atributo inerente a nenhuma das duas entidades em questão.

Não devemos nunca, ao pesquisar a Cardinalidade de um relacionamento, analisar somente uma ocorrência de associação entre os dados, pois no caso de relacionamentos condicionais, estas ocorrências podem não ser relacionadas, e mais, podem estar com cardinalidade somente de um-para-um eventualmente.

Para a descoberta da cardinalidade devemos analisar de forma macro a possibilidade de relacionamentos, sendo que a ocorrência de maior valor é que determina sempre o grau lógico do relacionamento.

Apresentamos à seguir a estrutura das entidades envolvidas na figura 5.11, e os atributos que são inerentes ao próprio relacionamento, ou seja, identificam ou qualificam o relacionamento:

ENTIDADE	ATRIBUTOS	RELACIONAMENTOS
Estudante	Nome do Estudante Matrícula do Estudante	com Disciplina 1:N
Disciplina	Código da Disciplina Nome da Disciplina	com Estudante 1:N

RELACIONAMENTO	ATRIBUTOS
Cursa	Data da Matrícula Turma

5.7 - A Modelagem dos Relacionamentos

Sabemos, por experiência adquirida ao longo dos anos e das salas de treinamento, que não nos basta conhecer as cardinalidades possíveis, ou seja, o grau dos relacionamentos. Existe o lado crítico do trabalho prático:

- Como saber qual o grau de um relacionamento que observamos no mundo real? E mais;
- Será mesmo um relacionamento, ou é um procedimento de sistema, que os vícios anteriores de análise procedural trazem a nós para tornar nebulosa a visão dos dados?

5.7.1 - A Descoberta dos Relacionamentos

Existem diversas situações em que vislumbramos turvamente um relacionamento.

Devemos sempre questionar se não estamos visualizando um procedimento.

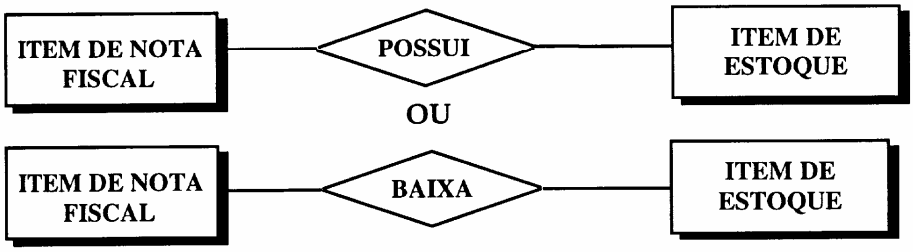


Figura 5.12

Certos verbos utilizados para expressar o relacionamento são tipicamente representações de procedimentos.

Seja então a situação da figura 5.12. Interpretando a entidade Item de Nota-Fiscal: ela possui em cada ocorrência, um objeto, que é um produto constante em alguma nota-fiscal.

A entidade Item-de-Estoque guarda em suas ocorrências os itens que compõem o estoque de produtos. Logo, temos então a seguinte formação de dados para o modelo:

ENTIDADE	ATRIBUTOS
Item de Nota Fiscal	Número da Nota Fiscal Código de Identificação do Produto Quantidade do Produto na Nota
Item de Estoque	Código de Identificação do Produto Descrição do Produto Quantidade em Estoque Valor Unitário do Produto

Quando estamos analisando o contexto de administração de estoques, muitas vezes a preocupação com os procedimentos nos leva a seguinte observação:

Quando enviamos uma nota fiscal, estamos também remetendo os itens desta nota fiscal, que são itens de estoque. Logo temos de dar baixa nos saldos em estoque. Esta baixa será realizada pela operação de subtração da quantidade do produto na nota fiscal, da quantidade em estoque do produto.

Muitas vezes temos encontrado modelos que embutem o procedimento no modelo como se ele fosse um relacionamento.

Então apareceria Item de Nota Fiscal Baixa Item de Estoque. Mas se nos permitirmos usar um verbo tão indicador de procedimentos no modelo, seguramente iremos cometer uma série de desvios que levará o modelo a ser concebido sob uma orientação de procedimentos, que levará, por sua vez, à construção de uma base de dados instável.

Figura 5.13

Entretanto se nos preocuparmos em somente fotografarmos os dados, procurando descobrir o que são e não o que é feito com eles, teremos o modelo da figura 5.13 que simplifica o entendimento:

- Um Item de Nota Fiscal está em Um Item de Estoque, ou,
- Um Item de Nota Fiscal Refere Um Item de Estoque, ou então,
- Um Item de Nota Fiscal Consta Um Item de Estoque.

O reverso sempre será obtido também, pois um Item de Estoque é referido por muitos Itens de Nota Fiscal

Uma leitura ampla do diagrama nos esclarece melhor a visão dos dados.

- Todo Item de Nota Fiscal É um Item de Estoque.

As ligações sugeridas dão maior semântica ao modelo e sua interpretação retrata a realidade dos dados com sua dinâmica.

5.7.2 - Como Testar se um Relacionamento Realmente Existe

Isto é na realidade uma tarefa, relativamente, complexa quando não temos de imediato uma estrutura de dados com informações comuns no primeiro instante de nossa análise.

Relacionamentos em um modelo podem surgir em função das exigências e necessidades de recuperação de informações por parte do usuário.

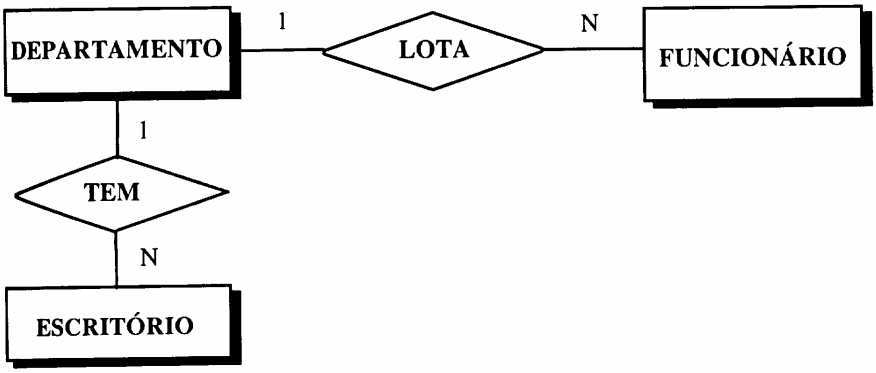


Figura 5.14

Seja o modelo Departamento Lota Funcionários e Departamento Tem Escritórios, da figura 5.14.

- Departamento lota 1 ou mais funcionários e um funcionário está lotado em um e somente um departamento.
- Um departamento possui 1 ou mais escritórios.

Para que estas instâncias aconteçam de fato, necessitamos que existam dados ou campos comuns às duas entidades:

Entidade:	**Atributos:**
Departamento	Código do Departamento Nome do Departamento Verba do Departamento
Funcionário	Código do Funcionário Nome do Funcionário Data de Admissão
Escritório	Número do Escritório Área do Escritório

Esta estrutura de dicionário possui os campos necessários à descrição de cada uma das entidades. Mas como efetivarmos o relacionamento?

Olhando a figura do E-R apresentada, vamos trabalhar o primeiro relacionamento, entre Departamento e Funcionário.

- Um Departamento tem Lotados muitos Funcionários;
- Um Funcionário está Lotado somente em um Departamento.

Sendo que se encontrarmos dois graus de relacionamento conforme o sentido de leitura, adotamos o grau maior como o efetivo do relacionamento.

Então, o lado que ficar com a cardinalidade N (muitos) deverá ter um campo, em sua estrutura, idêntico a um campo da outra entidade, o qual é chave primária nesta entidade.

Este é o conceito de chave estrangeira: um dado colocado em uma entidade que em outra é o identificador unívoco (chave primaria).

E isto é uma lógica normal das hierarquias naturais, senão vejamos:

A certidão de nascimento das pessoas que têm filhos, possui o nome dos filhos?

É óbvio que não. É a certidão de nascimento dos filhos que possui o nome dos pais.

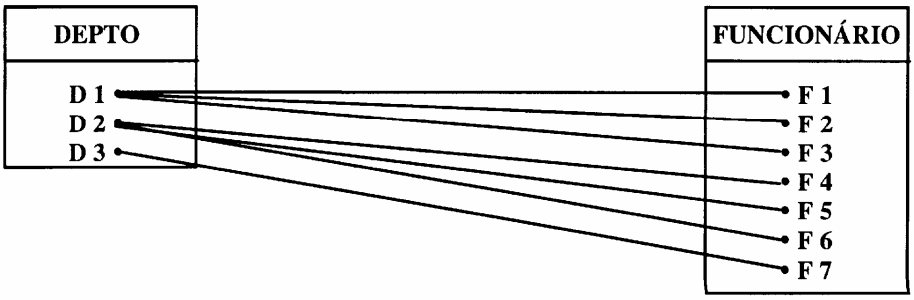
Resumindo, dependentes indicam de quem dependem.

Voltando então ao nosso caso, a estrutura de dados para Funcionário deve ser:

ENTIDADE	ATRIBUTOS	RELACIONAMENTOS
Funcionário	Código do Funcionário Nome do Funcionário Data de Admissão Código do Departamento *	Com Departamento 1:1 (Lotado)

* *Este campo exerce a função de Chave Estrangeira na entidade Funcionário.*

Vejamos então o diagrama de instâncias deste modelo de dados:



É importante sempre lembrar, que este campo utilizado como chave estrangeira, deverá ser na entidade referenciada (lado Um) correspondente à Chave Primária desta.

Bem, uma vez resolvido o relacionamento entre Departamento e Funcionário, temos ainda que resolver o relacionamento entre Departamento

e Escritório, já que é necessário recuperar, ou seja, consultar quais escritórios são de um determinado Departamento.

Seguindo o mesmo raciocínio lógico, uma vez que entre Departamento e Escritório existe uma ligação Um-para-Muitos, colocaremos a Chave Principal de Departamento na estrutura de Escritório.

ENTIDADE	ATRIBUTOS	RELACIONAMENTO
Escritório	Número do Escritório Área do Escritório * Código do Departamento	Com Departamento 1:1

* *Chave Estrangeira*

Temos assim em escritório uma referência lógica ao Departamento que ele pertence.

Poderíamos, na realidade, denominar estes campos inseridos na estrutura das entidades, do lado Muitos do relacionamento, como sendo "Pointers Lógicos", que nos indicam seus parceiros no relacionamento.

Mas ainda dentro de nosso modelo, devemos sempre validar se as necessidades de informações são possíveis de serem supridas no modelo resultante.

Por exemplo, se houver necessidade de obtermos a informação de onde trabalha o Funcionário, ou seja, em que Escritório trabalha um Funcionário, ou mais, que Funcionários trabalham em um determinado escritório.

Se observarmos os modelos, e mais a estrutura de dados que já obtivemos para este, concluímos que facilmente podemos saber o Departamento em que um Funcionário está Lotado, mas a partir de Departamento torna-se impossível descobrir-se em que escritório ele trabalha. Já que um departamento possui muitos Escritórios, o funcionário pode estar em qualquer um deles. Além do mais, a lógica do mundo real nos informa que em um Escritório deverão atuar muitos funcionários.

Este caso é na realidade, de solução muito fácil, pois basta-nos colocar o identificador único de escritório na estrutura de Funcionário, que estaremos estabelecendo condições para a existência de um relacionamento direto entre as duas entidades neste modelo.

ENTIDADE	ATRIBUTOS
Funcionário	Código do Funcionário Nome do Funcionário Data de Admissão • Código do Departamento • Número do Escritório

* *Identifica a Chave Estrangeira*

Após esta inserção na estrutura Funcionário, temos:

- Um Departamento Lota muitos Funcionários;
- Um Departamento Tem muitos Escritórios e em um Escritório Atuam muitos Funcionários.

Pode até parecer que estamos com um caminho redundante no modelo, no tocante a Funcionário Lotado Departamento Tem Escritório e Funcionário Atua Escritório, mas como já vimos, são fatos distintos, com graus de relacionamento distintos, o que nos permite recuperar a informação com mais segurança.

Se Funcionário está ligado a Escritório e Escritório está ligado a Departamento, será mesmo necessário ligar Funcionário a Departamento, ou pelo Escritório podemos saber o Departamento em que o funcionário está alocado?

Devemos considerar neste momento, que um Funcionário pode eventualmente estar atuando, exercendo suas atividades profissionais, em um Escritório que não pertence ao Departamento no qual ele está alocado.

Logo, não seria completa a visão dos dados se colocássemos restrições para viabilizar um modelo com menos relacionamentos.

O caminho de relacionamento entre Funcionário e Escritório é desta forma de grande valia para a obtenção de informações deste caso (figura 5.15).

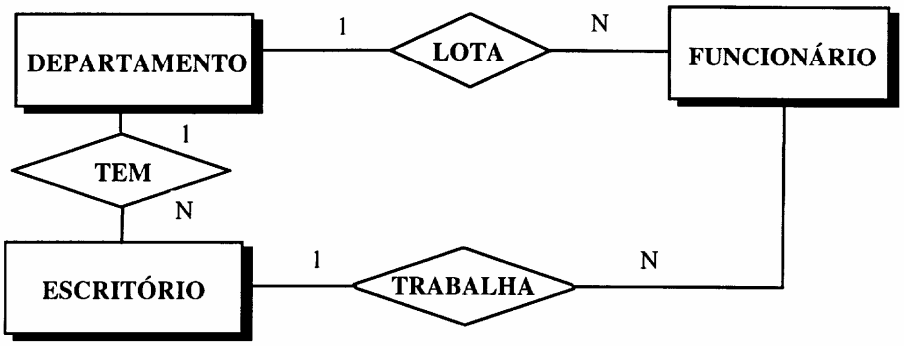
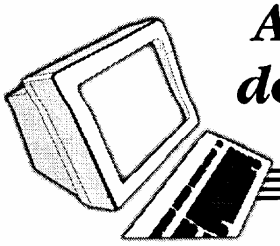

DEPARTAMENTO
1
LOTA
N
FUNCIONÁRIO
1
TEM
N
ESCRITÓRIO
1
TRABALHA
N

Figura 5.15

Capítulo



A Efetivação Lógica dos Relacionamentos

6

6.1 - A Expressão do Relacionamento

Apresentamos até este ponto a necessidade de incluirmos campos na estrutura de dados das entidades para que se efetuem os relacionamentos, ou seja, existem campos comuns para a ligação.

Quando um campo em uma entidade caracteriza-se por ser a chave de identificação única de ocorrências desta entidade, denomina-se, como já vimos, Chave Primária.

Quando em uma entidade temos um campo que é chave primária de outra entidade, denomina-se Chave Estrangeira.

Esta ligação realiza-se por comparação do valor da Chave estrangeira de uma talula com o valor da Chave Primária de outra tabela.

Se temos um Funcionário João e um Departamento Contabilidade, estes objetos somente estarão relacionados se:

- O valor do campo Código do Departamento na ocorrência de João da entidade Funcionário for igual ao valor do campo Código do Departamento da entidade Departamento na ocorrência Depto de Contabilidade.

Ora, isto nos fornece então uma expressão lógica, de comparação de valores, que explicita e estabelece uma regra para o relacionamento entre as duas entidades:

- **Código_do_departamento em Funcionário = Código_do_departamento em Departamento.**

Se desejarmos saber quais os funcionários de um departamento específico, bastará informarmos o valor do código deste departamento, para que sejam selecionadas todas as ocorrências de funcionário, cujo campo código do departamento seja igual ao valor informado.

Este é um processo de seleção, ou melhor, uma operação de seleção relacional.

Vamos observar as figuras das tabelas apresentadas a seguir, que simulam o conteúdo das duas entidades referidas.

Entidade: Departamento

Código Departamento	Nome Departamento	Verba
01	Contabilidade	500,00
10	Vendas	1.000,00
21	Faturamento	250,00

Entidade: Funcionário

Código	Nome	Data Admissão	Código Depto
0111	João	12/11/90	01
0112	Antônio	12/12/91	01
0271	Carlos	05/06/91	10
0108	Eduardo	03/03/90	10
0357	Luís	20/10/91	10
0097	Vera	15/02/92	21

Podemos então desta forma, responder às questões:

- Quais os funcionários do Departamento de Vendas ?
  - Código do Departamento de Vendas = 10
- Quais os Funcionários que têm Código de Departamento igual a 10?
  - Carlos; Eduardo e Luís.

Gostaríamos de salientar que temos colocado a preocupação de sempre nos orientarmos para recuperar informações combinadas, não dando ênfase neste momento à inserção, manutenção ou deleção de dados, aspectos que iremos comentar adiante neste livro, mas que são elementos que auxiliam na descoberta do contexto e sua consolidação.

Para um bom trabalho de modelagem devemos esquecer estas operações comentadas, preocupando-nos somente com os dados em si, não nos importando com procedimentos que serão inerentes ao sistema como um todo.

Na realidade, quando modelamos, não pensamos em sistemas, e sim em conseguir obter o entendimento de um negócio ou problema, estruturando os dados deste problema, com vistas ao seu domínio e sua solução.

Para que se solidifiquem os conceitos de uma técnica, não bastam apenas a apresentação de um exemplo de situação e sua aplicabilidade, mas bem pelo contrário, a massificação de casos analisados é que nos dará a possibilidade de ter segurança em nosso conhecimento adquirido.

A visualização de uma massa de casos tem por objetivo, além da solidificação de uma cultura, propiciar diversidade de situações de análise.

Entendemos que quanto mais situações passarmos com esta publicação, maior fonte de consulta o leitor terá para a sua vida profissional.

6.2 - Quando os Fatos podem Confundir-nos

O correto entendimento de uma informação depende muito da condição de interpretação dos fatos e da determinação da inerência do dado pelo analista de sistemas.

O saber interpretar o que um dado caracteriza, ou a quem este dado se refere é de suma importância para o resultado correto do modelo de dados.

Vamos então analisar uma situação em que se poderia ter interpretações errôneas da verdadeira caracterização que um dado efetua.

Figura 6.1

Em uma determinada empresa, são realizados diversos projetos de engenharia que alocam os funcionários disponíveis de seu quadro funcional conforme a necessidade, ficando estes funcionários alocados a somente um projeto até o encerramento do mesmo.

Uma vez alocado o funcionário a um determinado projeto, deve ser registrada a data de início de suas atividades no projeto, assim como o tempo em meses que o mesmo irá ficar alocado. O modelo E-R que representa este fato é o apresentado na figura 6.1.

Interpretando, ou melhor, lendo o diagrama que exprime o modelo de dados temos:

- Um Funcionário é Alocado a um Projeto e um Projeto Aloca muitos Funcionários.

Surge agora no tocante aos dois dados antes referidos, Data de Início e Tempo de Alocação, a dúvida de sua caracterização. Estes dados são inerentes ao fato Alocação, logo são campos, dados do relacionamento Alocado. Mas esta afirmativa está errada.

Seriam de fato, se um funcionário fosse Alocado a mais de um Projeto, mas como um funcionário é alocado a um projeto somente, estes dados são informações inerentes ao funcionário, pois possuem uma única existência para cada ocorrência de Funcionário, que esteja relacionado com o projeto.

Da mesma forma que na entidade Funcionário, para que se estabeleça o relacionamento com a entidade Projeto, necessita do atributo Código_do_Projeto em sua estrutura.

Temos então a seguinte estrutura de dados:

Entidade:	Atributos:	Relacionamento:
Funcionário	Matrícula do Funcionário Nome do Funcionário Endereço Código do Projeto Data de Início no Projeto Tempo Previsto de Alocação	Aloca N:1
Projeto	Código do Projeto Nome do Projeto	Aloca 1:N

Se a realidade colocada em análise fosse a de que um funcionário estivesse alocado a muitos projetos, esta seria uma informação do relacionamento entre Funcionário e Projeto, já que para cada associação do Funcionário com um Projeto, teríamos estes dados para caracterizá-la. Veremos mais a seguir um estudo deste tipo de relacionamento.

A figura 6.2 apresenta o diagrama de instâncias para o relacionamento Um-para-Muitos entre Funcionário e Projeto.

Propositalmente, colocamos no diagrama de instâncias, ocorrências em ambas as entidades, que não participam de nenhum relacionamento, ou seja, não estão relacionadas.

O funcionário F4 não está alocada a nenhum projeto, assim como o projeto-P3 não possui nenhum funcionário alocado.

Queremos dizer com isto, que a condicionalidade, ou melhor a opcionalidade de um relacionamento tem também o seu grau de importância no contexto, pois permite que uma ocorrência de projeto seja inserida na entidade Projeto sem que exista a necessidade, a obrigatoriedade de já possuir funcionários alocados antecipadamente.

Assim como a inserção de uma ocorrência em Funcionário, sem que esta esteja previamente relacionada com uma ocorrência em Projeto.

Na vida real, esta situação ocorre exatamente desta forma, pois em 99% dos casos em análise e projeto, os relacionamentos são condicionais no mínimo para um dos lados do relacionamento.

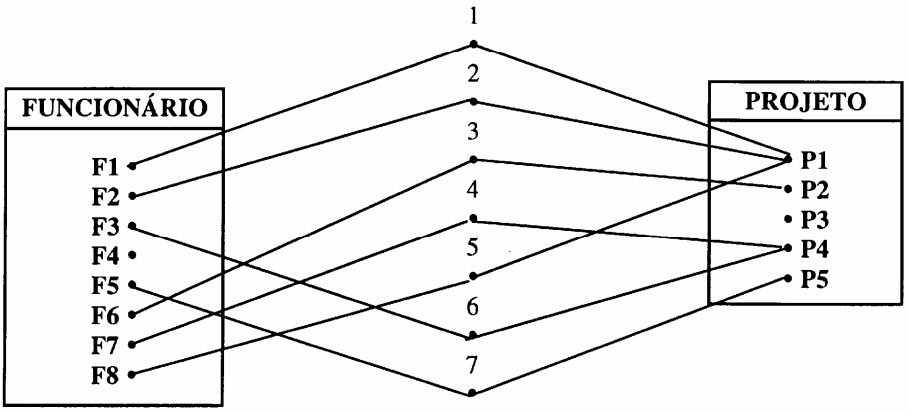


Figura 6.2

Que existem ocorrências de Funcionário que não estão alocados a nenhum projeto é fato concreto, mas como podemos visualizar este fato?

Vimos no capítulo 5 que uma característica do relacionamento um-para-muitos é o fato de ele necessitar da existência de chave estrangeira em uma das entidades.

Tomarmos como regra geral, que sempre que existir um relacionamento com cardinalidade de Um-para-Muitos, a referência lógica (chave estrangeira), estará colocada na entidade que possui o lado Muitos da cardinalidade.

Em nosso exemplo, a ligação da entidade Funcionário com a entidade Projeto será possível de se efetuar através da inserção do atributo Código_do_Projeto (Chave Primária de Projeto) na entidade Funcionário.

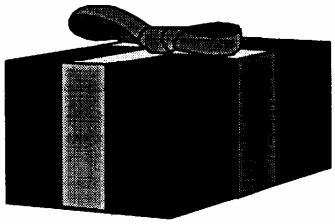
E como devemos entender a opcionalidade do relacionamento com relação aos valores dos atributos?

Para as ocorrências da entidade Funcionário que não estiverem relacionadas com nenhuma ocorrência de Projeto, o atributo também existirá porém, o valor deste atributo será NULO, isto é, uma informação desconhecida, inexistente.

6.3 - Valor Nulo

O desconhecido sempre nos assusta. Mas para entendermos o que é um valor nulo, o especialista Lívio Taufer apresentou, certo dia, uma forma simples de compreendermos este valor: - Vamos analisar uma situação típica de nosso dia-a-dia.

Se pararmos em frente a um supermercado, iremos observar diversas pessoas saindo deste estabelecimento comercial. A maioria carrega nas mãos sacolas ou pacotes fechados que denominaremos de Compras.



QUAL O CONTEÚDO?

Para nós que estamos ali parados, olhando a movimentação, qual o valor do conteúdo daquelas sacolas e pacotes? **Nulo**.

Ora, todos irão concordar que é desconhecido, ou seja, é Nulo, porque não sabemos o que existe dentro das sacolas e pacotes.

Então Nula é a informação desconhecida, é o dado não informado. Em Banco de Dados, atributo tem valor Nulo, quando este dado não é obrigatório de se informar, é opcional. Quando não informamos nenhum valor para ele, torna-se seu valor Nulo.

E as ocorrências de Projeto que não têm nenhuma ocorrência de Funcionário relacionada?

O valor da chave de identificação destas ocorrências (Chave Primária) obviamente não estará constando como chave estrangeira (valor do atributo Código_do_Projeto na entidade Funcionário) em nenhuma ocorrência da entidade funcionário.

A observação do diagrama Entidade-Relacionamento e dos atributos de cada objeto pode não ser suficiente para você leitor, adquirir a visão espacial dos dados.

Para uma melhor compreensão de conteúdo, torna-se necessário a utilização de uma simulação dos dados, criando-se as tabelas para as estruturas necessárias ao diagrama e colocando valores nos campos destas tabelas.

As tabelas a seguir apresentam a simulação das ocorrências das entidades Funcionário e Projeto.

Funcionário

Matrícula	Nome	Código do Projeto	Data de Início no Projeto	Tempo de Alocação
1466	Pedro Antônio	P-25	12/12/91	16 meses
2322	Luiz Paulo Diniz	P-18	05/01/92	4 meses
7712	Carlos Estevão	NULO	NULO	NULO
4415	Sílvia Cristina	P-18	18/04/92	5 meses

Projeto

Código do Projeto	Nome do Projeto
P-18	Projeto Phoenix
P-25	Projeto Minerva
P-32	Projeto Corrup (Cruzes!!)
P-55	Projeto Nova Ponte
P-203	Orçamento 95

Existem ocorrências que justificam a cardinalidade Muitos-para-Um, independentemente de existirem ocorrências com cardinalidade Um-para--Um.

Na tabela de instâncias de Funcionários apresentada, existe um Projeto que só é referenciado por uma ocorrência de Funcionário. Somente o funcionário Pedro Antônio está com código de projeto P-25, mas como podemos perceber, existe pelo menos um caso de um projeto estar tendo seu valor de Código_do_projeto referenciado em mais de uma ocorrência de Funcionário, o que nos é suficiente para estabelecer esta cardinalidade de Muitos-para-Um, entre Funcionário e Projeto.

A tabela que representa a entidade Projeto, se agora analisada, em conjunto com a tabela de instâncias de Funcionário, possibilita a visualização da parcialidade do relacionamento entre as entidades.

Existe na entidade Projeto uma ocorrência (projeto p-32) que não está sendo referenciada por nenhuma ocorrência da tabela da entidade Funcionário, assim como a entidade Funcionário, possui uma ocorrência que têm valor nulo para os atributos relativos ao relacionamento com a entidade Projeto: Código_do_Projeto, Data_de_início, e Tempo_de_Alocação, é a ocorrência do funcionário Carlos Estevão.

É preciso, pelo apresentado nos exemplos, que tenhamos uma massa crítica de dados para simularmos as ocorrências dos mesmos no mundo real, para que a definição de cardinalidades seja realizada, com segurança.

Agora vamos estudar outra situação neste mesmo caso.

Se por exemplo, neste caso, a empresa informa que um funcionário pode atuar em mais de um projeto, ou melhor, está alocado a mais de um projeto simultaneamente.

Vamos ver como ficaria o nosso modelo de dados, quais alterações iria sofrer. Teríamos um relacionamento do tipo Muitos-para-Muitos, conforme o diagrama da figura 6.3.

Figura 6.3

Como já estudamos na definição de relacionamento com cardinalidade Muitos-para-Muitos, este tipo de relacionamento tem a característica de possuir campos para representá-lo, informações inerentes ao fato, ao evento, que é o relacionamento, sendo no mínimo dois campos, que são as chaves primárias das entidades relacionadas.

Neste caso em estudo, os dados Data_de_início_no_Projeto e Tempo_de_Alocação_no_Projeto, são informações relativas ao evento que une Funcionário a Projeto, e como pode existir mais de um evento destes para cada ocorrência de Funcionário, assim como de Projeto, estes dados deixam de ser únicos por ocorrência de Funcionário, passando a ser múltiplos.

Como o número de ocorrências do relacionamento é indeterminado, não podemos mantê-lo como atributo de Funcionário, pois não saberíamos quantas ocorrências colocar destes atributos em Funcionário, sendo necessário o desdobramento.

Logo, o relacionamento Alocado, passa a ter uma existência física, ou seja, uma tabela o implementa.

As estruturas de dados correspondentes ao modelo figura 6.3 ficam assim delineadas:

Entidades:	**Atributos:**
Funcionário	Matrícula_Funcionário Nome_Funcionário
Projeto	Código_Projeto
	Nome_Projeto

Relacionamento:	**Atributos:**
Alocado	Matrícula_Funcionário Código_Projeto Data_Início_no_Projeto Tempo_de_Alocação

6.4 - Como se Efetiva este Relacionamento?

O Relacionamento efetiva-se através de uma expressão relacional que indica como deve ser feita a comparação entre os campos comuns às Entidades, só que agora com uma característica diferente: a comparação é realizada entre campos das entidades e campos do relacionamento, formando uma expressão composta.

Expressão de Relacionamento:

- **Funcionário.Matrícula-funcionário = Alocado.Matrícula-Funcionário e Alocado.código-projeto = Projeto.código-projeto.**

Esta expressão quer nos dizer que o valor do campo matrícula na entidade Funcionário deve ser igual ao valor do campo matrícula no relacionamento Alocado, e que o valor do campo Código_do_Projeto no

relacionamento Alocado deve ser igual ao valor do campo Código_do_Projeto na entidade Projeto, conjuntamente.

Quando isto acontecer com uma ocorrência de Funcionário, uma ocorrência de Alocado e uma ocorrência de Projeto, estaremos relacionando as duas entidades que são Funcionário e Projeto.

Vamos então visualizar estes fatos na simulação das tabelas relacionais que representam esta realidade.

Funcionário:

Matrícula	Nome	Data_Admissão
1466	Pedro Antônio	12/05/90
2322	Luiz Paulo Diniz	18/06/91
7712	Carlos Estevão	24/05/90
4415	Silvia Cristina	05/05/91
1216	Sandra Chi Min	01/02/92
1401	Maurício Abreu	15/05/92

Projeto:

Código_do_Projeto	Nome_do_Projeto
P-18	Projeto Phoenix
P-25	Projeto Minerva
P-32	Projeto Corrup (Cruzes!!)
P-55	Projeto Nova Ponte
P-203	Orçamento 95

Vamos então simular relacionamentos Muitos-para-Muitos com estas duas tabelas de ocorrências das entidades, criando o relacionamento Alocado, e suas possíveis ocorrências.

ALOCADO:

Matrícula Funcionário	Código_do_Projeto	Data_Início_ no_Projeto	Tempo_de_Alocação no_Projeto
1466	P-18	24/05/90	24 MESES
1466	P-25	12/11/91	06 MESES
1466	P-32	02/01/92	12 MESES
7712	P-18	10/06/91	04 MESES
7712	P-79	12/12/91	12 MESES
4415	P-18	15/01/92	03 MESES
1216	P-25	01/03/92	05 MESES

Vamos interpretar o mundo real, através da tabela de ocorrências do relacionamento Alocado:

- A ocorrência de Funcionário com matrícula 1466 está alocada a três (03) Projetos, respectivamente P-18, P-25 e P-32, isto é, Um Funcionário Alocado a Muitos Projetos;
- A ocorrência de Funcionário com matrícula 7712 está também alocada a muitos projetos (dois);
- Já as ocorrências de funcionário de matrícula 4415 e a de matrícula 1216 estão cada uma alocada a somente um projeto, pois só constam uma vez dentro do relacionamento com campos Alocados.

Novamente lembramos que o fato de existirem ocorrências relacionando-se com a cardinalidade Um-para-Um não invalida a cardinalidade básica do relacionamento, uma vez que possuímos ocorrências que realizam a cardinalidade Muitos-para-Muitos.

É sempre muito importante que se efetue a leitura do modelo de dados em dois sentidos, para compreensão perfeita da realidade. Então vamos agora analisar a situação por outro sentido de leitura do relacionamento:

- O projeto de código P-18 possui Muitas ocorrências de Funcionário a ele alocadas, ou seja, respectivamente 1466, 7712 e 4415;
- Assim como o projeto de código P-25 possui também muitas ocorrências de Funcionário a ele relacionadas (1466 e 1216);

- Já os projetos P-32 e P-79 possuem somente uma ocorrência de Funcionário a eles relacionada.

Observem que interpretamos, ou melhor, realizamos a leitura pura e simples da tabela que representa este relacionamento, não considerando ainda neste instante, as ocorrências das duas entidades que não figuram no relacionamento. Estas ocorrências são irrelevantes para a interpretação do Relacionamento.

Observem o diagrama da figura 6.4, para que se estabeleça uma regra formal para determinação de um relacionamento Muitos-para-Muitos.

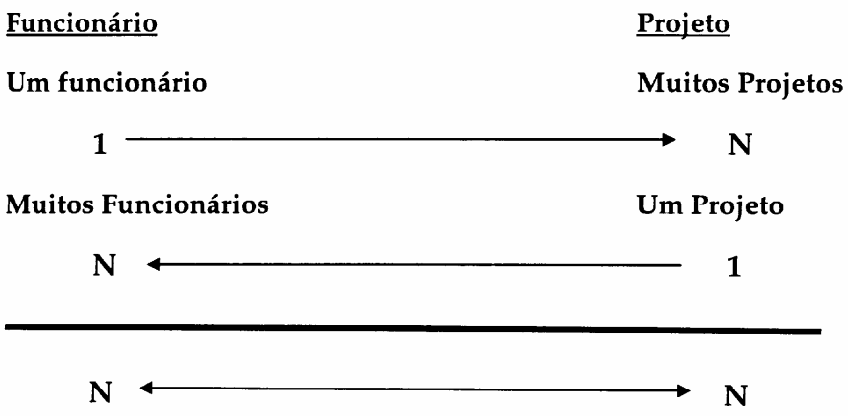
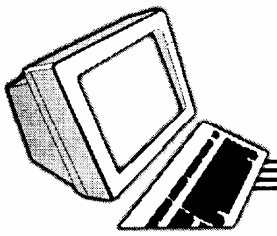


Figura 6.4

Capítulo



Relacionamentos Especiais

7

7.1 - Relacionamentos entre Múltiplas Entidades

Até o momento, neste livro, estudamos e analisamos situações em que as entidades se relacionavam aos pares. Este é o princípio de descoberta dos relacionamentos na construção de um modelo de dados: analisar as entidades aos pares.

Entretanto um relacionamento pode existir envolvendo mais de duas entidades, que podem ser três, quatro, ou uma quantidade indeterminada de entidades que o fato do relacionamento envolve.

Os relacionamentos entre múltiplas entidades expressam um fato em que todas as entidades ocorrem simultaneamente, ou seja, todas as ocorrências do relacionamento possuem, sempre, ligações com todas as entidades envolvidas no relacionamento. Não pode existir de um relacionamento triplo, em um determinado momento, se transformar em duplo.

Vamos observar então o diagrama da figura 7.1 que apresenta uma situação de relacionamento ternário que envolve três entidades simultaneamente.

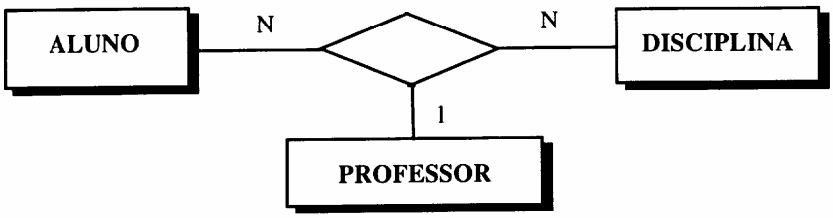


Figura 7.1

Para podermos descobrir as cardinalidades do relacionamento ternário da figura 7.1, devemos proceder da seguinte forma:

- Separar a entidade ALUNO e analisar o par PROFESSOR, DISCIPLINA. Para cada par PROFESSOR / DISCIPLINA podemos ter de **1** até **N** ALUNOS relacionados;
- Separar a entidade PROFESSOR e analisar o par ALUNO, DISCIPLINA. Para cada par ALUNO / DISCIPLINA podemos ter **1** e somente **1** PROFESSOR relacionado;
- Separar a entidade DISCIPLINA e analisar o par PROFESSOR, ALUNO. Para cada par PROFESSOR / ALUNO podemos ter de **1** até **N** DISCIPLINAS relacionadas.

Relacionamento múltiplo com mais de quatro entidades relacionadas é extremamente difícil de se encontrar na realidade do dia a dia. Quando você encontrar com algum fato que dê origem a um relacionamento múltiplo, analise com cuidado, pois o mesmo pode ser desmembrado em mais de um relacionamento. A implementação de relacionamento múltiplo em bancos de dados torna o trabalho de manipulação bastante complexo.

7.1.1 - Relacionamentos Múltiplos Muitos-para-Muitos

No diagrama da figura 7.1, temos a seguinte realidade descrita:

- Quando um aluno está matriculado em uma disciplina, este tem sempre um professor;
- Um aluno pode estar matriculado em várias disciplinas;
- Uma disciplina tem vários alunos, e somente um professor;
- Um professor leciona uma disciplina para vários alunos.

Este relacionamento é ternário, pois envolve três entidades simultaneamente.

Observem que quando ocorrências de duas das entidades se relacionam, obrigatoriamente relaciona-se a elas uma ocorrência da terceira entidade.

Isto coloca características interessantes e restritivas na leitura deste relacionamento, senão vejamos:

- Um aluno em uma disciplina, "sempre" tem um professor;
- Uma disciplina com alunos, "sempre" tem um professor;
- Um professor com aluno, "sempre" tem uma disciplina.

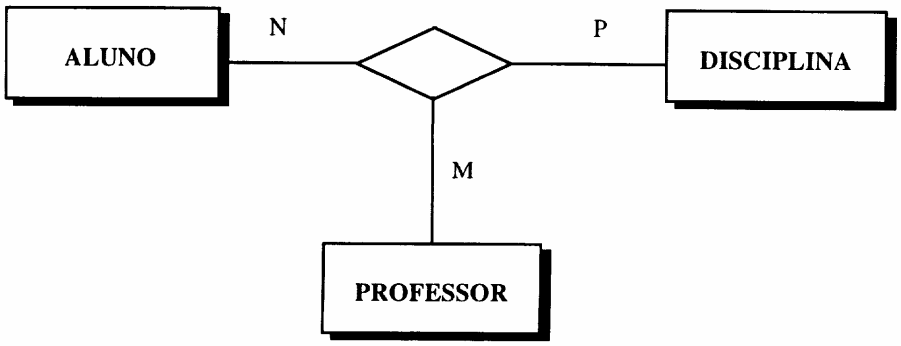


Figura 7.2

O exemplo da figura 7.2 possui uma cardinalidade Muitos-para-Muitos, tendo a terceira entidade envolvida, uma cardinalidade de participação neste relacionamento também de muitos.

Vamos analisar o diagrama de instâncias deste relacionamento (figura 7.3), para que se possa melhor entendê-lo, e após construir as tabelas com dados para as entidades e os relacionamentos.

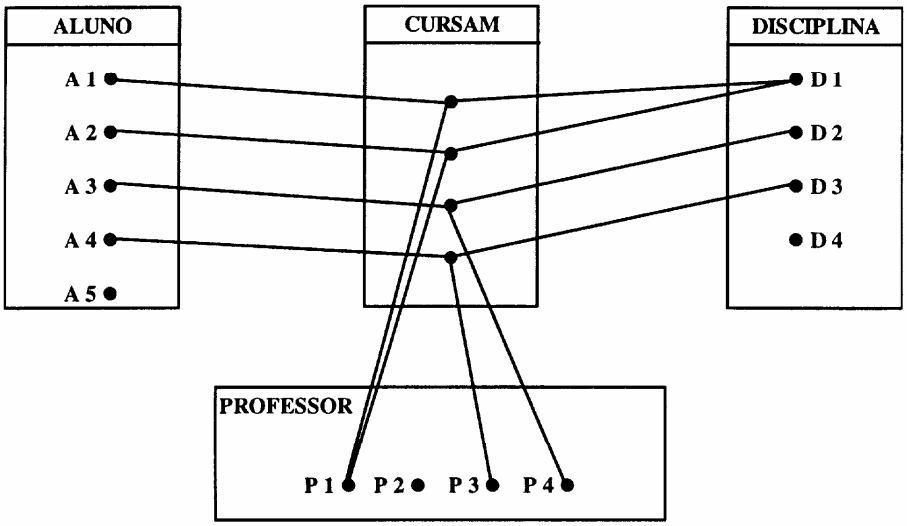


Figura 73

Colocamos no diagrama de instâncias, ocorrências em cada uma das entidades envolvidas que não estão participando do relacionamento.

A interpretação correta deste tipo de relacionamento nos apresenta um ponto de interseção das três ocorrências das entidades envolvidas em cada evento de relacionamento, caracterizando uma tabela (CURSAM) para representar este tipo de relacionamento, que tem então cardinalidade de Muitos-para-Muitos.

Teríamos neste caso, as seguintes estruturas para as entidades e o relacionamento:

ENTIDADE:	ATRIBUTOS:	CONEXÕES:
ALUNO	Número_do_Aluno Nome_do_Aluno Data_da_Matrícula	Com CURSAM 1:N Parcial
DISCIPLINA	Código_Disciplina Nome_Disciplina	Com CURSAM 1:N Parcial
PROFESSOR	Código_Professor Nome_Professor	Com CURSAM 1:N Parcial
Cursam	Número_do_Aluno Código_Professor Código_Disciplina	com Aluno N:1 com Professor N:1 com Disciplina N:1

Vamos observar as tabelas que simulam estes objetos do relacionamento ternário:

Professor (Entidade) **Aluno (Entidade)**

Código_Professor	Nome_Professor	Número_do_Aluno	Nome_do_Aluno
12	Antônio Furtado	120	Carlos Antônio
11	M. de Medeiros	122	Luís Carlos
14	Juarez Antônio	123	Silvia Regina
45	João Clóvis	124	Irene Maria
66	Celso Bressany	200	Pedro Luís

Cursam (Relacionamento) **Disciplina (Entidade)**

Código_Professor	Número_Aluno	Código_da Disciplina	Código_da Disciplina	Nome_Disciplina
14	120	D24	D24	Matemática I
14	123	D24	D55	Física Aplicada
14	122	D24	D66	Laboratório Física II
11	200	D27	D99	E. P. Brasileiros II
11	122	D27	D27	Cobol I (Uuhgh!!)
66	120	D99		
66	123	D99		
45	120	D66		

Observando a tabela de dados do relacionamento **Cursam**, podemos ver que existem ocorrências de Aluno que não figuram no relacionamento, assim como existem ocorrências de Professor que também não figuram, e igualmente Disciplina, colocando a opcionalidade no relacionamento em relação às ocorrências de cada entidade.

No entanto podemos observar que sempre que existe uma ocorrência no relacionamento, esta apresenta referência às três entidades, não existindo, por exemplo, nenhuma ocorrência somente com professor e disciplina.

7.1.2 - Relacionamentos Ternários Um-para-Muitos

No diagrama da figura 7.4, é apresentado um relacionamento de Um-para-Muitos entre Funcionário, Projeto e Máquina.

Este relacionamento está retratando a seguinte realidade:

Um funcionário trabalha somente em um Projeto, e um Projeto pode ter somente um funcionário trabalhando nele. Quando um funcionário trabalha em um projeto, ele tem um Supervisor. Um Supervisor pode supervisionar muitos funcionários que estão trabalhando em algum projeto.

Vamos estudar com carinho este caso, já que está nos apresentando uma cardinalidade de Um-para-Um entre duas das entidades, e de Muitos-para-Um em relação à terceira entidade envolvida.

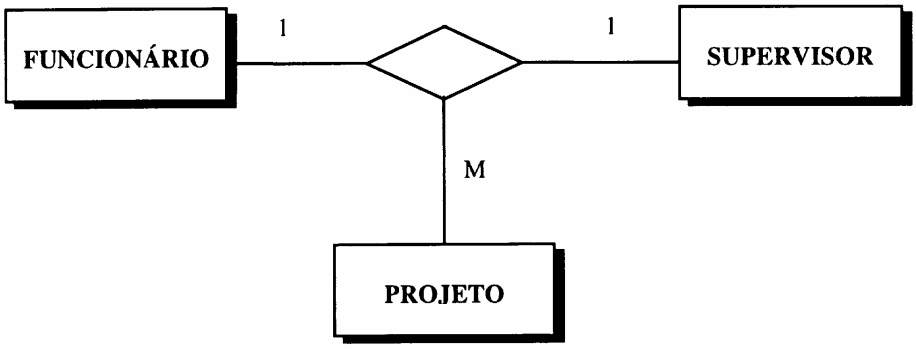


Figura 7.4

Devemos atentar para a realidade que está sendo retratada, ou seja, um supervisor supervisiona as atividades de um funcionário no projeto, e não supervisiona o projeto em si. Então:

- Luís está alocado no Projeto Ponte do Rio Caí e tem como supervisor Sílvio;
- Carlos Alocado no Projeto Reforma de Viadutos tem como seu Supervisor Antônio Pedro;
- Antônio Pedro é supervisor de Carla no Projeto Brasília ("mais um rumo ao Planalto").

Resumindo, decompomos o relacionamento em dois fatos, ou seja, um supervisor controla as atividades de muitos funcionários alocados em projetos.

Não somos partidários da utilização de relacionamentos múltiplos, por considerar que eles não espelham claramente uma realidade, já que como podemos observar, em um relacionamento desta natureza temos sempre dois fatos envolvidos.

Vamos visualizar como ficam as estruturas de dados destes objetos para este exemplo.

ENTIDADE	ATRIBUTOS
Funcionário	Código_Funcionário Nome_Funcionário Código_Projeto Código_Supervisor
Projeto	Código_Projeto Nome_Projeto
Supervisor	Código_Supervisor Nome_Supervisor

RELACIONAMENTO	EXPRESSÃO
Trabalham	Funcionário.Código_Projeto= Projeto.Código_Projeto Funcionário.Código_Supervisor=Supervisor.Código_Supervisor

A expressão de relacionamento, por ser composta, explicita a obrigatoriedade da existência de três elementos no relacionamento.

7.2 - Relacionamentos Binários de Um-Para-Um

Os relacionamentos de cardinalidade Um-para-Um existem efetivamente no mundo real. A questão é: Como implementá-los através de referências lógicas?

Este tipo de relacionamento tem sua existência sempre questionada, pois é seguidamente resultante de um erro de interpretação do contexto das entidades existentes em um modelo, devendo por isto termos atenção redobrada sempre que nos depararmos com um deles.

Vejamos então os diagramas propostos na figura 7.5.

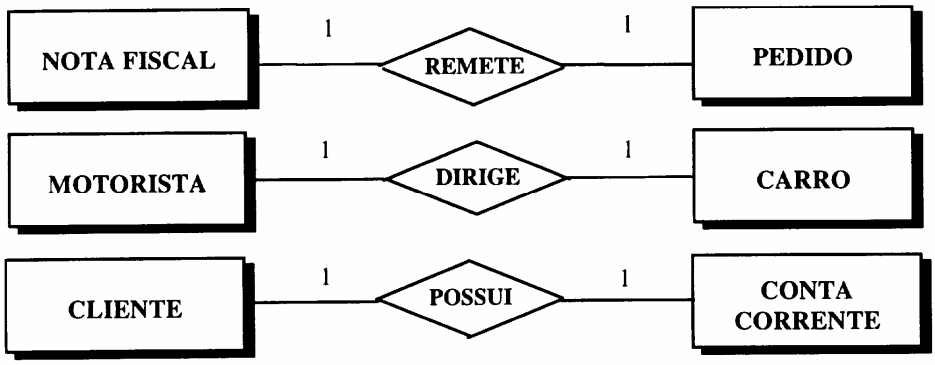


Figura 7.5

No primeiro exemplo, temos Nota Fiscal relacionada com Pedido de Um-para-Um, isto é, uma ocorrência de Nota Fiscal atende (relaciona-se) a uma ocorrência de Pedido. Na realidade, queremos dizer que através de uma Nota Fiscal remetemos somente um Pedido solicitado. Independente da existência de outros pedidos, uma nota fiscal só remete um pedido. Por outro lado, um pedido só é remetido por uma única nota fiscal.

Que implicação tem este relacionamento no mundo real?

Este relacionamento é restritivo quanto a operacionalidade do evento remeter, estabelece uma regra significativa e ampla: Um pedido nunca pode ser entregue parcialmente, já que isto implicaria em associar-se a ele mais de uma nota fiscal, e por outro lado, uma nota fiscal jamais poderá remeter dois ou mais pedidos, pela mesma implicação.

No segundo diagrama, apresentado temos uma entidade, que contém os carros de uma empresa, e outra entidade o elenco de motoristas da mesma organização.

O relacionamento está afirmando que um carro da empresa somente pode ser dirigido por um único motorista e que um motorista somente poderá dirigir um único carro.

O terceiro exemplo de diagrama apresenta uma situação típica de um crediário de uma loja, onde temos o cliente e sua conta corrente de crediário.

Sendo que um cliente somente tem uma conta corrente na loja, e por sua vez cada conta corrente existente pertence a somente um cliente.

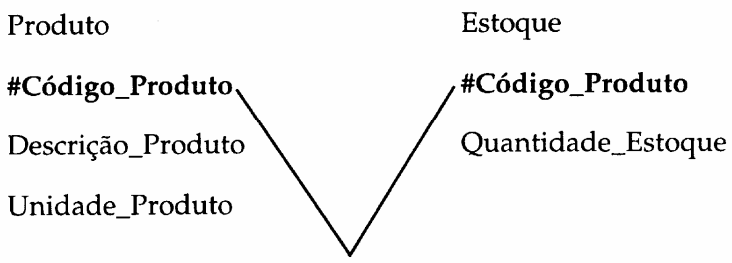
Mas a questão que gostaríamos de apresentar aos leitores é a seguinte: Onde colocamos a chave estrangeira para efetivar, logicamente, este relacionamento?

Sendo um relacionamento em que ambos os participantes possuem grau de cardinalidade Um, a referência lógica da chave estrangeira pode estar colocada em qualquer um dos lados do relacionamento. Como orientação de segurança e estabilidade, sugerimos sempre que esta chave estrangeira seja colocada no lado que tem a possibilidade semântica de evoluir, no futuro, para uma cardinalidade Muitos.

O questionamento da existência deste relacionamento ocorre quando, por um erro de interpretação da realidade, criamos duas entidades em que ambas descrevem ou caracterizam um mesmo objeto.

Neste caso, podemos visualizar que os atributos das duas entidades caracterizam um único objeto, e ainda mais, suas chaves identificadoras, as chaves primárias, são idênticas.

Vejamos o exemplo no diagrama da figura a seguir.

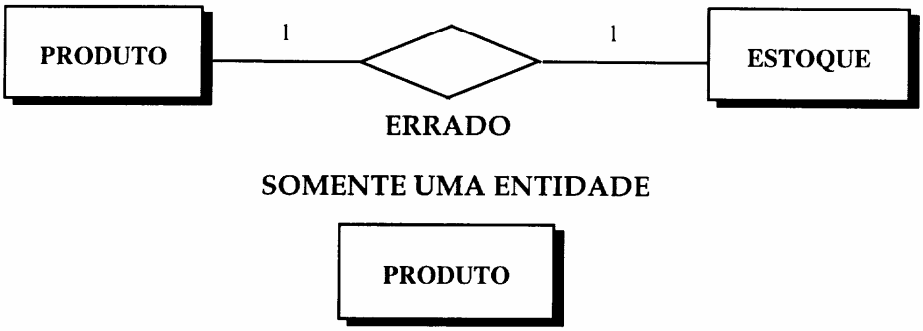


Chaves Primárias Idênticas

O exemplo é típico, pois quando estamos realizando nossos primeiros trabalhos de modelagem, cria-se uma entidade chamada Estoque para relacionar-se com Produto (Um-para-Um), por considerarmos o aspecto físico do Estoque como distinto de Produto.

Observando com mais atenção, podemos chegar à solução correta. Se dado um produto, este possui somente uma quantidade em estoque, considerando que esta quantidade representa o quanto temos do produto na empresa, o atributo quantidade em estoque é, na realidade, nada mais do que um dos atributos de produto.

É necessário usar um relacionamento 1:1? Não.



Quando estamos realizando o Projeto Físico de Banco de Dados, podem surgir situações em que iremos efetivamente colocar uma decomposição vertical da entidade, mas por um motivo nada conceitual. O objetivo de uma decomposição desta ordem poderia ser, por exemplo: a grande concentração de acessos à entidade Produto, sendo que nem todos estes acessos estejam interessados em recuperar o dado quantidade em estoque.

Os relacionamentos Um-para-Um podem ser utilizados quando estivermos com entidades complementares de outra, como nos casos de generalização de dados.

7.3 - Usar N :N ou Construir 2 vezes 1:N - A Escolha

Como modelagem de dados, antes de ser uma técnica também é uma arte, pois reflete a interpretação de um ser humano da realidade que o cerca, podemos ter uma mesma situação representada de formas diferentes.

Desta forma, a semântica de um modelo de dados pode, freqüentemente, esconder em conjuntos de relacionamentos Um-para-Muitos, um relacionamento (na realidade) Muitos-para-Muitos.

Vamos utilizar para esta análise a figura 7.6.

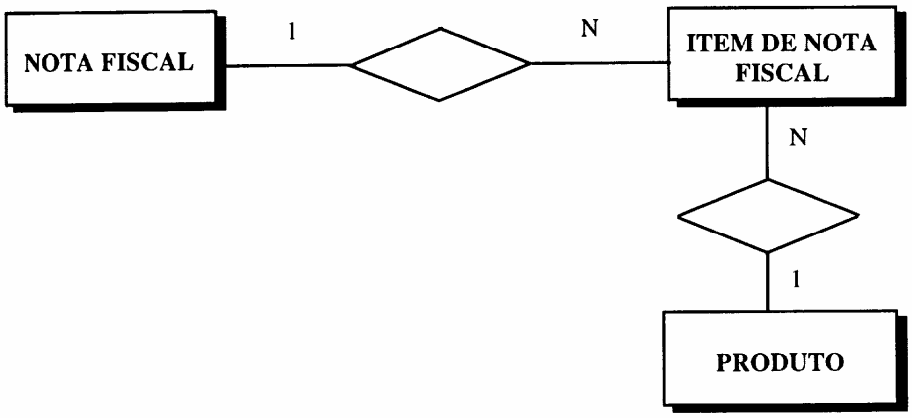


Figura 7.6

- Todas as notas fiscais têm, no mínimo, um item de nota fiscal relacionado;
- Todo Item de Nota Fiscal está relacionado a uma Nota Fiscal;
- Todo Item de Nota Fiscal está relacionado a um Produto.

A entidade Item de Nota Fiscal contém em suas ocorrências, os registros dos itens que constam em uma nota fiscal. Vejamos então as estruturas de dados que temos no exemplo:

ENTIDADE	ATRIBUTO	RELACIONAMENTOS
Nota Fiscal	Número_da_Nota_Fiscal *Código_do_Cliente_da_Not* a Data_da_Nota	Com Item de Nota Fiscal 1:N total *(todas as notas têm no mínimo* um item de nota fiscal relacionado)
Item de Nota Fiscal	Número_da_Nota_Fiscal Código_do_Produto Quantidade_Produto	Com Nota Fiscal 1:1 total (todo item de Nota Fiscal está relacionado a uma Nota Fiscal) Com Produto 1:1 total (todo item de Nota Fiscal está relacionado a um Produto)
Produto	Código_Produto Descrição_Produto Unidade_Produto Preço_Unitário_Produto Quantidade_Estoque_Produto	Com Item de Nota Fiscal 1:N Parcial

Colocamos o relacionamento entre Produto e Item de Nota Fiscal como parcial, no sentido de Produto para Item de Nota Fiscal, porque nem todos produtos são referenciados por uma ocorrência de Item de Nota Fiscal.

Já, por sua vez, o relacionamento entre Item de Nota Fiscal e Produto, no sentido de Item para Produto é total, porque todo item de uma nota fiscal é obrigatoriamente uma ocorrência de Produto.

Apesar de entidade Item de Nota Fiscal possuir uma nomenclatura bastante semântica, está na realidade exercendo uma função de relacionamento no modelo de dados.

Através desta entidade, estamos, na verdade, relacionando Nota Fiscal e Produto. Poderíamos substituí-la por um relacionamento de Muitos-para-Muitos, pois **Nota Fiscal Contém Produtos**.

Olhando atentamente o diagrama da figura 7.6, pode-se notar que a entidade Item-de-Nota-Fiscal apresenta uma convergência de cardinalidade Muitos nos dois relacionamentos ligados a ela. Este fato pode caracterizar um relacionamento de Muitos-para-Muitos. Vários autores denominam esta entidade com uma tipologia muito clara: Entidade Associativa, aquela que associa, que relaciona.

A figura 7.7 apresenta o Diagrama E-R, com a utilização de um Relacionamento no lugar da entidade associativa.

Figura 7.7

A estrutura de dados do modelo e sua lógica relacional no dicionário de dados ficará da seguinte forma:

ENTIDADE	ATRIBUTOS	RELACIONAMENTOS
Nota Fiscal	Número_da_Nota_Fiscal Código_do_Cliente_da_Nota Data_da_Nota	Com Produto através de Contém 1:N total (Todas as Notas têm no mínimo um produto fiscal relacionado)

ENTIDADE	ATRIBUTOS	RELACIONAMENTOS
Produto	Código_Produto Descrição_Produto Unidade_Produto Preço_Unitário_Produto Quantidade_Estoque_Produto	Com Nota Fiscal através de Contém 1: N Parcial (Nem todos os produtos estão contidos em uma Nota Fiscal)

RELACIONAMENTO	ATRIBUTOS	CONEXÃO
Contém	Número_da_Nota Código_do_Produto Quantidade_Produto	com Nota Fiscal (total) e com Produto (total)

A expressão do relacionamento seria então:

- Nota_Fiscal. Número_da_Nota = Contém.Número_da_Nota, e;
- Contém.Código_Produto = Produto.Código_Produto.

Este processo de comparação através de uma expressão lógica é denominado de navegação no modelo de dados.

Esta navegação, executada quando temos uma expressão composta para comparação dos campos, é denomina de navegação disjunta, pois executamos a comparação na seqüência linear do relacionamento. Comparamos primeiro o Número_da_Nota na entidade Nota_Fiscal com o número da nota no relacionamento Contém e na seqüência comparamos o Código_Produto no relacionamento Contém com um Código_Produto na entidade Produto.

capítulo

Agregação

8

8.1 - A Decomposição de um Relacionamento

Existem momentos em que temos uma visão dos dados que nos deixa em dúvida de como representar um fato que está relacionado a outro fato. Isto equivaleria a dizer que um relacionamento está relacionado a outro. Mas conceitualmente, não existem relacionamentos entre relacionamentos, é uma inverdade conceitual.

O que existe em verdade no mundo real, são relacionamentos dependentes de outros, que somente existem após a ocorrência do outro, considerado fundamental.

Vamos iniciar nosso estudo com um caso policial, como se estivéssemos lendo um romance policial moderno, localizado na Baixada Fluminense, que tanto assistimos nos telejornais. Uma típica (credo!) chacina, ou extermínio de menores (arhg!!, quando isto vai parar?).

Os assassinos em nosso caso foram encontrados, mas necessitamos de provas, tais como as armas do crime.

Podemos, nesta situação, afirmar que um criminoso assassinou várias vítimas e que uma vítima foi assassinada por vários criminosos, já que não podemos determinar com exatidão qual o tiro que realmente causou a morte do coitado, uma vez que vários meliantes participaram do fato. Estamos frente a frente com um relacionamento de Muitos-para-Muitos. Já imaginaram isto sendo explicado em um telejornal?

Figura 8.1

O diagrama da figura 8.1 apresenta um impasse: temos a entidade Meliante, e a entidade Vítima, relacionadas com cardinalidade Muitos-para-Muitos. Aparece então em cena uma terceira entidade, denominada de Arma, que contém as armas apreendidas. Como iremos ligar as ocorrências de Arma, ao fato, ao evento do Assassinato, este relacionamento com campos?

A entidade Arma está com suas ocorrências relacionadas a Meliante, ou está relacionada a Vítima?

Se ligarmos Arma a Meliante, não podemos afirmar que ela foi utilizada no assassinato de uma Vítima. Se ligarmos à Vítima, mais difícil torna-se estabelecermos uma relação com o Assassino.

Como resolver esta questão?

É relativamente simples esta solução, basta que nos detenhamos em retratar a realidade da mesma forma como ela é expressa em linguagem natural, ou seja:

- Um Meliante Assassina Vítimas. Quando ele Assassina Vítimas, Usa Armas. Esta realidade é apresentada graficamente (modelo E - R) na figura 8.2.

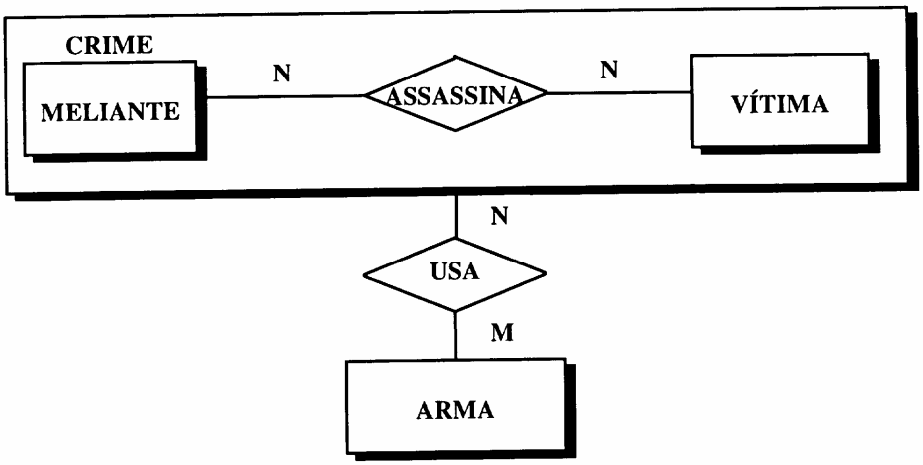


Figura 8.2

Existem, nesta expressão da realidade, dois verbos inter-relacionados, ou seja, existe um relacionamento entre objetos dependentes um da existência de outro.

São na realidade dois relacionamentos para retratar um fato por completo.

O que desejamos é relacionar uma ocorrência de Arma, com uma ocorrência do fato, do relacionamento Assassina (Se "A Gata Triste" raciocinasse assim, perdiam a graça, todos os seus livros!).

O relacionamento Assassina, como colocamos na situação, possui campos. Vamos então relacionar uma ocorrência deste relacionamento com uma ou várias ocorrências da entidade Arma, criando um relacionamento entre Assassina e Arma, que denominamos de Usa.

A figura 8.3, mostra a representação da agregação relacionada com Arma.

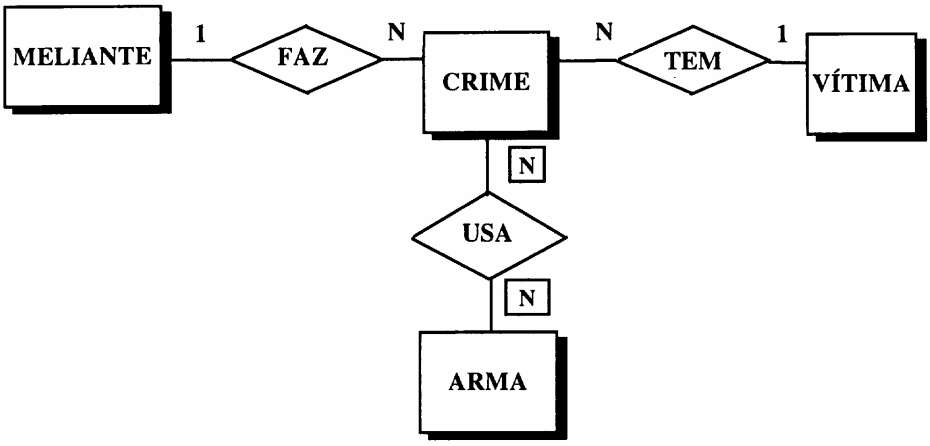


Figura 8.3

Interpretando o diagrama Entidade-Relacionamento, temos que enxergar o conjunto resultante de Meliante Assassina Vítima, relacionado com o conjunto das ocorrências da entidade Arma. Na realidade, estamos lendo este conjunto resultante como se ele mesmo fosse uma entidade.

Quando deixamos de lado um pouco o conceito entidade, podemos interpretar como o objeto Assassinato (Meliante Assassina Vítima) relacionado com Arma.

Conseguimos desta forma ligar a Arma à cena do Crime (Assassinato).

Para melhor entendimento vamos conhecer as estruturas de atributos das entidades envolvidas neste caso policialesco, e de seus relacionamentos.

DICIONÁRIO DE DADOS:

ENTIDADE	ATRIBUTOS	RELACIONAMENTO
MELIANTE	Registro do Meliante Nome Oficial Codinome Idade	Com Vítima através de Assassina 1:N (Parcial) (Nem todo Meliante assassinou uma Vítima)
VÍTIMA	Num. documento da vítima Nome do infeliz Sexo Idade	Com Meliante através de Assassina 1:N (Total) (Toda Vítima foi assassinada por um Meliante)
ARMA	Num. Apreensão da Arma Marca da arma Tipo da arma Número de série	Com Assassina através de Usa 1:N (Parcial) (Nem toda arma apreendida foi usada num assassinato)

RELACIONAMENTO	ATRIBUTOS	CONEXÕES
ASSASSINA	Data do crime Num. documento da Vítima Registro do Meliante	Com Meliante N:1 (Total) Com Vítima N:1 (Total)
USA	Num. documento da Vítima Registro do Meliante Num de apreensão da arma	Com Assassina N:1 (Total) Com Arma N:1 (Total)

Vamos agora observar as tabelas a seguir, que nos apresentam a simulação das tabelas representativas das entidades e dos relacionamentos envolvidos em nossa novela policial.

MELIANTE

REGISTRO	NOME	CODINOME	IDADE
121212	João Pedrum	Jão Caveira	33
221134	Luiz Cabrum	Lú Vampiro	42
334455	Carlos Sá	Carlinhos Mau	18
212378	Sérgio Bruks	Balaustre	26
335588	Tonho Silva	Tonho Facão	19

VÍTIMA

DOCUMENTO	NOME	SEXO	IDADE
111111111	Antônio Moacir	M	58
243387569	Júlio A. Macedo	M	35
806578913	Sazaina Moraeis	F	24
684714325	Ana Luiza Martins	F	32

ARMA

NÚMERO	MARCA	TIPO	SÉRIE
191	Taurus	Pistola	A656B767
192	Magnus	Pistola	Mg457T8V9

As tabelas apresentadas a seguir mostram apenas uma parte dos relacionamentos possíveis para as ocorrências das entidades Meliante e Vítima , considerando-se que, os casos que não estão relacionados, não foram até esta edição solucionados.

ASSASSINA

DATA	VÍTIMA	MELIANTE
11/02/92	111111111	121212
14/03/92	243387569	121212
03/04/92	806578913	334455

USA

VÍTIMA	MELIANTE	ARMA
111111111	121212	191
111111111	121212	192

O fato de um crime associar duas armas ao mesmo meliante, é um exercício de imaginação, mas que nos permite observar que a agregação admite que se tenha uma cardinalidade de Muitos-para-Muitos entre o objeto resultante do relacionamento entre Meliante e Vítima com a entidade Arma.

Agora que já vimos uma realidade literária, vamos trabalhar um caso mais normal em termos de sistemas de aplicação.

8.2 - Agregação e Cardinalidade

Seja a situação de relacionamento Muitos-para-Muitos que havíamos estudado quando da análise deste tipo de relacionamento, entre a entidade Funcionário e a entidade Projeto.

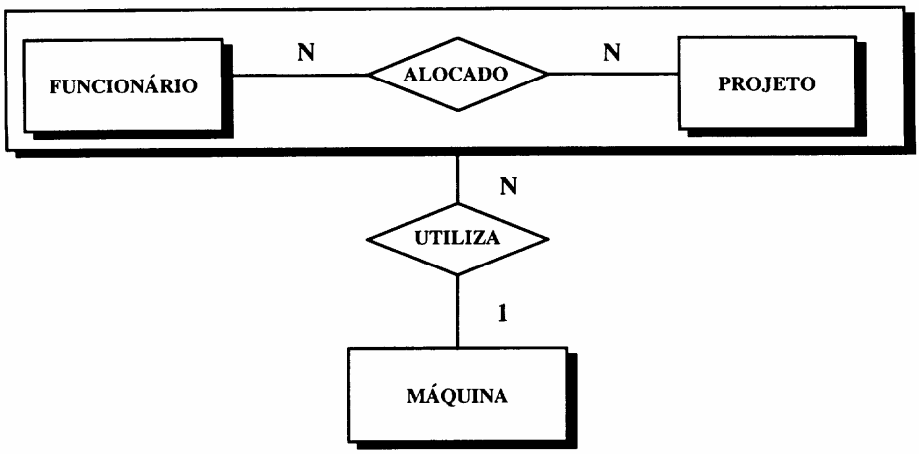


Figura 8.4

Um Funcionário Atua em muitos Projetos e, um Projeto tem trabalhando nele muitos Funcionários, conforme o diagrama E-R da figura 8.4.

Quando um Funcionário está trabalhando em um Projeto, ele pode Utilizar uma ou nenhuma Máquina para a realização de suas atividades.

Novamente temos uma situação de um evento decorrente de outro, e ainda com uma característica adicional, que é a opcionalidade do segundo evento acontecer.

A visão agregada de Funcionário Alocado Projeto, permite-nos tratar este bloco de modelo, como sendo uma entidade consolidada por um fato, mas vejam bem que estamos considerando um bloco, e que este bloco então relaciona-se com Máquina, opcionalmente.

Diferentemente do caso anterior onde uma Arma era usada por um criminoso somente em Assassinato, agora temos a colocação de que uma máquina pode estar sendo utilizada por diversos funcionários atuando em projetos.

Como devemos tratar esta situação?

Lendo, a situação fica mais simples, senão vejamos: um Funcionário Atuando em Um Projeto caracteriza uma ocorrência de Alocado, e esta ocorrência utiliza uma ou nenhuma Máquina, por outro lado uma Máquina pode ser utilizada por "n" Funcionários atuando em Projeto.

Temos então um relacionamento de um-para-muitos entre Máquina e, a visão de Funcionário Alocado Projeto.

Como existe uma cardinalidade de Um-para-Muitos no relacionamento Utiliza, este não será um relacionamento com campos, sendo realizado pela execução de uma expressão de comparação lógica.

Vamos então observar as estruturas do dicionário de dados para melhor compreendermos as ligações lógicas e como se realizam.

DICIONÁRIO DE DADOS:

ENTIDADE	ATRIBUTOS	RELACIONAMENTO
FUNCIONÁRIO	Matrícula Funcionário Nome Funcionário Data Admissão	Com Projeto através de Atual 1:N (Parcial)
PROJETO	Código Projeto Nome Projeto Verba Projeto	Com Funcionário através de Atua 1:N (Parcial)
MÁQUINA	Código Máquina Nome Máquina	Com Atua através de Utiliza 1:N (Parcial)

RELACIONAMENTO	ATRIBUTOS	CONEXÕES	RELACIONAMENTO DO BLOCO AGREGADO
Alocado	Código_Projeto Matrícula_Func Cód_Máquina	Com Projeto N:1 (Total) Com Funcionário N:1 (Total)	Com máquina através de utiliza N:1 (Parcial)

RELACIONAMENTO	ATRIBUTOS	CONEXÕES	RELACIONAMENTO DO BLOCO AGREGADO
UTILIZA		Com Alocado N: 1 Com Máquina 1:1	

Este caso de agregação apresenta da mesma forma uma decomposição de um relacionamento em dois, pois ao analisarmos a situação, encontraremos sempre dois fatos, dois eventos acontecendo, sendo um dependente de outro.

Funcionário Utiliza Máquina, "quando" Alocado a Projeto. Este tipo de temporalidade representado pela palavra "quando" é que nos dá a pista, o caminho correto para a solução do caso.

Sempre que nos depararmos com um relacionamento envolvendo mais de duas entidades, devemos questionar as entidades que se ligam em um relacionamento básico, através da colocação da questão:

- **Quando acontece o fato?**

Vamos então conhecer um outro caso de agregação, agora já visualizando uma temporalidade, e executando o questionamento temporal que discutimos.

Seja então a seguinte situação:

Um médico atende a muitos pacientes, que o consultam, e um paciente pode realizar consultas com muitos médicos. Sempre que um paciente consulta um médico, este fornece uma receita, que pode ter um, ou vários remédios.

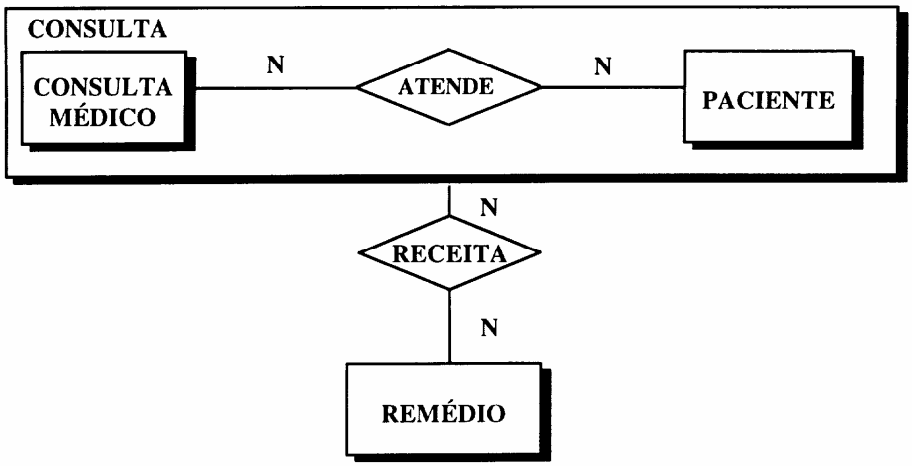


Figura 8.5

A figura 8.5 nos mostra o diagrama E-R para esta visão dos dados.

Este caso envolve o mesmo tipo de ligação dos outros dois, mas detalhando-se agora que esta visão é de um relacionamento com o agregado de Muitos-para-Muitos.

Interpretando o diagrama temos:

- Um Remédio é receitado em muitas Consultas;
- Uma Consulta receita muitos Remédios.

Mas o que é esta Consulta da qual estamos falando?

É nada mais nada menos que o relacionamento Atende existente entre a entidade Médico e a entidade Paciente, que possui cardinalidade de Muitos-para-Muitos.

Vamos observar então como fica a construção do dicionário de dados para este exemplo em pauta.

Observem, leitores, que agora neste caso temos três entidades se dois relacionamentos com campos, para solucionar o problema.

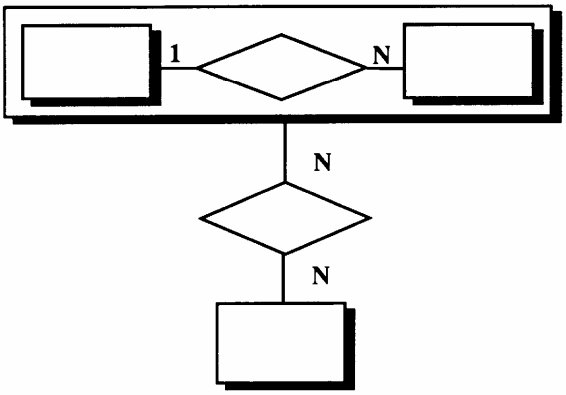
DICIONÁRIO DE DADOS:

ENTIDADE	ATRIBUTOS	RELACIONAMENTOS
MÉDICO	Código Médico Nome Médico	Com Paciente através de Consulta 1:N (Parcial)
PACIENTE	Número Paciente Nome Paciente	Com Médico através de Consulta 1:N (Parcial)
REMÉDIO	Código Remédio Nome Remédio	Com Consulta através de Receita 1:N (Parcial)

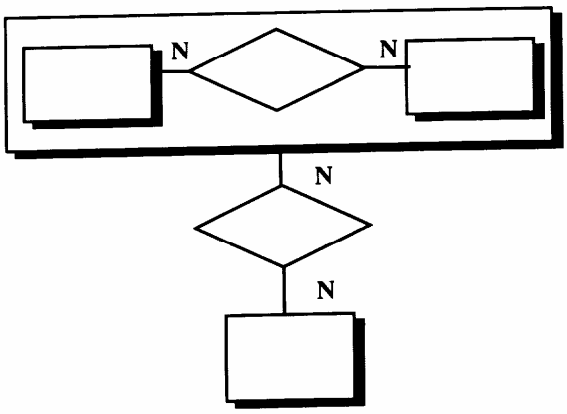
RELACIONAMENTO	ATRIBUTOS	CONEXÕES
ATENDE	Código Médico Número Paciente Data Consulta	Com Médico N:1 Com Paciente N:1 Com Remédio através de Receita 1:N
RECEITA	Código Remédio Código Médico Número Paciente Posologia Remédio	Com Remédio N:1 Com Consulta N:1

8.3 - Restrições para Uso de Agregação

Queremos que o leitor observe e grave em sua mente a regra básica para que se possa utilizar uma Agregação em um modelo de dados:



ERRADO



CERTO

Figura 8.6

Só podemos utilizar agregação quando temos um relacionamento de Muitos-para-Muitos, que representa um fato, pois caso contrário a terceira entidade envolvida estará relacionada sempre com uma das entidades em questão (figura 8.6).

Para melhor exemplificar, e permitir a compreensão desta restrição, vamos considerar que o relacionamento entre Funcionário e Projeto, estudado anteriormente, tivesse uma cardinalidade de Um-para-Muitos, ou seja, Um Projeto tem muitos Funcionários, mas um Funcionário trabalha somente em um Projeto.

Ora, se o funcionário só trabalha em um Projeto, a máquina ou as máquinas que ele utiliza estão relacionadas diretamente a ele, uma vez que ele só possui uma existência de relacionamento com projeto.

Sempre que tivermos relacionamentos de cardinalidade Um-para-Muitos, a terceira entidade está relacionada com uma das duas (Figura 8.7).

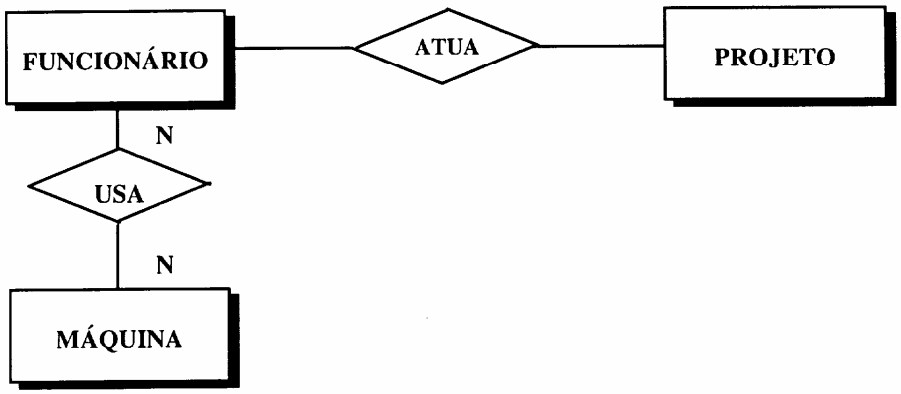


Figura 8.7

Mesmo que se interprete como um relacionamento decorrente do evento, este evento só ocorre uma vez, logo podemos relacionar o terceiro objeto participante com o objeto que tem cardinalidade muitos.

Por favor, leitores, gravem bem: somente utilizamos agregação quando temos relacionamentos de cardinalidade Muitos-para-Muitos.

Vamos agora então analisar uma situação que faz muitos autores negarem o conceito de agregação e evitarem sua utilização.

8.4 - Como Utilizar Agregação por Implementação de Entidades Associativas, (ou meu Banco de Dados não Permite todas as Extensões E-R)

Existe uma preocupação, que muitas vezes leva os analistas e projetistas de bancos de dados a abandonarem suas visões de dados, e atuarem diretamente no modelo físico, deixando de lado o modelo lógico.

Uma destas preocupações é referente ao Sistema Gerenciador de Banco de Dados, que não permite ou não possui características que possibilitem a implementação de soluções de relacionamentos com campos, e ainda mais com agregações de dados.

Isto em nada afeta os trabalhos de modelagem, ou sequer impossibilita esta forma de ver os dados. Devemos, isto sim, modelar de acordo com a

realidade, e após, adaptarmos esta visão que obtemos da realidade ao esquema permitido pelo SGBD.

Vamos ver a situação do exemplo Funcionário Alocado em Projeto Usa Máquina, em um banco de dados qualquer, supondo que ele não implemente este conceito de Muitos-para-Muitos e agregação. Mostraremos que a visão lógica nunca deve ser superada pela visão física, quando da análise e modelagem conceitual de dados.

Figura 8.8

A figura 8.8 apresenta o primeiro estágio do desdobramento lógico do relacionamento Muitos-para-Muitos, utilizando-se uma **entidade associativa** para representá-lo.

A leitura do diagrama agora se faz como Funcionário Tem Alocação Associada a Projeto.

Temos então o desdobramento do relacionamento Alocado, em dois relacionamentos simples: Tem e Associado.

Sendo que os campos que pertenciam ao relacionamento Alocado, agora pertencem a uma entidade associativa denominada Alocação.

E como fica a ligação entre este evento antes representado por uma relacionamento único, com a entidade Máquina?

É relativamente simples. Máquina relaciona-se da mesma forma com o evento, independente deste ser representado por um relacionamento com campos ou uma entidade, já que o importante no contexto é o entendimento e a expressão de uma realidade.

Mas e se o relacionamento entre Máquina e Alocação for mais um relacionamento com campos, como em nosso exemplo anterior?

Novamente vamos desdobrar este relacionamento com campos em dois relacionamentos com uma outra entidade associativa, para a efetivação do fato dependente.

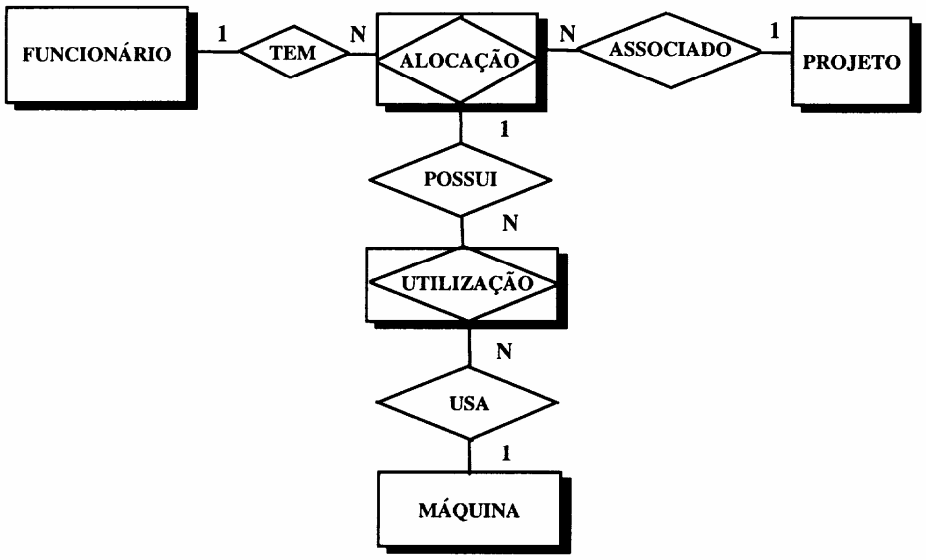


Figura 8.9

Temos então, como na figura 8.9, Máquina Usa Utilização Possui Alocação.

Propositalmente escrevemos a leitura do modelo em sua ordem inversa, para que se possa destacar que este tipo de conversão de modelo provoca sempre **cacofonismos** semânticos, mas o modelo continua expressando uma realidade. Vamos efetivar a leitura linear dos fatos, para ver o resultado que encontramos:

- Funcionário Tem Alocação Associada Projeto;
- Alocação Possui Utilização Usa Máquina.

É, os leitores haverão de concordar, resolve, mas que não fica muito semântico, isto é verdade, não fica mesmo.

O importante é que se você, leitor, utiliza um banco de dados que permite este tipo de implementação (agregação e Muitos-para-Muitos), não deixe de utilizar, já que em projetos de grande complexidade e volume de entidades e relacionamentos, o modelo ficará extremamente sujo em termos de semântica para expressar uma realidade. O ideal seria que todos os bancos de dados fossem construídos para permitir esta implementação conceitual,

permitindo que se desenhe um esquema conceitual mais coerente com a realidade do negócio para o qual desenhamos um sistema.

8.5 - Relacionamentos entre Blocos do Modelo

Já vimos até aqui que uma agregação de dados pode relacionar-se com outras entidades, mas existe ainda a possibilidade de realizarmos relacionamentos de qualquer grau entre dois blocos de agregação existentes no modelo de dados.

Acreditem, os autores deste livro não enlouqueceram não! Simplesmente estamos procurando conduzir a modelagem de dados em sua forma mais pura, buscando sempre uma eficiência e fidelidade do modelo de dados ao mundo real, e a forma como ele é expresso em linguagem natural, aquela que utilizamos quando não estamos preocupados com Programas, Máquinas e Softwares.

Vamos estudar então esta extensão da utilização de agregações em um modelo de dados.

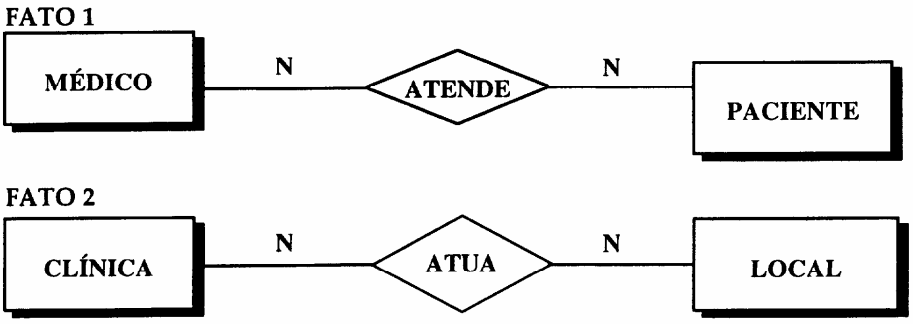


Figura 8.10

A figura 8.10 apresenta dois blocos distintos com relacionamentos Muitos-para-Muitos em cada um, expressando dois fatos do mundo real.

O fato número 1 retrata uma situação que já estudamos, em que um médico atende a muitos pacientes, e um paciente faz consultas com muitos médicos.

O fato número 2, aparentemente dissociado, representa uma Clínica Atua em muitos Locais, sendo que em um Local Atuam muitas Clínicas.

Como as consultas médicas não são realizadas no espaço, deve ser possível associarmos o fato da Consulta a uma Localização deste acontecimento.

A questão é, quais as consultas realizadas pela ocorrência X de Clínica no Local Y?

Podemos querer saber, por outro lado, em que Local e Clínica foi realizada uma determinada consulta.

Isto nos conduz a afirmar que os dois blocos do modelo de dados não estão realmente dissociados, que estão profundamente relacionados. Então o que nos falta é representar a associação dos dois fatos.

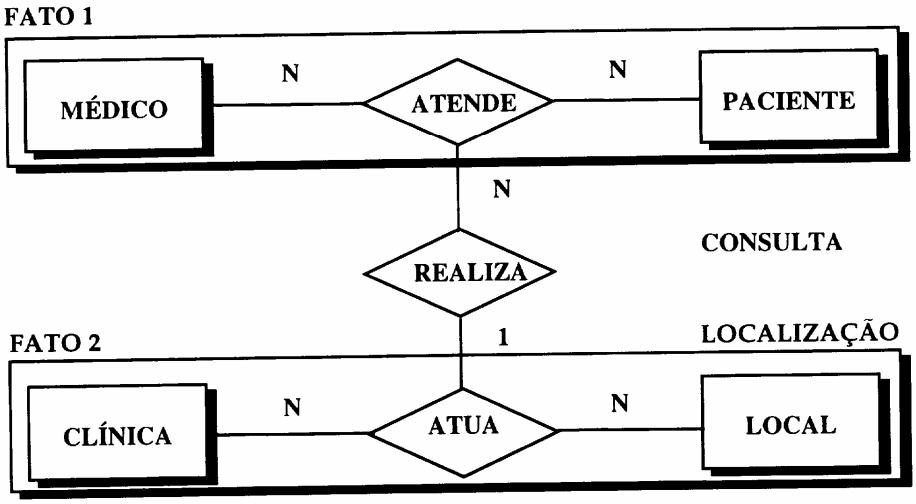


Figura 8.11

A figura 8.11 nos apresenta a solução, com o relacionamento existindo entre as duas agregações, através de Realiza.

Temos então um relacionamento com cardinalidade de Um-para-Muitos entre a agregação Localização e a agregação Consulta.

A leitura do diagrama E-R reflete claramente agora o que acontece no mundo real: uma Clínica Atua em muitos Locais, quando uma Clínica Atua em um Local Realiza muitas Consultas, isto é, Médico Atende Paciente.

Esta mesma situação pode, como já afirmamos, ser implementada com desdobramento dos relacionamentos Muitos-para-Muitos em entidades associativas, apesar de perder grande parte da simplicidade semântica desta solução.

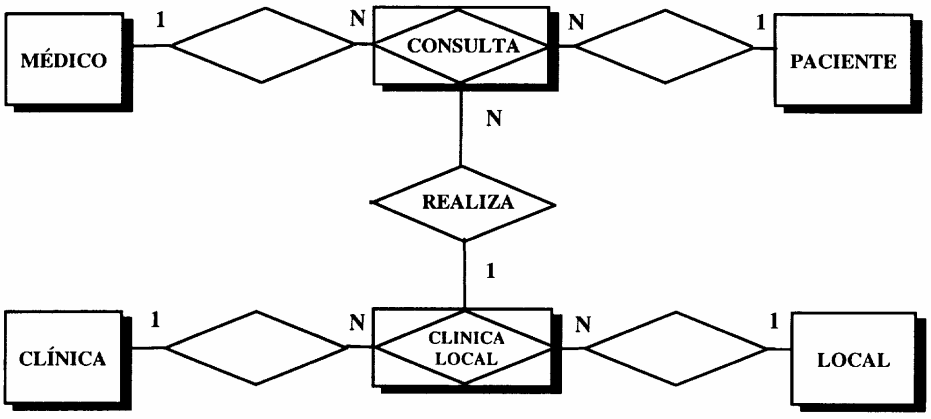
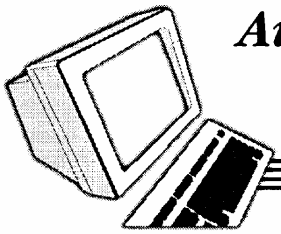


Figura 8.12

A figura 8.12 apresenta o mesmo caso que estudamos, representado através da utilização de entidades associativas.

Um detalhe importante e que muitos projetistas esquecem, ou fingem esquecer, que a fidelidade semântica dos dados e relacionamentos deve sempre ser mantida, procurando-se dar nomenclaturas às entidades e relacionamentos que expressem um sentido, evitando-se de todas as formas, a utilização de siglas, tanto para entidades quanto para relacionamentos, que impeçam a leitura simples e natural de um diagrama de Entidades e Relacionamentos.

Capítulo



Auto-Relacionamento 9

9.1 - Introdução

Para que possamos entender bem este tipo especial de relacionamento, devemos conceituar melhor o que é auto-relacionar-se.

Nos dias atuais, muito se comenta sobre qualidade de vida. E um dos comentários mais ouvidos, visto que não somos psicólogos e nem pretendemos, é de que para se estar de bem com a Vida, devemos em primeiro lugar, estar de bem com nós mesmos. Devemos nos auto-relacionar muito bem, gostar de si mesmo, antes de tudo, para poder encarar a Vida de frente.

Estas afirmativas apresentam um auto-relacionamento de nós seres humanos com nós mesmos, isto é auto-relacionar-se.

Em nosso estudo, é até certo ponto bem mais simples de entender o auto-relacionamento, pois em uma classe de objetos, eles se relacionam entre si.

Em uma entidade, suas ocorrências possuem relacionamentos próprios entre elas.

9.2 - Auto-Relacionamento Um-para-Muitos

Auto-relacionamentos são, em verdade, representações de estruturas de hierarquias na maioria das vezes. Por exemplo, vamos considerar uma entidade Pessoa, cujas ocorrências são representativas de inúmeras pessoas de um determinado local. Pois bem, entre estas inúmeras ocorrências de pessoas

existem relacionamentos bem-definidos, como É_filho_de. Isto é, algumas pessoas são filhas de outras pessoas.

Um outro exemplo seriam os funcionários de um empresa. Entre estes funcionários existe uma relação de hierarquia. Podemos afirmar que alguns funcionários são gerentes de outros, que por sua vez são subordinados a um gerente.

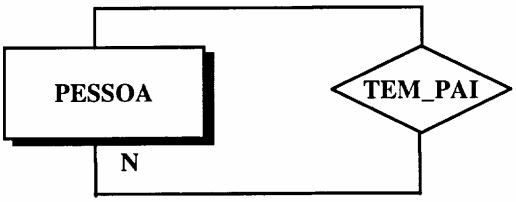


Figura 9.1

A figura 9.1 apresenta um diagrama para a situação em que Uma Pessoa possui Muitos Filhos, um relacionamento de cardinalidade Um-para-Muitos, mas um auto-relacionamento, já que temos uma única entidade envolvida.

E como este fato se efetiva logicamente?

Como já comentamos neste livro, da mesma forma que as certidões de nascimento referenciam o nome dos pais, estas ocorrências estariam realizando referências lógicas a outras ocorrências da mesma entidade.

Vamos definir o dicionário de dados correspondente a esta situação para a entidade Pessoa, mostrados nas tabelas a seguir.

ENTIDADE	ATRIBUTOS
PESSOA	Identificação_Pessoa Nome Identificação_Pessoa_PAI Identificação_Pessoa_MÃE

RELACIONAMENTO	EXPRESSÃO
Tem_PAI	Pessoa.Identificação_Pessoa_PAI = Pessoa.Identificação_Pessoa
Tem_MÃE	Pessoa.Identificação_Pessoa_MÃE = Pessoa.Identificação_Pessoa

Seja então a entidade Pessoa e seus auto-relacionamentos Tem_PAI e Tem_MÃE, como enxergar estes dados em uma única tabela? A solução está mostrada na tabela simulada apresentada na figura 9.2.

Tabela da Entidade Pessoa

IDENTIFICAÇÃO PESSOA	NOME	IDENTIFICAÇÃO PESSOA_PAI	IDENTIFICAÇÃO PESSOA_MÃE
I-68	Carlos Feliciano	null	null
I-99	Jussara Pinto	I-68	I-29
I-29	Cláudia Bicoy	null	null
I-45	Pedro Luiz Bil	I-68	I-29
I-34	Cláudio Carvil	I-55	I-78
I-55	Antônio Luiz	null	null
I-78	Orvandina	null	null

Figura 9.2

Pelo conceito de generalização temos, na realidade, múltiplos subconjuntos de dados, dentro da entidade Pessoa, tais como Filhos de Pai, e Filhos da Mãe (desculpem a expressão!!!).

Brincadeiras à parte, a leitura dos dados da tabela nos apresenta a interpretação do mundo real que está retratada no diagrama da figura 9.3.

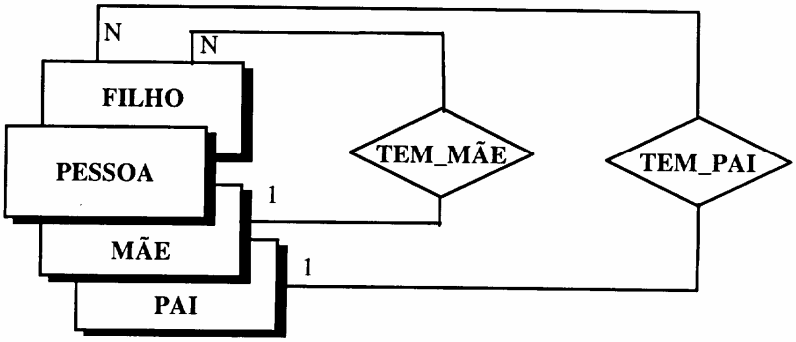


Figura 9.3

- Jussara é filha de Carlos Feliciano, pelo relacionamento Tem_Pai, e filha de Cláudia Bicoy pelo relacionamento Tem_Mãe.

A própria interpretação do relacionamento mostra-nos que as ocorrências da entidade assumem papéis diferentes conforme seu posicionamento no relacionamento.

O diagrama que vimos até este instante não expressa estes papéis. Este conceito de papel exercido, função, role em inglês, deve ser passado também para o diagrama de entidades e relacionamentos, porque permite que se entenda o sentido do relacionamento e qual tipo de participação no relacionamento tem uma ocorrência.

A figura anterior apresenta o diagrama E-R, agora com a forma de expressão do role, papéis da entidade em cada lado do relacionamento.

9.3 - Auto-Relacionamento Muitos-para-Muitos

Os auto-relacionamentos podem possuir qualquer um dos tipos de cardinalidade. Vamos então aprender a visualizar o tipo de cardinalidade Muitos-para-Muitos.

É muito comum este tipo de cardinalidade quando desejamos representar composições de algum objeto, por exemplo:

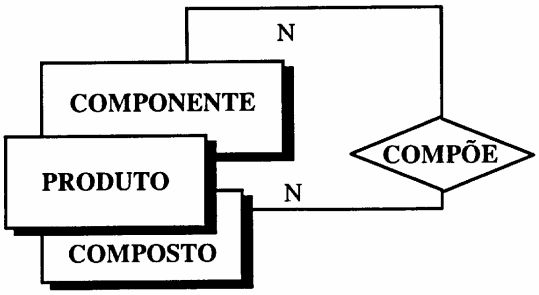


Figura 9.4

Em uma indústria, um produto é composto de vários outros produtos, que são componentes. Por outro lado, um produto componente pode participar da composição de muitos produtos.

Observem que estamos lidando com um único tipo de objeto, Produto.

Temos então um auto-relacionamento Muitos-para-Muitos, com campos, que expressam composição de objetos.

Os campos deste relacionamento representam a estrutura de engenharia de um produto.

A figura 9.4 apresenta este relacionamento, com os papéis existentes (componente e composto) em função do relacionamento.

Outro aspecto importante é entendermos como fica a estrutura dos atributos pertinentes à entidade Produto e ao relacionamento Compõe (figura 9.5).

ENTIDADE	ATRIBUTO
PRODUTO	Código_Produto Descrição_Produto Unidade_Produto

RELACIONAMENTO	ATRIBUTO
COMPÕE	Código_Produto_Composto Código_Produto_Componente Quantidade_Participação_Componente

Figura 9.5

Vamos imaginar uma tabela com Produtos e uma tabela do Relacionamento.

TABELA DE PRODUTOS

PRODUTO	DESCRIÇÃO	UNIDADE
001	Terminal de vídeo	peça
002	Tubo Imagem 14"	peça
003	Caixa para terminal P22	peça
004	Transformador XXX	peça
005	Botão de controle	peça
010	Gabinete para CPU 386	peça
011	Botão dois estágios	peça
023	Capa plástica para vídeo	peça
045	Parafuso Phillips 3/4"	peça
068	Cola Superbonder	tubo
087	Fita isolante borracha	rolo

PRODUTO	DESCRIÇÃO	UNIDADE
055	Tinta anticorrosiva verde	litro
076	leed vermelho	peça
077	leed verde	peça
081	Painel frontal	peça
099	Caixa para vídeo	peça

Vamos ver então como ficaria o relacionamento Compõe para a estrutura de engenharia de um terminal de vídeo fictício.

Compõe para o Produto 001 Terminal de Vídeo

Código_Produto Composto	Código_Produto Componente	Quantidade Componente
001	002	1
001	003	1
001	004	1
001	005	1
001	011	1
001	023	1
001	068	4
001	087	3
001	099	1

O que queremos mostrar aos leitores, é a fácil visualização dos dados no interior do relacionamento.

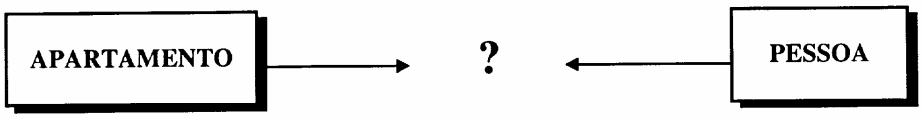
Para uma estrutura de produto mais complexa, vários níveis, o modelo não sofre mudanças, pois nada mais é do que a interiorização do relacionamento, isto é, a composição de um produto componente, e assim por diante.

9.4 - Auto-Relacionamento e Papel em um Relacionamento

Freqüentemente, vemos nossos alunos confundirem-se quando enfrentam determinados contextos de sistema, entre os papéis assumidos por ocorrências de uma entidade relacionada a outra, com um auto-relacionamento.

Vamos examinar uma situação em que este caso aparece.

Supondo que temos uma entidade Apartamento, e uma entidade Pessoa.



Existe então para cada apartamento um Locador (que é o proprietário) e um Locatário (a pessoa que aluga o apartamento), que por sua vez são ocorrências de Pessoa.

Muitas vezes temos visto os alunos realizarem a ligação de Pessoa com apartamento por um relacionamento Possui, para identificar os proprietários e um auto-relacionamento em Pessoa, para representar o fato de alguém ser Locatário.

Mas a realidade de que quem aluga o apartamento está relacionada com o objeto apartamento diretamente, e com o proprietário através deste apartamento indiretamente.

O papel de Locador é o do objeto que se relaciona com Apartamento através de um relacionamento que denominamos Possui, e o papel de Locatário é o obtido através do relacionamento entre Apartamento e Pessoa, denominado Aluga. São na realidade dois relacionamentos distintos entre Pessoa e Apartamento.

Esta forma de enfoque permite que uma ocorrência de Pessoa possa estar relacionada com uma ocorrência de Apartamento por Possui (proprietária do apartamento), e esta mesma ocorrência de Pessoa pode ter um relacionamento com outra ocorrência de Apartamento através do relacionamento Aluga, referindo-se ao fato de que esta pessoa pode estar alugando um apartamento.

O que é mais importante ainda de entendermos é que entre duas entidades pode haver mais de um relacionamento, como neste caso.

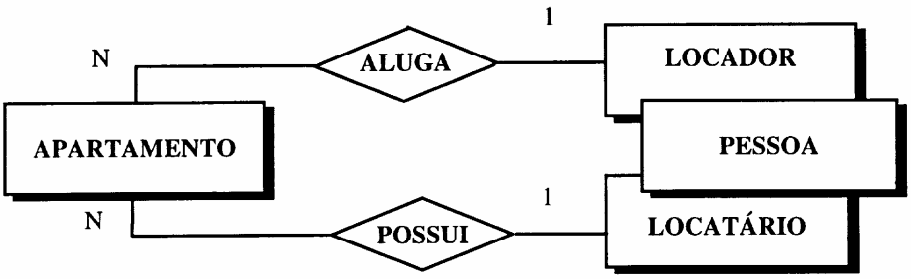


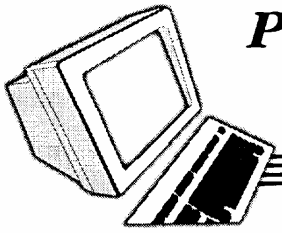
Figura 9.6

O diagrama da figura 9.6 mostra estes dois relacionamentos, e como ambos possuem cardinalidade de Um-para-Muitos, concluímos que existem duas chaves estrangeiras em apartamento, uma para ligação com o papel Locatário e outra para a ligação com o papel Locador.

A seguir, apresentamos a estrutura de dados do dicionário, para estes dois relacionamentos.

ENTIDADE	ATRIBUTOS
PESSOA	Código_Pessoa Nome_Pessoa
APARTAMENTO	Número_Apartamento Número_do_Prédio Endereço_Predio Código_do_Proprietário # Chave estrangeira para Possui Código_do_Locatário # Chave estrangeira para Aluga

Capítulo



Por Onde Começar? 10

Agora que já conhecemos o modelo Entidade-Relacionamento proposto por Peter Chen, agora que já sabemos o que são visões de dados, vamos por mãos à obra.

Mas é natural que o ponto mais crucial seja ligar os motores. Vamos então aprender como podemos iniciar um trabalho de Modelagem de dados para um sistema, já que nosso objetivo neste livro é não sermos meramente didáticos, e sim passar experiências e acelerar as possibilidades de você, leitor, efetuar realmente trabalhos de desenvolvimento de sistemas a partir de modelos de dados.

A maior dificuldade nos trabalhos de modelagem de dados encontra-se nos trabalhos iniciais de identificação das entidades.

Afinal o que vem a ser as entidades em um sistema, como podemos localizá-las e identificá-las sem que incorramos numa margem acentuada de erros?

Se o analista vem de uma formação tradicional, totalmente orientada a procedimentos, por onde começar?

Como conduzir o usuário a nos mostrar estas entidades que tanto necessitamos conhecer?

Embora seja um tema um tanto quanto ambíguo, de difícil formalização, existem na verdade formas e regras que podemos seguir para termos produtividade para realizar estas descobertas, que nos levarão a dominar um problema de negócios a ser informatizado.

10.1 - Iniciando um Modelo

Cumpre aqui lembrarmos que a visão de dados não nos é tão distante quanto à primeira vista nos parece.

Analisemos então um sistema a ser desenvolvido dentro de enfoque com orientação a dados e que irá atender à área de vendas de uma empresa, ou melhor, ao negócio Vendas de uma empresa.

Ora como não conhecemos nenhum detalhe, ou ainda mais, não temos nenhuma experiência anterior com sistemas de vendas, existe neste instante uma abstração global designada como **Venda**, sobre a qual iremos partir para nosso trabalho de modelagem. Teríamos num primeiro momento a entidade Venda. Mas o que caracteriza uma Venda?

Ao questionarmos este objeto denominado Venda, passaremos a observar que este elemento abstrato, na realidade, tem um elenco de atributos e possui ainda outros objetos encapsulados relativamente simples, de fácil percepção, pois inclusive convivemos com eles em nosso dia-a-dia. Exemplificando então: uma venda é descrita por um vendedor tirando pedidos de produtos feitos por alguém que deseja adquirí-los.

Esta afirmativa desmembra, na realidade, a abstração inicial denominada Venda em três objetos agora melhor delineados. Senão vejamos: existem os objetos Vendedor, Pedido e Cliente.

A questão que define conceitualmente uma entidade é se desejamos guardar dados para o objeto em si.

Ora, Pedidos qualifica a existência de uma certa quantidade de ocorrências de objetos Pedido, os objetos Cliente e Vendedor da mesma forma. Logo temos já neste instante descobertas três das entidades que participam do contexto de Vendas.

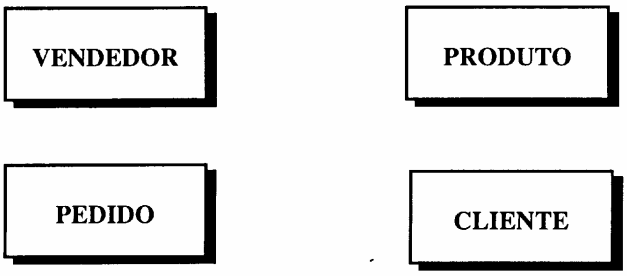
Estes objetos são qualificados como entidades porque podemos identificar suas ocorrências individualmente, podendo ser construídas tabelas com informações sobre clientes, sobre pedidos e sobre os vendedores da empresa.

Mas uma análise mais apurada logo irá, através de uma simples interpretação, nos lembrar de que uma venda se efetiva para alguma coisa.

Se realizamos uma venda, estamos obviamente vendendo alguma coisa, a qual denominamos de Produto.

Se quando vendemos, o fazemos para diversos produtos possíveis, logo temos também uma entidade Produto, pois existem diversas ocorrências de objetos produto no contexto de uma venda.

Obtemos então neste momento um diagrama E-R, com entidades assim delineadas:



Podemos agora então começar a analisar e verificar a existência de relacionamentos entre estas entidades, ou seja, quais são objetos do tipo fato, que existem ligando estas entidades.

Neste momento, visualizamos um relacionamento de características importantes dentro do contexto.

Quando afirmamos que um pedido contém produtos, estamos relacionando as entidades pedido e produto. Bem, neste instante já sabemos que existe o relacionamento pois ele é claro no mundo real. Necessitamos, isto sim, compreendê-lo melhor. Qual é a sua cardinalidade, como os dados se relacionam?

Sabemos, bastando para isto lembrarmos de um documento Pedido, que um pedido pode conter 1 ou vários produtos, não existindo um número exato de quantos produtos estarão em um pedido, pois depende da venda, do momento, de uma negociação.

Então observando no sentido de visão de pedido para produto, encontramos uma cardinalidade que diz que um pedido contém muitos produtos, logo cardinalidade Um-para-Muitos.

Mas os produtos da empresa são vendidos uma única vez na vida? É óbvio que não. Um produto da empresa é vendido diversas vezes para clientes diferentes ou para o mesmo através de vários pedidos.

Novamente analisando agora o sentido inverso de interpretação, de produto para pedido, encontramos uma cardinalidade de um produto em muitos pedidos, isto é, Um-para-Muitos.

Ora, conforme já vimos no modelo Entidade-Relacionamento, quando temos um relacionamento entre duas entidades, no qual ambos os sentidos de leitura do diagrama apresentam cardinalidade Um-para-Muitos, isto está caracterizando que o relacionamento é do tipo Muitos-para-Muitos.

Logo já temos uma das representações do diagrama de Entidades e Relacionamentos.

Mas neste ponto perguntamos:

- Onde estão as informações de quantidade de produto vendida?
- O preço total do produto no pedido?
- A seqüência do produto dentro do pedido?

Ora, já sabemos que os relacionamentos com cardinalidade Muitos-para-Muitos possuem campos, têm a forma de uma tabela, logo estas informações fazem parte do jogo de atributos deste relacionamento entre Pedido e Produto, e que denominaremos de Contém, conforme apresentado na tabela da figura 10.1, para expressar melhor a realidade.

Devemos continuar o nosso trabalho no sentido de verificar a possibilidade de desmembrar as entidades já conhecidas até então. Para isto torna-se necessário que se desenhem os atributos que descrevem cada entidade.

O que caracteriza um pedido?

Um pedido pode ser caracterizado por um número de identificação, pela sua data, a identificação do cliente que o fez, a identificação do vendedor que o emitiu, e basicamente os produtos que estão vendidos em um pedido.

ENTIDADE	ATRIBUTOS	RELACIONAMENTO
VENDEDOR	Identificação_Vendedor Nome_Vendedor	
CLIENTE	Identificação_Cliente Nome_Cliente	

ENTIDADE	ATRIBUTOS	RELACIONAMENTO
PRODUTO	Identificação_Produto Descrição_Produto	Com Pedido 1:N (Contém)
PEDIDO	Número_Pedido Valor_Pedido Data_Pedido	Com Produto 1:N (Contém)

RELACIONAMENTO	ATRIBUTOS	LIGAÇÕES
CONTÉM	Número_Pedido Identificação_Produto Quantidade_Produto Número_de_Seqüência_no_Pedido	Muitos Com Pedido (Número_Pedido) Muitos com Produto (Identificação_Produto)

Figura 10.1

Quando comerçamos a descrever um jogo básico de entidades, irão existir atributos que, na realidade, nos mostram que pertencem não às entidades já definidas, mas sim, que são caracterizadores de um objeto ainda não determinado por nós.

E quais seriam os relacionamentos existentes entre o restante das entidades até aqui descobertas?

Vamos recordar que o importante em modelagem de dados é retratar o mundo real, logo a interpretação da sentença: Cliente faz Pedido, nos apresenta além de duas entidades, mais um fato, que também é um objeto deste contexto, o ato de fazer um pedido. Faz é o relacionamento entre a entidade Cliente e a entidade Pedido, e efetua-se por intermédio de referência lógica em pedido a um cliente.

Observa-se que em Pedido temos o atributo "Identificação_do_Cliente", que caracteriza uma chave estrangeira em Pedido, permitindo uma expressão de comparação: Cliente.Identificação_de_Cliente = Pedido.Identificação_de_Cliente.

Recordando o capítulo de relacionamentos, a chave estrangeira está em Pedido, logo o relacionamento possui uma cardinalidade de Um-para-Muitos no sentido de Cliente para Pedido, isto é, um Cliente Faz Muitos Pedidos (figura 10.2).

Figura 10.2

Na figura 10.3 é apresentada uma simulação das tabelas Cliente e Pedido.

CLIENTE

IDENTIFICAÇÃO_CLIENTE	NOME_CLIENTE
2412	Atacado José
2122	Mercado Pedro
5577	Banca Mauricio
2901	Lojas du Pedter

PEDIDO

NÚMERO_ PEDIDO	DATA_PEDIDO	VENDEDOR	IDENTIFICAÇÃO_CLIENTE
7709/92	21/02/92	76	2122
5588/92	23/04/92	55	5577

Figura 10.3

Seguindo a linha de raciocínio, analisemos agora as ligações possíveis de Vendedor com outras entidades.

Vendedor está relacionado com o quê?

A primeira sentença do caso em estudo, retrata-nos o vendedor tirando os pedidos, realizando o atendimento a um cliente.

Mas o que é neste contexto um fato relevante? Tirar o pedido ou Atender a um Cliente?

Seguidamente, em treinamentos realizados de modelagem de dados, os nosso alunos apresentam a tendência a efetivar uma ligação entre vendedor e cliente.

Perguntamos então se esta ligação se efetiva diretamente, ou necessita de um meio para realizar-se?

A efetivação desta ligação no mundo real realiza-se através de um meio, através de um Pedido. Logo, as ocorrências da entidade Vendedor estão relacionadas com as ocorrências da entidade Pedido, este relacionamento sim, devemos efetivar no modelo.

Para obtermos o vendedor que atendeu a um cliente, devemos obviamente, verificar os pedidos de um cliente, para nos pedidos obter quem o atendeu.

Analisando o modelo até este ponto trabalhado, temos então os seguintes objetos delineados:

- 4 Entidades:
  - Cliente;
  - Pedido;
  - Produto;
  - Vendedor.

- 3 Relacionamentos:
  - Faz;
  - Tira;
  - Contém.

Sendo este último um relacionamento com campos, que representa os itens de um pedido.

O modelo de dados representativo da análise realizada é composto então por um Diagrama de Entidades e Relacionamentos um descritivo de cada entidade com seus atributos, e os relacionamentos existentes entre estas entidades, com seus atributos e expressões de relacionamento (Dicionário de Dados).

A figura 10.4 procura mostrar ao leitor o diagrama ER resultante da análise realizada.

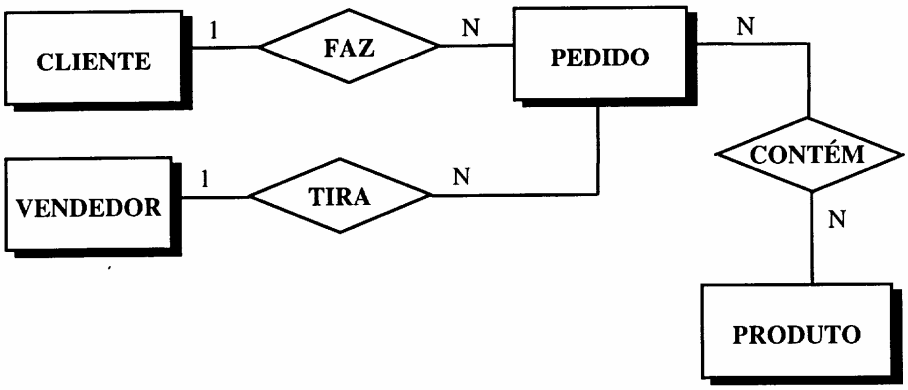


Figura 10.4

A construção de uma simulação é importante para que possamos obter o entendimento e a validação do modelo que estamos definindo. Ao visualizar as tabelas de dados, temos então o domínio do problema simulado, permitindo-nos correções por interpretação de contexto.

Na figura 10.5 é apresentado uma simulação da realidade, com valores colocados nas tabelas criadas.

CLIENTE

IDENTIFICAÇÃO_CLIENTE	NOME_CLIENTE
2412	Atacado José
2122	Mercado Pedro
5577	Banca Mauricio
2901	Lojas du Pedter

PEDIDO

NÚMERO_ PEDIDO	DATA_PEDIDO	VENDEDOR	IDENTIFICAÇÃO_CLIENTE
7709/92	21/02/92	76	2122
5588/92	23/04/92	55	5577
8936/94	20/07/94	97	2901
2738/95	01/11/95	17	2412
4976/95	15/12/95	76	2901

Figura 10.5

VENDEDOR

NUM_VENDEDOR	NOME_VENDEDOR
55	Santos Sílvio
76	Braúlio
97	Felipe
17	Jorge

PRODUTO

NÚMERO_PRODUTO	NOME_PRODUTO
111	Camisa
256	Rádio
387	Pomada
358	Garrafa
470	Tijolo
631	Bicicleta

CONTÉM

NÚMERO_PEDIDO	NÚMERO_PRODUTO	QUANTIDADE
7709/92	111	100
7709/92	387	20
5588/92	256	2
8936/94	256	3
2738/95	358	400
2738/95	470	1000
2738/95	631	4
4976/95	256	1

Figura 10.5 (continuação)

Como no primeiro momento da modelagem o que nos importa é determinar quais são as entidades que coexistem no modelo, alguns dados são colocados à parte, divididos em grupos similares quanto ao tipo de informação que possivelmente irão compor.

Por exemplo, em nosso caso, podem surgir dados como comissão do vendedor por suas vendas, expressos por um valor financeiro totalizador de um mês específico, reserva de mercadorias para as vendas já colocadas ou as condições de pagamento concedidas ao cliente para o pedido que ele está realizando.

Podem, eventualmente, surgir adjetivos de qualificação genéricos que não devem nos induzir a criar novas entidades, já que identificam grupos seletivos de uma mesma entidade já delineada, tais como: o fato de que pedidos podem ficar pendentes. Pedidos Pendentes é uma adjetivação de um pedido, um status do pedido, não caracterizando em nenhuma hipótese uma nova entidade, já que não tem uma existência própria, desvinculada como um objeto distinto de Pedido, sendo na realidade uma especialização de Pedido, como já vimos.

Historicamente, os nossos usuários quando descrevem seu ambiente operacional, apresentam-nos na realidade, não uma foto do setor, mas sim um filme do mesmo em movimento.

Não existe a preocupação dos mesmos sobre o que está efetivamente participando, informando-nos seguidamente distorções, principalmente quanto aos dados por eles processados.

As técnicas de normalização de dados são extremamente úteis, como estudaremos no capítulo 12, mas a forma ideal de utilização seria somente a partir do reconhecimento e identificação de um grupo principal de entidades, não sendo normalização um conjunto de regras fundamental para a identificação de entidades.

A questão maior é a condução do processo de elaboração do chamado modelo descritivo de sistema.

Devemos criar atas de reuniões com usuários? Ou existe uma forma de trabalho que nos leve diretamente a modelos de dados?

A análise de abstrações apresentada não caracteriza a necessidade de longas reuniões, mas possui, isto sim, a característica de ser uma atividade a ser realizada em conjunto com os usuários desde o início da atividade de projeto de sistemas.

Voltando a lembrar, este é o primeiro passo efetivo do trabalho de análise, só sendo suplantado pela apresentação formal das pessoas que executam o papel de usuários, para fins de relações interpessoais.

Mas para que este livro não fique parco de casos, vamos simular outras situações de modelagem para o passo de descoberta de entidades, de forma que possam, estas interpretações de análise, serem aproveitadas como experiências de modelagem pelos leitores. Mas como relacionamentos são

objetos que aparecem conjuntamente com as entidades, passaremos então a tratar de forma global, os casos propostos e resolvidos.

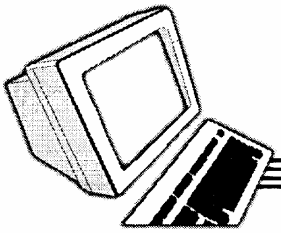
No exemplo que até aqui utilizamos, aplicamos a forma *top-down* da visão dos dados de um contexto, baseados na atividade-fim, para a qual existia o setor da empresa.

Denominamos de entidade principal de um sistema, aquela que possui a característica de representar numa análise de abstrações *top-down*, o nível mais alto de abrangência.

É fator primordial para um resultado eficiente de modelagem de dados, que se identifique claramente esta entidade (ou entidades) com os objetivos organizacionais do setor, para elaborar-se um sistema, pois poderemos desta forma, conduzir um projeto visando cumprir os objetivos e metas de um negócio.

No próximo capítulo, vamos analisar uma série de casos propostos e resolvidos, tratando de situações mais abrangentes, simulando casos reais.

Capítulo



Estudo de Casos 11

11.1 - Introdução

Esta é a parte que consideramos fundamental neste livro. É composta de três casos práticos que estão baseados em fatos verídicos, situações que podem perfeitamente ser encontradas no mundo real, e, é claro, nós as encontramos.

As soluções serão comentadas passo a passo, procurando transmitir o processo de levantamento que utilizamos na vida prática e nos treinamentos realizados por nós. Será criada uma proposta básica de diagrama Entidade-Relacionamento para cada situação.

Em modelagem de dados, não existe um modelo do gênero "A resposta é esta, e somente esta". Como estamos trabalhando com a visão que seres humanos têm dos dados de uma realidade, eles podem ter diferentes interpretações para um mesmo fato, e como conseqüência, representações diferenciadas podem e certamente vão ocorrer.

O objetivo não é restringirmos os resultados, uma vez que desejamos conduzi-los para implementações em ambientes de Banco de Dados diversos.

Vamos apresentar, além do diagrama ER que será construído neste capítulo, a definição da base de dados no SGBD ORACLE v. 7® e Microsoft® Sql Server™ 7.0 utilizando a linguagem SQL (Strutured Query Language). Esta definição das bases de dados será apresentada no final do capítulo 14.

A apresentação dos resultados será narrativa, objetivando que a visão dos dados cresça gradativamente na interpretação dos fatos em um problema.

11.2 - Estudo de Caso 1

Controle de Cinemas

Vamos iniciar nossos estudos de caso com uma situação que envolve diversos níveis de abstração, possibilitando a utilização de extensões do modelo Entidade-Relacionamento.

Vamos projetar um modelo de dados que atenda às necessidades de controle dos cinemas e filmes em uma determinada empresa de distribuição de filmes.

Visão do negócio:

- Área de Negócio: Departamento de Programação;

- Função Gerencial: Administração de Cinemas.

Após várias reuniões com os futuros usuários do sistema, relacionamos uma série de regras do negócio e que serão a base para o desenvolvimento do diagrama ER:

- A empresa de distribuição possui vários cinemas em diversas localidades;

- Cada cinema possui uma identificação única, um nome fantasia, um endereço completo, incluindo rua, avenida, bairro, município, estado e sua capacidade de lotação;

- Os filmes podem ser dos mais variados tipos e gêneros;

- Cada filme é registrado com um título original, e se for filme estrangeiro, possuirá também o título em português, o gênero, sua duração, sua impropriedade e seu país de origem, informações sobre os atores que compõem seu elenco, e o seu diretor. Existirá um único diretor para cada filme;

- Alguns cinemas apresentam mais de um filme em cartaz, sendo, nesses casos, sessões alternadas com um filme e outro;

- As sessões possuem horários que variam de acordo com a duração do filme, havendo sempre um intervalo de aproximadamente 15 minutos entre elas;

- Os atores de um filme podem, obviamente, atuar em diversos filmes, assim como o diretor de um filme pode também ser ator nesse filme ou, ainda mais, ser ator em outro filme. Um ator possui as seguintes características: um número de identificação, um nome, uma nacionalidade e uma idade;

- As sessões de cinema devem ter seu público registrado diariamente, para que se permita a totalização dos assistentes quando o filme sair de cartaz, ou a qualquer instante.

Nas reuniões de levantamento, os usuários nos passaram as seguintes necessidades de informação:

- Apuração do público por município, por cinema e por sessão de cada cinema;

- Permitir uma forma de consulta, que dado um determinado ator, sejam localizados os cinemas onde estão em cartaz os filmes em que esse ator atua;

  ex: "Quais cinemas (nomes) passam filmes em que atue a atriz "Júlia Roberts"? Queremos obter somente os nomes dos cinemas, independentemente dos filmes." Para testarmos esta solicitação utilizaremos comandos SQL.

- Em quais cinemas está sendo exibido um determinado gênero de filme;

- Em quais cinemas estão sendo exibidos filmes nacionais.

Então vamos começar nossa análise dos objetos da realidade:

- Qual(is) o(s) objeto(s) que se destaca(m) no mundo real?

- Existe um objeto, uma abstração global, ou mais de uma?

A resposta a estas questões é fundamental, pois é a partir dela que iremos iniciar o desenvolvimento do modelo.

Vejamos então.

Temos claramente duas classes de objetos que podemos considerar como grandes entidades neste contexto: **Filme** e **Cinema**.

Filme possui uma série de atributos que o caracterizam, além do que podemos visualizar que existem muitas ocorrências de filme em nosso caso.

Cinema, da mesma forma, já que possui atributos próprios que o caracterizam, além de ser fácil imaginar a existência de diversos cinemas.

Mas neste ponto, os leitores irão perguntar: E daí, o que é possível montar com o obtido até este momento?

Antes de seguirmos desenhando entidades "alopradamente", vamos analisar a caracterização de cada uma delas, em relação ao contexto que nos foi apresentado.

Cinema tem os seguinte atributos:

CINEMA

Identificação do cinema
Nome fantasia
Endereço
  Rua/Avenida
  Bairro
  Município
  Estado
Capacidade de lotação

Vejamos quais seriam os atributos possíveis para Filme, de acordo com o contexto apresentado:

FILME
Identificação do filme Título Original Título em português Duração Identificação do Diretor País de origem Gênero Impropriedade Atores (**Quantos**?)

Como podemos observar no quadro, surgiu a informação sobre os atores de um filme. Podemos concluir que temos um objetivo que representará os atores de um filme, ou seja, o seu elenco.

Desta forma, como está afirmado no contexto do problema que um ator pode atuar nos mais diversos filmes, e possui atributos independentes de participação em filmes, caracteriza uma entidade para armazenarmos informações sobre os atores.

Neste instante, temos as entidades:

FILME	CINEMA	ATOR

Com os objetos que já identificamos, o procedimento mais correto para o trabalho de modelagem é, neste instante, efetuarmos a análise dos relacionamentos que são possíveis de existir no mundo real entre estas entidades.

Vamos então ao cinema, para entendermos melhor o mundo real onde as coisas acontecem.

A primeira premissa que podemos utilizar para ligarmos entidades é a de que um Filme passa em um Cinema. Neste momento, caracterizamos uma existência no mundo real.

Na seqüência de raciocínio, sabemos que um mesmo Filme passa em vários Cinemas ao mesmo tempo; logo, temos um relacionamento entre as entidades Cinema e Filme, mas qual será sua cardinalidade correta?

Ora, existe no contexto das regras de negócio apresentado a informação de que um cinema pode apresentar mais de um filme em cartaz. Por outro lado, sabemos que quando estamos modelando, não estamos preocupados com o processo de informação; logo, um cinema pode passar vários filmes em datas diferentes, caracterizando-se então que um cinema passa mais de um filme.

Com esta análise podemos desenhar então um bloco de modelo com as entidades Filme e Cinema, e seu relacionamento, que é então um relacionamento de muitos para muitos.

Leia-se o modelo seguinte como:

MODELO INICIAL

Um Cinema passa muitos Filmes — Um Filme passa em muitos Cinemas

Já estudamos que quando nos defrontamos com cardinalidades de Um-para-Muitos nos dois sentidos de leitura do modelo, o relacionamento é na realidade Muitos-para-Muitos.

Mas devemos seguir analisando o restante dos objetos até aqui descobertos.

A questão que se coloca em seguida, são os relacionamentos que existem entre o que já delineamos do modelo e a entidade Ator, que contém os dados de cada ator registrado como tal.

Ator está ligado com Cinema?

Somente se os atores comparecerem em cada cinema que passa o Filme.

Afirma-se que um filme tem um elenco de atores, isto é, muitos atores em um Filme, tornando-se óbvio que o ator relaciona-se com filme exclusivamente.

Mas com que cardinalidade?

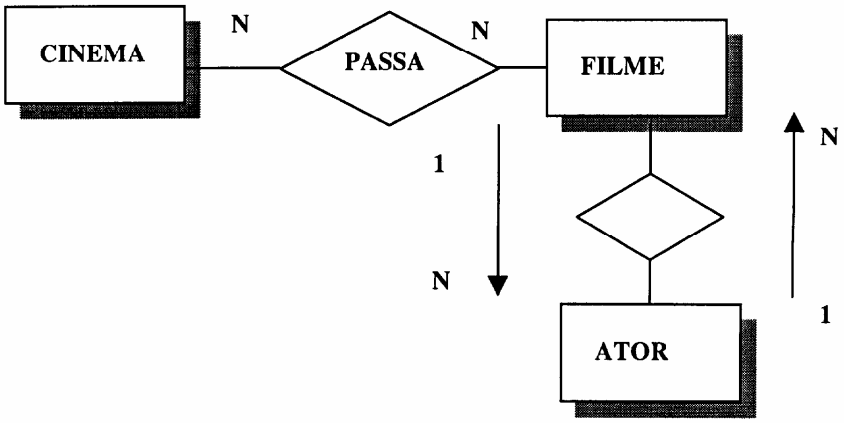


Figura 11.1

Um Filme possui muitos Atores, o que nos fornece a cardinalidade de Um-para-Muitos no sentido Filme - Ator. Por outro lado, mudando nosso posicionamento para a entidade Ator, temos: um determinado Ator (ex: Linda Mulher "Júlia Roberts") atua, ou melhor, relaciona-se com muitos Filmes, no sentido de Ator para Filme. Esta situação está representada na figura 11.1.

Logo, temos novamente a situação que já havíamos encontrado (Cinema / Filme), um relacionamento entre Filme e Ator com cardinalidade Muitos-para-Muitos.

E que campos poderia ter este relacionamento?

Vamos deixar para um pouco mais adiante na solução deste exercício, o delineamento dos campos deste relacionamento.

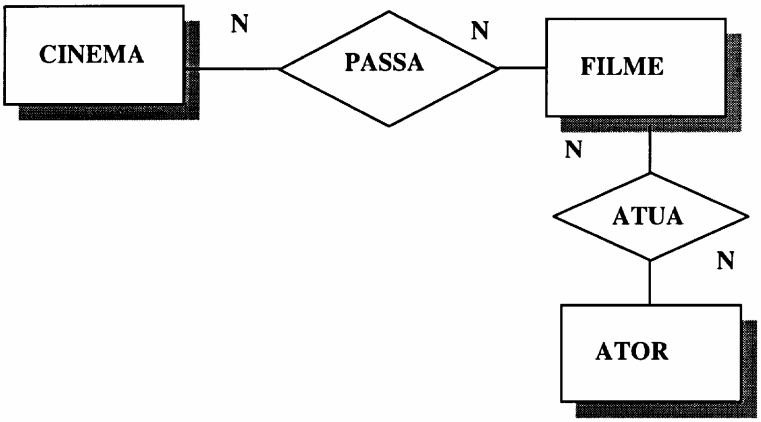


Figura 11.2

Já temos então o modelo agora com três entidades e dois relacionamentos, conforme nos mostra a figura 11.2.

Vamos analisar o contexto que diz que um filme possui um diretor, e esse diretor pode também ser um ator, e por conseqüência possui os mesmos dados cadastrais de ator.

Existe então, mais um relacionamento entre Filme e Ator?

Qual a sua cardinalidade?

Quando é afirmado que um ator pode também ser um diretor, estamos criando um sinônimo simples para ator, denominado Diretor, que será o papel

exercido por uma ocorrência da entidade Ator, quando ela estiver relacionada com Filme por meio de um relacionamento chamado Dirige, que é distinto do relacionamento Atua.

No conjunto de atributos de Filme, consta um atributo desde o início de nosso caso, que é a identificação do diretor, o que caracteriza uma chave estrangeira em filme, referenciando uma outra entidade, no caso a entidade Ator.

Criamos então o relacionamento DIRIGE, que é um fato puro que ocorre no mundo real; todos os filmes são dirigidos por alguém. No diagrama ER, criamos um papel para a entidade ATOR, chamado DIRETOR. Estabelecemos então, que relacionamento DIRIGE existe entre a entidade FILME e a entidade DIRETOR, que é na verdade a mesma coisa que ATOR.

Na figura 11.3, é apresentado o modelo ER com as considerações apresentadas a respeito de ATOR, DIRETOR e FILME.

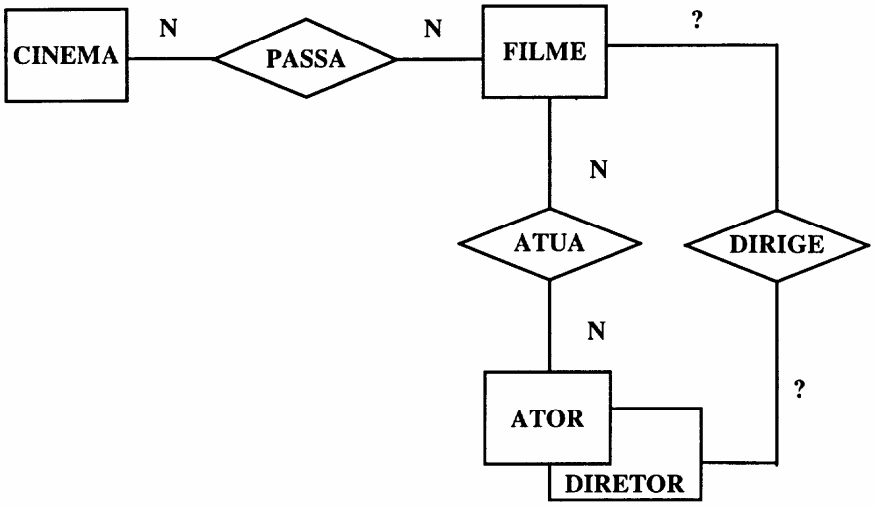


Figura 11.3

A cardinalidade tornou-se óbvia em função de existir uma chave estrangeira na entidade FILME, pois conforme já havíamos citado neste livro, a chave estrangeira está do lado muitos. Na figura 11.4, é apresentado o modelo ER com as cardinalidades ATOR/FILME.

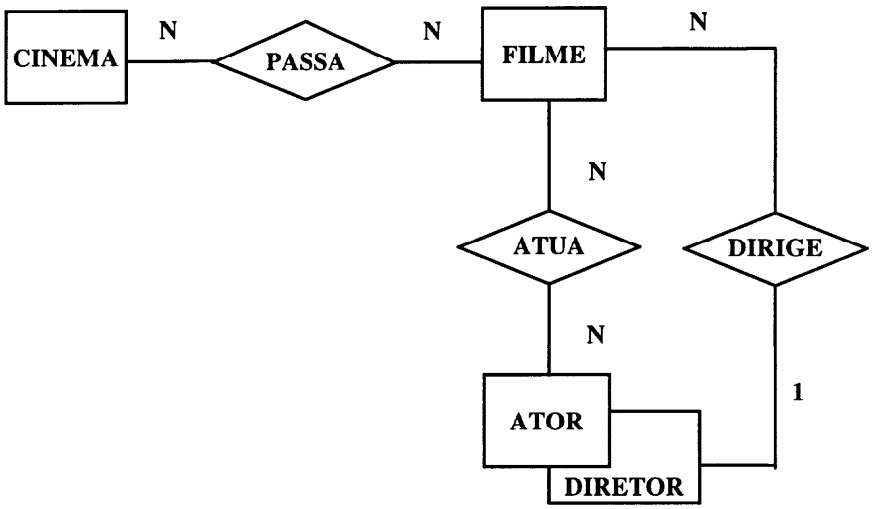


Figura 11.4

Pode-se perguntar, por que estabelecemos uma cardinalidade de Um--para-Muitos, baseados em que princípios? Somente na chave estrangeira?

Não poderia ser um relacionamento de Um-para-Um?

Existe, nas premissas do caso em estudo, uma afirmativa de que cada filme somente possui um diretor, mas nada que limite um diretor somente dirigir um único filme; logo, a cardinalidade é Um-para-Muitos.

Bem, já possuímos uma boa parte da solução para a modelagem solicitada. Vejamos então o que nos falta.

É solicitado o controle de público que assistiu a um determinado Filme. Essa informação deve ser possível de obter tanto em nível de cinema, quanto em nível de cidade, e o público do filme como um todo.

Vamos recordar que quando vamos a um cinema para assistir a um Filme, vamos a um Cinema que Passa um Filme.

Então, estando em frente ao Cinema que Passa um Filme, vemos que quando o Cinema Passa um Filme, existem sessões para assistirmos.

Logo, iremos fazer parte do público que assistiu a um determinado filme num cinema específico, em uma localidade qualquer.

Vamos analisar um objeto que nos parece interessante, a SESSÃO de cinema que passa um filme.

O objeto SESSÃO só existe se antes existir um CINEMA que PASSA um FILME, ou seja, deve existir um Filme relacionado com um cinema para que exista uma ou mais sessões.

Isto nos leva a modelar dados, ou seja, atributos para sessão inicialmente:

SESSÃO
Data da Sessão Hora da Sessão Público da Sessão

Mas com esses atributos somente, não poderíamos identificar em que cinema foi realizada essa sessão e nem para que filme ela aconteceu.

Logo, necessitamos relacionar cada ocorrência de Sessão com um Filme passando em um Cinema, isto é, com o bloco de modelo que representa o fato CINEMA PASSA FILME.

Um CINEMA quando PASSA um FILME tem muitas SESSÕES, e por outro sentido de leitura dos fatos, uma SESSÃO só está relacionada a um FILME em um CINEMA.

Isto nos apresenta a necessidade do uso de uma agregação, uma extensão do modelo E-R, que estudamos no capítulo 8. A figura 11.5 representa esta situação do uso de agregação.

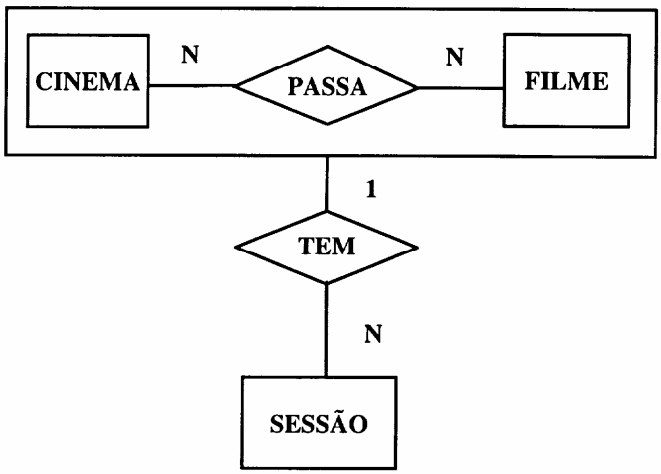


Figura 11.5

Agora já podemos fechar o modelo ER, o qual pode nos permitir visualizar todos os fatos que envolvem o caso prático. Vejamos a figura 11.6, e vamos executar a sua leitura.

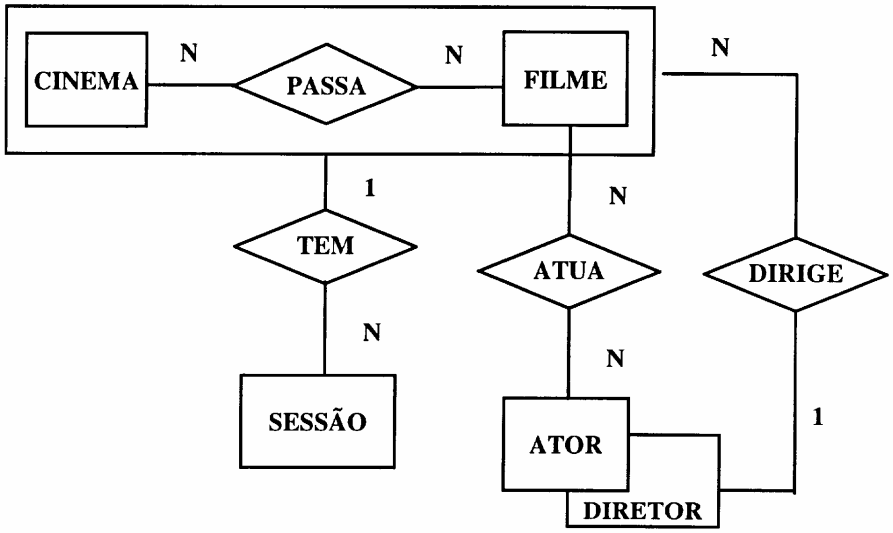


Figura 11.6

Um cinema passa muitos filmes, e um filme passa em muitos cinemas.

Quando um filme passa em um cinema, este tem muitas sessões.

Cada filme possui atores que participam dele, e por sua vez um ator pode participar de muitos filmes.

Todo filme tem um diretor que pode ser também um ator desse filme. Um diretor pode dirigir muitos filmes.

Na figura 11.7, são apresentados os atributos de cada objeto.

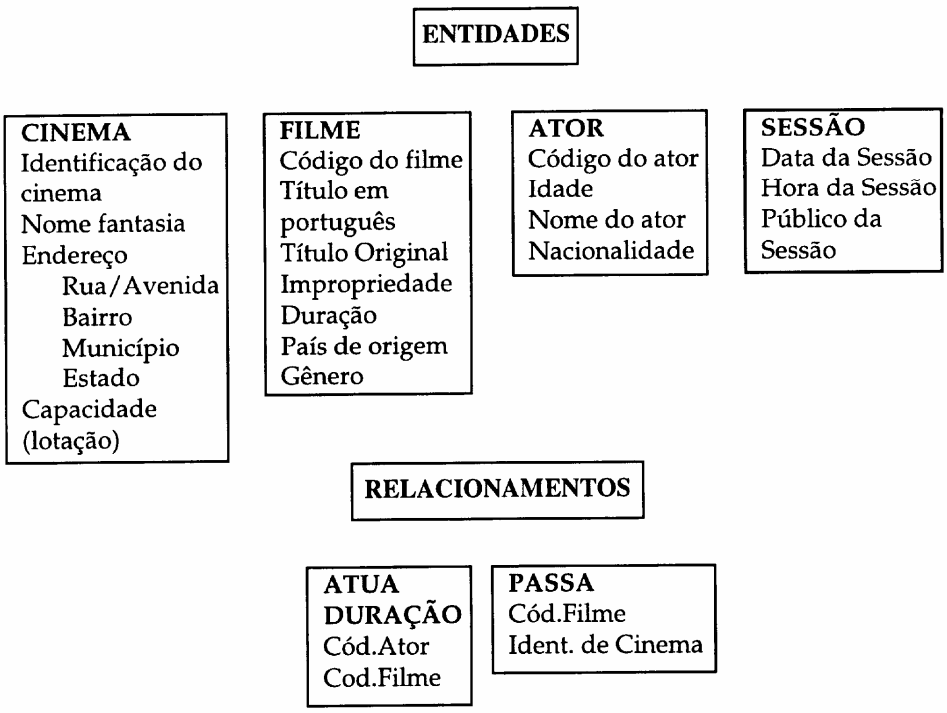


Figura 11.7

O modelo criado já atende de forma satisfatória às necessidades de informação da Distribuidora de Filmes. Com base neste modelo podemos, a partir de agora, aprofundar nossas observações a respeito da realidade enfocada, tentando obter uma visão cada vez mais detalhada a respeito dos objetos do negócio.

Se fixarmos a atenção na entidade FILME, podemos observar que existe um agrupamento de nacionalidade dos filmes: filme nacional e filme estrangeiro. Observando mais atentamente a entidade FILME, podemos observar também que a ocorrência de Gênero pode ter muitas repetições. Esta particularidade sobre redundâncias será tratada no capítulo 12 que discorre

sobre Normalização. Com estas duas observações podemos melhorar o modelo ER proposto anteriormente (figura 11.8).

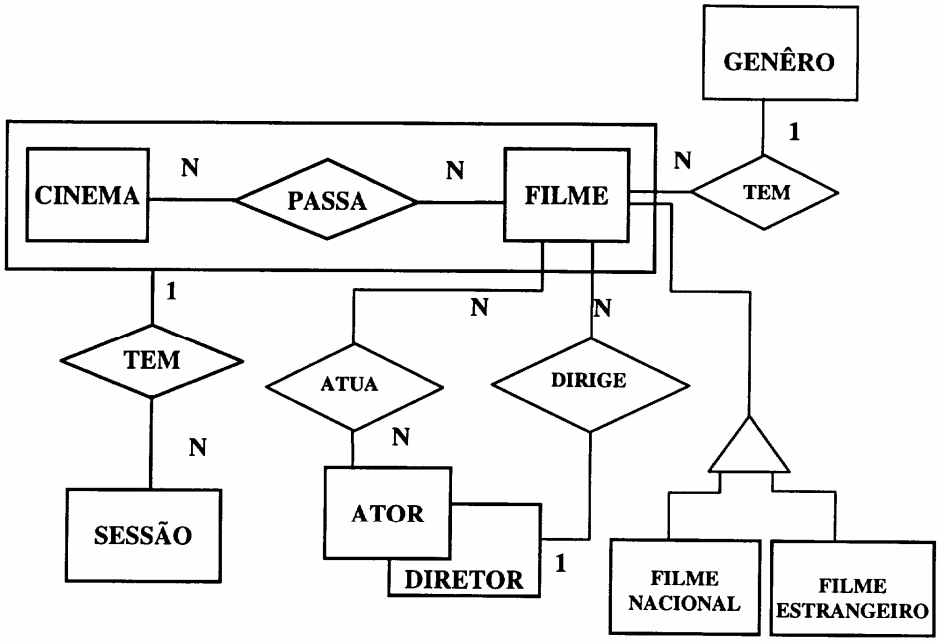


Figura 11.8

Na figura 11.9, são apresentados os atributos dos novos objetos.

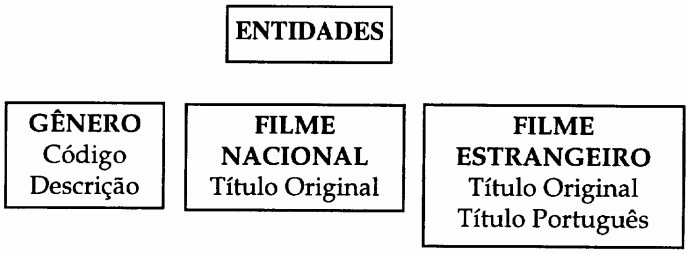


Figura 11.9

Fica bem evidente que para a implementação deste modelo alguns aspectos poderiam ser desconsiderados, tais como a ocorrência dos subtipos: filme nacional e filme estrangeiro, devido a sua simplicidade de atributos.

11.3 - Estudo de Caso 2

Gerência Acadêmica de uma Universidade

Decidiu-se automatizar alguns procedimentos da Gerência Acadêmica (GA) da Universidade UNITESTE. Com a finalidade de auxiliar esta tarefa, foi solicitado o desenvolvimento de um banco de dados.

A Gerência Acadêmica mantém um controle centralizado de alunos, cursos, disciplinas, turmas de matérias, professores e histórico escolar de alunos.

Os alunos são admitidos nos cursos por meio de um vestibular ou transferência, e um aluno só pode estar ligado a um curso, em um dado instante. Os alunos, quando ingressam na universidade, preenchem uma ficha cadastral (com número de matrícula pré-impresso) com nome e endereço.

De acordo com as normas (estatutos) da UNITESTE, cada disciplina para ser oferecida, necessita de um mínimo de dez alunos e para que o alto padrão de ensino oferecido seja mantido, cada disciplina deverá ter no máximo 50 (cinqüenta) alunos. Os cursos são compostos por disciplinas, as quais podem ser obrigatórias ou optativas, dependendo do curso a que pertencem. Cada disciplina está sob a responsabilidade de um departamento da universidade, e é codificada de acordo com um padrão preestabelecido pelo conselho.

Segundo uma convenção adotada pela UNITESTE, os professores podem ser cadastrados na GA sem estar lecionando uma disciplina. Cada professor pode ministrar até o máximo de 3 (três) matérias. Para que um professor ministre uma disciplina, o mesmo, deve estar devidamente habilitado pelo CFE (Conselho Federal de Educação). Cada professor está vinculado a um departamento e possui um código específico para sua diferenciação dentre os demais professores.

Para o perfeito acompanhamento acadêmico do aluno durante o curso, a UNITESTE possui um histórico escolar. Esse documento é o conjunto de todas as disciplinas cursadas pelo aluno em toda a sua vida acadêmica dentro da UNITESTE. Contém o registro das disciplinas e indica a nota (conceito) final e a data em que a disciplina foi cursada.

Os departamentos são responsáveis pelos cursos de suas áreas de atuação. As responsabilidades envolvem a definição do número de créditos exigidos para a conclusão do curso, o número total de horas exigidas para o curso e o número total de horas nas disciplinas obrigatórias.

A UNITESTE adota um sistema progressivo de aprendizado, no qual cada disciplina pode ter no máximo 3 (três) e no mínimo 0 (zero) pré-requisitos. Geralmente, as matérias sem nenhum pré-requisito, ou estão no primeiro período, ou são disciplinas eletivas (não obrigatórias).

De acordo com o Conselho Acadêmico, um aluno pode, em um dado semestre, não estar matriculado em nenhuma disciplina, caracterizando um trancamento de matrícula. Em um período letivo, um aluno pode se matricular, no máximo, em 7 (sete) disciplinas. O conselho também fixou que um aluno pode repetir no máximo 3 (três) vezes a mesma disciplina.

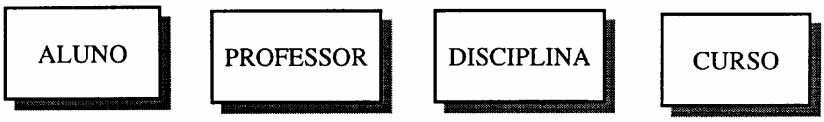
No total, a UNITESTE pode comportar 5000 (cinco mil) alunos matriculados em seus diversos cursos. A cada ano, são admitidos 800 (oitocentos) novos alunos via vestibular e as transferências externas podem ser no máximo 60 (sessenta). Formam-se em torno de 300 (trezentos) alunos por semestre. A UNITESTE oferece 10 cursos e 280 (duzentos e oitenta) disciplinas, possuindo cerca de 120 (cento e vinte) professores.

Visão do negócio:

- Área de Negócio: Gerência Acadêmica;

- Função Gerencial: Controle Administrativo.

Por meio da análise das informações obtidas no levantamento, podemos observar claramente algumas classes de objetos que podemos considerar como centrais para as necessidades da Gerência Acadêmica: Alunos, Cursos, Disciplinas e Professores.



Cujos atributos são:

ALUNO	PROFESSOR	DISCIPLINA	CURSO
Número de Matrícula	Código do Professor	Código da Disciplina	Código do Curso
Nome	Nome	Nome da Disciplina	Nome do Curso
Endereço	Inscrição CFE	Descrição curricular	**Departamento**
Rua/Avenida	**Departamento**	**Departamento**	
Bairro			
Município			
Estado			

A partir desta análise preliminar das classes de objetos, podemos então traçar os relacionamentos (fatos) que interligam estas entidades. Podemos retirar do levantamento feito os seguintes relacionamentos:

- ALUNO Cursa DISCIPLINA: um aluno cursa de 0 a N (máx. 7) disciplinas; e uma disciplina pode estar sendo cursada por 0 a N (min. 10 / máx. 50) alunos;

- ALUNO Realizou DISCIPLINA (histórico escolar): um aluno realizou de 0 a N disciplinas; e uma disciplina pode ter sido realizada por 0 a N alunos. O fato da realização só pode ocorrer no máximo 3 vezes (repetição) para a mesma ligação entre um aluno e uma disciplina;

- PROFESSOR Rege DISCIPLINA: um professor pode reger nenhuma, uma ou várias disciplinas; e uma disciplina pode ter a regência de nenhum, um ou vários professores;

- PROFESSOR Possui-Habilitação DISCIPLINA: um professor deve ter pelo menos uma habilitação de disciplina para poder pertencer aos quadros da UNITESTE; e uma disciplina pode ter nenhum, um ou vários professores habilitados;

- ALUNO Pertence a CURSO: um aluno só pode pertencer a um curso; e um curso pode permitir a matrícula de vários alunos.

Com base nas observações descritas anteriormente, podemos construir o diagrama ER preliminar, apresentado na figura 11.10.

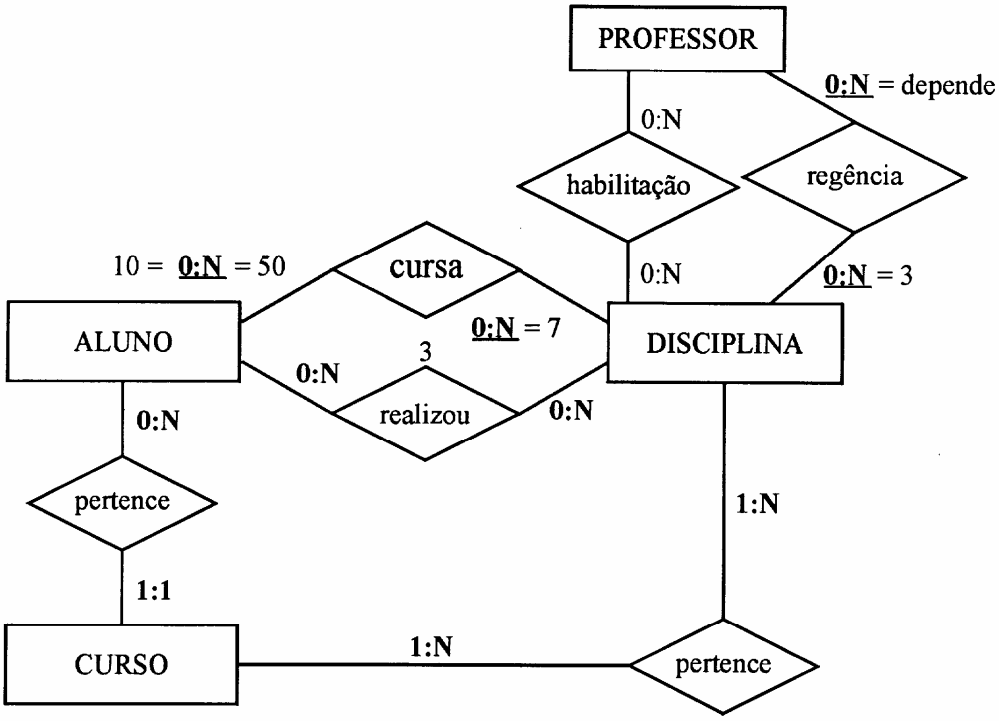


Figura 11.10

Observe que em algumas cardinalidades foram colocados os valores reais (mínimos e máximos). Na vida prática, algumas ferramentas automatizadas de modelagem (CASE) permitem a colocação destes valores para uso futuro, principalmente em prototipadores. **No diagrama, estes valores não devem aparecer.**

Observando mais atentamente os conjuntos de atributos das várias classes de objetos até aqui levantadas, podemos notar que existe um elemento comum a pelo menos três delas (CURSO, DISCIPLINA e PROFESSOR): o atributo Departamento. Este fato nos leva à conclusão de que Departamento também é uma classe de objetos a ser controlada pela Gerência Acadêmica. Com isso vai aparecer uma entidade chamada DEPARTAMENTO, que possui os seguintes relacionamentos:

- **DEPARTAMENTO Controla CURSO:** um departamento pode controlar de 1 até N cursos; e um curso só é controlado por um e somente um departamento;

- **DEPARTAMENTO é Responsável DISCIPLINA:** um departamento é responsável por uma ou várias disciplinas; e uma disciplina é de responsabilidade de um e somente um departamento;

- **PROFESSOR está Ligado DEPARTAMENTO:** um professor está ligado a um e somente um departamento; e um departamento pode possuir vários professores.

Além do aparecimento da entidade DEPARTAMENTO, a realidade mostra-nos que uma DISCIPLINA está ligada a outra por uma relação de pré-requisito. Essa ligação caracteriza um auto-relacionamento com dois papéis bem-definidos: exige e satisfaz.

A partir destes novos objetos podemos construir o diagrama final (figura 11.11) para a situação apresentada para o controle da Gerência Acadêmica.

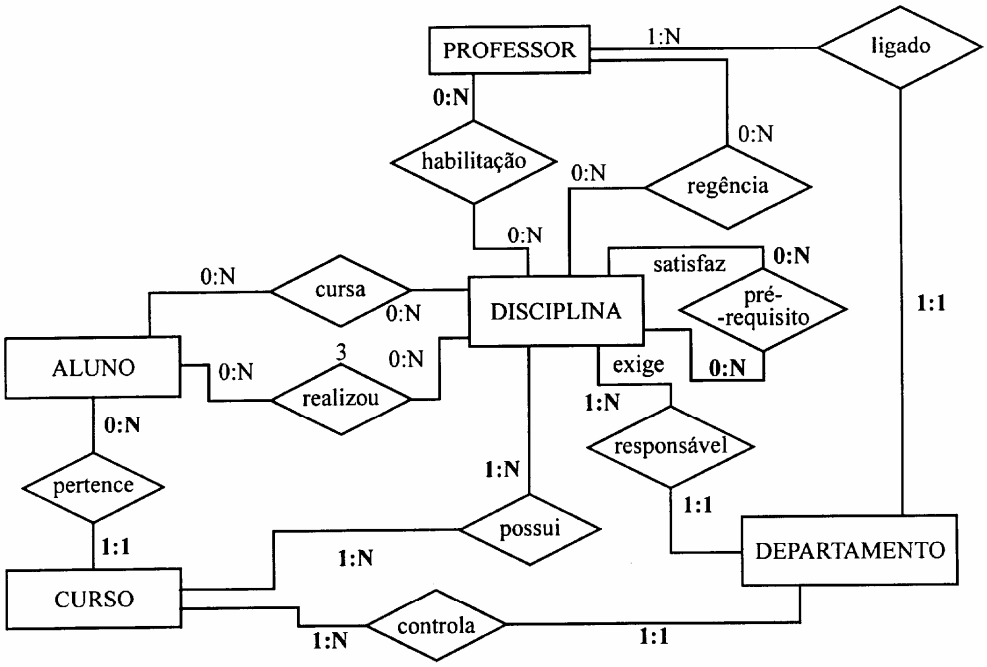


Figura 11.11

O conjunto de atributos deste modelo é apresentado em seguida:

ENTIDADES

ALUNO
Número de Matrícula
Nome
Endereço
Rua/Avenida
Bairro
Município
Estado
Código do Curso

PROFESSOR
Código do Professor
Nome
Inscrição CFE
Cód. Departamento

DEPARTAMENTO
Cód. Departamento
Nome do Depto.

CURSO
Código do Curso
Nome do Curso
Cód. Depto.

DISCIPLINA
Código da Disciplina
Nome da Disciplina
Descrição curricular
Cód. Departamento

RELACIONAMENTOS

POSSUI
Código da Disciplina
Código do Curso
Norma do CFE

CURSA
Núm. de Matrícula
Código da Disciplina
Nota final

REALIZOU
Código da Disciplina
Núm. de Matrícula
Média
Período

PRÉ-REQUISITO
Código da Disciplina
Cód. Disc. Pre-req.
Norma do CFE

HABILITAÇÃO
Cód. do Professor
Cód. da Disciplina
Data da Habilitação

REGÊNCIA
Cód. do Professor
Cód. da Disciplina
Período
Avaliação

11.4 - Estudo de Caso 3

Grupo de Pesquisa sobre Vírus

Um grupo de pesquisa médica de um grande hospital deseja construir e manter um banco de dados sobre todas as publicações relativas a certos tipos de vírus.

A informação registrada sobre cada vírus inclui o nome científico e um texto livre para sua descrição científica. Cada publicação é impressa em uma edição particular do jornal científico do hospital, identificado pelo nome do jornal, o número do volume e o número da edição.

Uma publicação pode ter um ou mais autores e ser referente a um ou mais tipos de virose.

O resumo (*abstract*) da publicação também é armazenado no banco de dados, junto com o nome do autor (autores) e o nome da instituição (instituto) à qual a pesquisa está associada, caso esta seja de fora do grupo de pesquisas.

Cada publicação contém uma lista de referências a outras publicações e essa informação é registrada na base de dados.

As publicações editadas pelo grupo de pesquisa, além das informações normais armazenadas para cada publicação, possuem informações a respeito do contrato de pesquisa (número do contrato, valor, data de início e término).

A seguir, são apresentadas algumas das necessidades de informação por parte dos usuários:

- Entrar uma nova publicação com todas as informações;
- Listar os detalhes de todas as publicações relativas a um vírus específico;
- Listar as publicações de um específico autor;
- Listar as publicações associadas a um específico contrato de pesquisa.

Visão do negócio:

- Área de Negócio: Pesquisa Médica;
- Função Gerencial: Controle de Publicações Técnicas.

Utilizando as mesmas práticas realizadas nos dois estudos anteriores, podemos observar a existência de classes de objetos necessárias para o Controle de Publicações Técnicas: Publicação, Vírus e Autor.

A relação entre estes três objetos é mostrada em seguida:

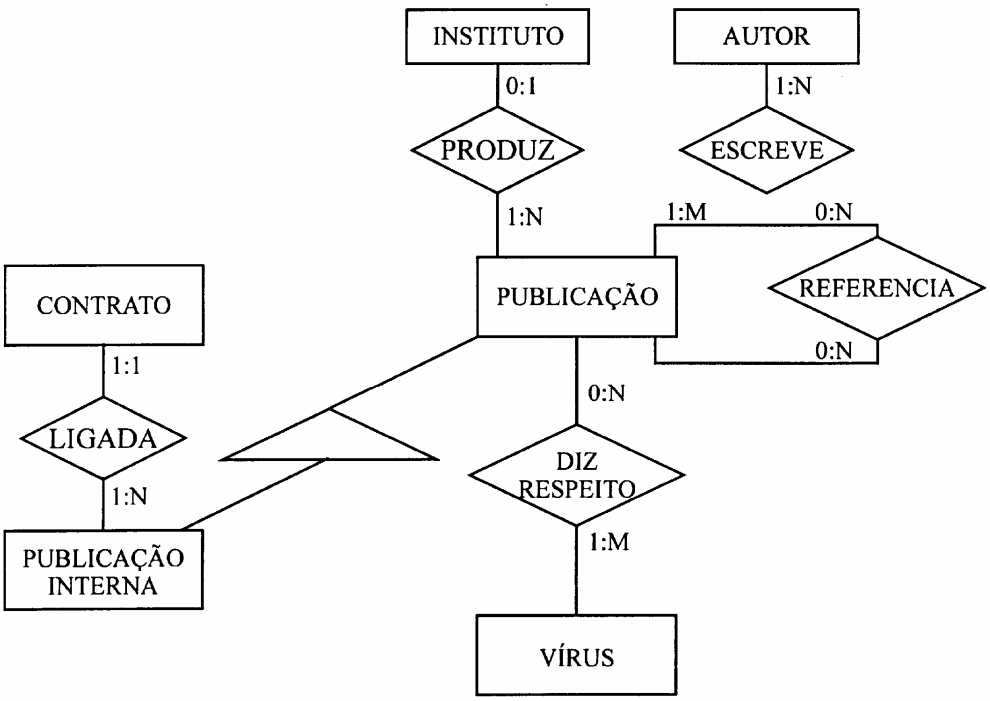
- AUTOR Escreve PUBLICAÇÃO: N:M;

- PUBLICAÇÃO Diz-Respeito VÍRUS: N:M.

A partir destas relações, outros objetos podem ser visualizados, tais como: **instituto** produz pelo menos uma publicação (externa) e **contrato** que financia a publicação interna. Podemos observar que é necessário o particionamento da publicação (externa / interna), porém a publicação externa não será necessário representar, pois ela possui os mesmos atributos das publicações internas. Estas, por sua vez, devem ser representadas, pois apresentam atributos específicos para a ligação com contrato.

As publicações referenciam outras publicações, evidenciando, assim, a ocorrência de um auto-relacionamento (REFERÊNCIA) entre publicações.

Estas análises nos levam à construção do seguinte modelo a seguir.



A seguir, é apresentada a relação de atributos do modelo anterior:

ENTIDADES

PUBLICAÇÃO (chave de publicação, título, chave do instituto, jornal, número do volume, número da edição, ano)

PUBLICAÇÃO_INTERNA (chave de publicação, número do contrato)

AUTOR (chave do autor, nome do autor, nacionalidade, data de nascimento)

INSTITUTO (chave do instituto, nome do instituto, endereço, tipo)

CONTRATO (número do contrato, valor, data de início, data de término)

VÍRUS (chave do vírus, nome do vírus, descrição).

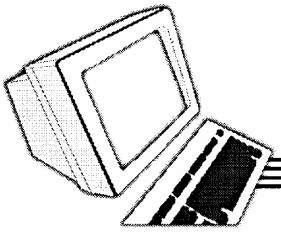
RELACIONAMENTOS

Escreve (chave de publicação, chave do autor, data de entrega)

Referência (chave de publicação, referência de publicação)

Diz_Respeito (chave de publicação, chave do vírus)

Capítulo



Normalização

12

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

Através deste processo pode-se, gradativamente, substituir um conjunto de entidades e relacionamentos por um outro, o qual se apresenta "purificado" em relação às anomalias de atualização (inclusão, alteração e exclusão) as quais podem causar certos problemas, tais como: grupos repetitivos (atributos multivalorados) de dados, dependências parciais em relação a uma chave concatenada, redundâncias de dados desnecessárias, perdas acidentais de informação, dificuldade na representação de fatos da realidade observada e dependências transitivas entre atributos.

Neste capítulo, vamos mostrar como estes problemas podem ser minimizados, sensivelmente, através da normalização, tornando o modelo de dados elaborado bastante estável e sujeito a poucas manutenções.

Os conceitos abordados neste capítulo, podem ser aplicados às duas formas de utilização da normalização:

- sentido de cima para baixo (TOP-DOWN):

Após a definição de um modelo de dados, aplica-se a normalização para se obter uma síntese dos dados, bem como uma decomposição das entidades e relacionamentos em elementos mais estáveis, tendo em vista sua implementação física em um banco de dados;

- sentido de baixo para cima (BOTTON-UP):

Aplicar a normalização como ferramenta de projeto do modelo de dados, usando os relatórios, formulários e documentos utilizados pela realidade em estudo, constituindo-se em uma ferramenta de levantamento.

12.1 - Anomalias de Atualização

Observando-se o formulário de PEDIDO apresentado na figura 12.1, podemos considerar que uma entidade formada com os dados presentes neste documento, terá a seguinte apresentação:

- Atributos da entidade PEDIDO:
  - número do pedido
  - prazo de entrega
  - cliente
  - endereço
  - cidade
  - UF
  - CGC
  - inscrição estadual
  - código do produto (*)
  - unidade do produto (*)
  - quantidade do produto (*)
  - descrição do produto (*)
  - valor unitário do produto (*)
  - valor total do produto (*)
  - valor total do pedido (*)
  - código do vendedor
  - nome do vendedor

(*) Atributos que se repetem no documento

Código do Vendedor: 1791 Nome do Vendedor: Anibal da Silva
Prazo de Entrega: 20 dias

Num. Ped.	Cliente	Endereço Cidade/UF	CGC	IE	Cód. Prod.	Unid.	Quant.	Descrição	Val. Unit.	Tot. Prod.	Tot. do Ped.
3445	TCA.....	R.Meira....	11111111	1111111	45	L	20	álcool	5,00	100,00	1799,00
3445	TCA.....	R.Meira....	11111111	1111111	130	M	2	tecido	20,00	40,00	1799,00
3445	TCA.....	R.Meira....	11111111	1111111	35	Kg	30	farinha	1,00	30,00	1799,00
3445	TCA.....	R.Meira....	11111111	1111111	78	Kg	50	cimento	30,00	1500,00	1799,00
3445	TCA.....	R.Meira....	11111111	1111111	90	L	40	cola	3,00	120,00	1799,00
3445	TCA.....	R.Meira....	11111111	1111111	39	Kg	3	chumbo	3,00	9,00	1799,00
2610	Lopes....	R. 127.....	23232323	343434	45	L	50	álcool	5,00	250,00	2650,00
2610	Lopes....	R. 127.....	23232323	343434	78	Kg	47	cimento	30,00	1410,00	2650,00
2610	Lopes....	R. 127.....	23232323	343434	21	Kg	20	pregos	5,00	100,00	2650,00
2610	Lopes....	R. 127.....	23232323	343434	98	L	15	tinta azul	25,00	375,00	2650,00
2610	Lopes....	R. 127.....	23232323	343434	90	L	15	cola	3,00	45,00	2650,00
2610	Lopes....	R. 127.....	23232323	343434	43	M	10	arame	3,00	30,00	2650,00
2610	Lopes....	R. 127.....	23232323	343434	25	F	10	algodão	2,00	20,00	2650,00
2610	Lopes....	R. 127.....	23232323	343434	65	L	5	querosene	8,00	40,00	2650,00
2610	Lopes....	R. 127.....	23232323	343434	51	M	20	fio elétrico	13,00	260,00	2650,00
2610	Lopes....	R. 127.....	23232323	343434	74	M	30	linha 10	4,00	120,00	2650,00
5675	Mario.....	Av. Treze..	78787878	595968	45	L	48	álcool	5,00	240,00	1710,00
5675	Mario.....	Av. Treze..	78787878	595968	98	L	35	tinta azul	25,00	875,00	1710,00
5675	Mario.....	Av. Treze..	78787878	595968	43	M	25	arame	3,00	75,00	1710,00
5675	Mario.....	Av. Treze..	78787878	595968	51	M	40	fio elétrico	13,00	520,00	1710,00
9756	HTZ.....	Trv. Ney....	45636352	484747	90	L	60	cola	3,00	180,00	814,00
9756	HTZ.....	Trv. Ney....	45636352	484747	35	Kg	20	farinha	1,00	20,00	814,00
9756	HTZ.....	Trv. Ney....	45636352	484747	23	M	40	nylon	5,00	200,00	814,00
9756	HTZ.....	Trv. Ney....	45636352	484747	456	Kg	46	arame farpado	9,00	414,00	814,00

Figura 12.1

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

- anomalia de inclusão: ao ser incluído um novo cliente, o mesmo tem que estar relacionado a uma venda;

- anomalia de exclusão: ao ser excluído um cliente, os dados referentes as suas compras serão perdidos;
- anomalia de alteração: caso algum fabricante de produto altere a faixa de preço de uma determinada classe de produtos, será preciso percorrer toda a entidade para se realizar múltiplas alterações.

12.2 - Primeira Forma Normal (1FN)

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

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

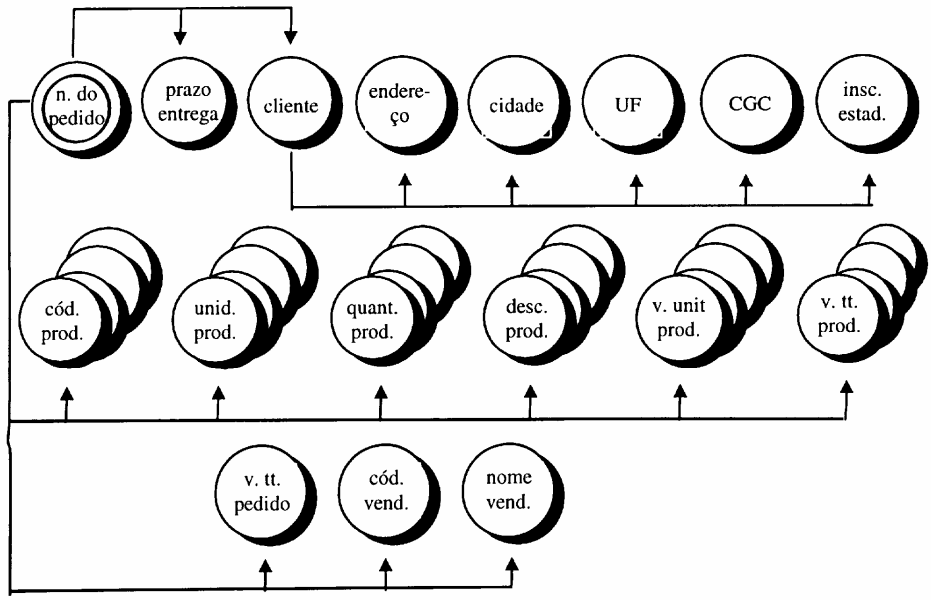
A 1FN diz que: cada ocorrência da chave primária deve corresponder a uma e somente uma informação de cada atributo, ou seja, a entidade não deve conter grupos repetitivos (multivalorados).

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

Para entidade PEDIDO, temos:

Entidade **não normalizada**:

PEDIDO:



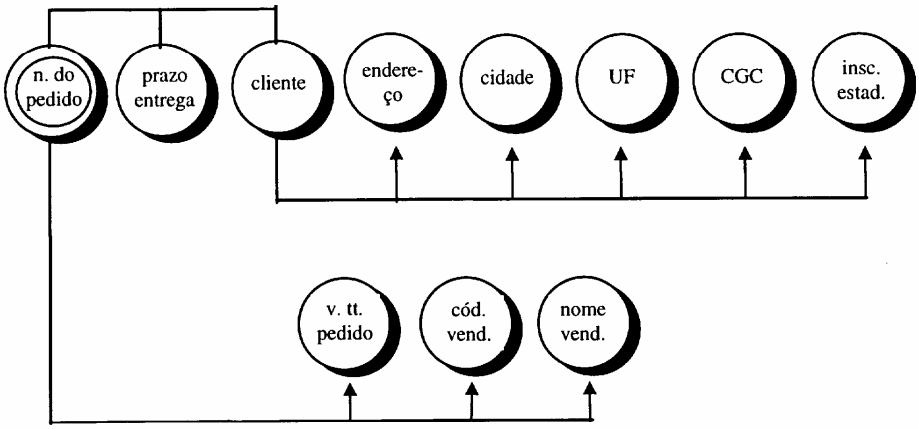
Atributo chave da Entidade:

Representação no ER: **PEDIDO**

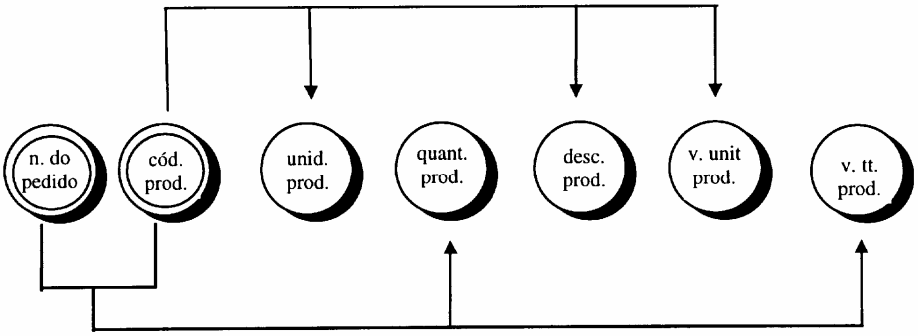
Ao aplicarmos a 1FN sobre entidade PEDIDO, obtemos mais uma entidade chamada de ITEM-DE-PEDIDO, que herdará os atributos repetitivos e destacados da entidade PEDIDO.

Entidades na 1FN:

PEDIDO



ITEM-DE-PEDIDO



Representação no ER

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

12.3 - Variação Temporal e a Necessidade de Histórico

Observamos que normalmente, ao se definir um ambiente de armazenamento de dados, seja ele um banco de dados ou não, geralmente se mantém a última informação cadastrada, que às vezes, por sua própria natureza, possui um histórico de ocorrências. Mas como a atualização é sempre feita sobre esta última informação, perdem-se totalmente os dados passados.

A não-observação deste fato leva a um problema na hora de uma auditoria de sistemas, que em vez de utilizar uma pesquisa automatizada sobre os históricos, se vê obrigada a uma caçada manual cansativa sobre um mar imenso de papéis e relatórios, e que na maioria das vezes se apresenta incompleta ou inconsistente devido a valores perdidos (documentos extraviados) ou não documentados.

Com a não-utilização de históricos e a natural perda destas informações, a tomada de decisões por parte da alta administração de uma empresa pode levar a resultados catastróficos para a corporação.

Toda vez que a decisão de armazenar o histórico de algum atributo for tomada, cria-se explicitamente um relacionamento de um para muitos (1-N), entre a entidade que contém o atributo e a entidade criada para conter o histórico deste atributo. Passa a existir então uma entidade dependente, contendo (no mínimo) toda data em que houve alguma alteração do atributo, bem como o respectivo valor do atributo para cada alteração. A chave desta entidade de histórico será concatenada, e um de seus atributos será a data de referência.

Com base nesta necessidade de armazenamento de históricos, após a aplicação da 1FN devemos observar para cada entidade definida, quais de seus atributos se transformarão com o tempo, se é preciso armazenar dados históricos deste atributo e em caso afirmativo, observar o período de tempo

que devemos conservar este histórico, ou através de quantas alterações foram realizadas neste atributo.

12.4 - Dependência Funcional

Para descrevermos as próximas formas normais, se faz necessária a introdução do conceito de dependência funcional, sobre o qual a maior parte da teoria de normalização foi baseada.

Dada uma entidade qualquer, dizemos que um atributo ou conjunto de atributos A é dependente funcional de um outro atributo B contido na mesma entidade, se a cada valor de B existir nas linhas da entidade em que aparece, um único valor de A. Em outras palavras, A depende funcionalmente de B.

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

O exame das relações existentes entre os atributos de uma entidade deve ser feito a partir do conhecimento (conceitual) que se tem sobre a realidade a ser modelada.

12.5 - Dependência Funcional Total (Completa) e Parcial

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

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

A dependência total ou completa só ocorre quando a chave primária for composta por vários (concatenados) atributos, ou seja, em uma entidade

de chave primária composta de um único atributo não ocorre este tipo de dependência.

12.6 - Dependência Funcional Transitiva

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

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

Com base na teoria sobre as dependências funcionais entre atributos de uma entidade, podemos continuar com a apresentação das outras formas normais.

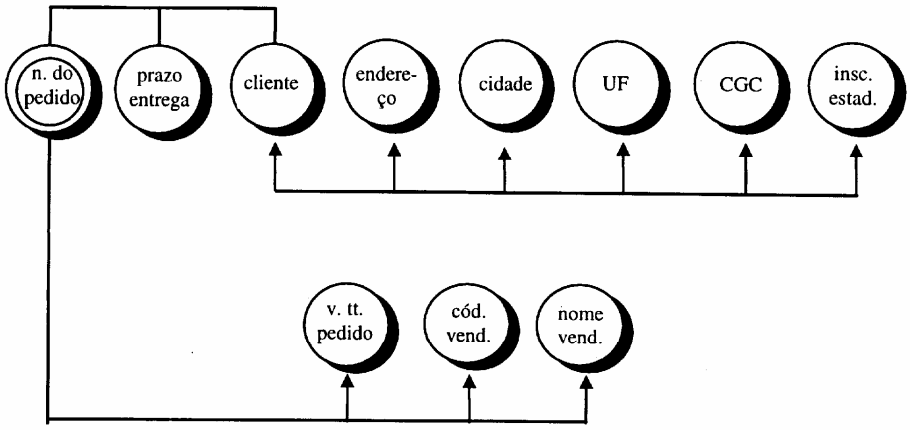
12.7 - Segunda Forma Normal (2FN)

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

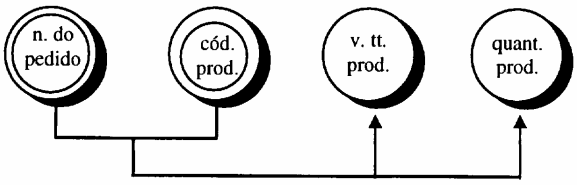
Com a finalidade de tornar ainda mais estável o modelo de dados, a aplicação da 2FN sobre as entidades em observação geram novas entidades, que herdarão a chave parcial e todos os atributos que dependem desta chave parcial, ou seja, uma entidade para estar na 2FN não pode ter atributos com dependência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

Ex: A entidade ITEM-DO-PEDIDO apresenta uma chave primária concatenada e por observação, notamos que os atributos UNIDADE-DO-PRODUTO, DESCRIÇÃO-DO-PRODUTO e VALOR-UNITARIO dependem de forma parcial do atributo CODIGO-DO-PRODUTO, que faz parte da chave primária. Logo devemos aplicar a 2FN sobre esta entidade. Quando aplicamos a 2FN sobre ITEM-DO-PEDIDO, será criada a entidade PRODUTO que herdará os atributos UNIDADE-DO-PRODUTO, DESCRIÇÃO-DO-PRODUTO e VALOR-UNITÁRIO e terá como chave primária o CODIGO-DO-PRODUTO.

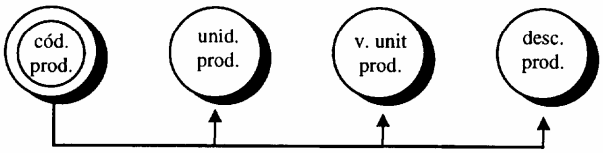
PEDIDO



ITEM-DO-PEDIDO



PRODUTO



Representação no ER:

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

12.8 - Terceira Forma Normal (3FN)

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

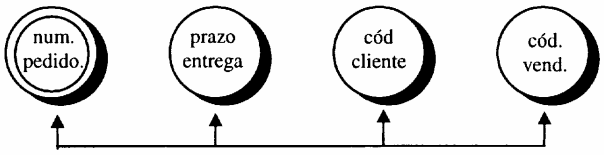
Ao retirarmos a dependência transitiva, devemos criar uma nova entidade que contenha os atributos que dependem transitivamente de outro e a sua chave primária é o atributo que causou esta dependência.

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

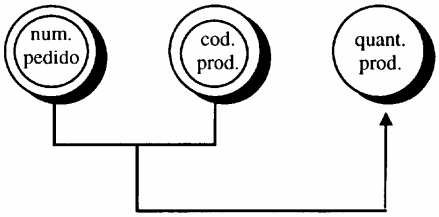
Ex: Na entidade PEDIDO, podemos observar que o atributo NOME-DO-VENDEDOR depende transitivamente do atributo CODIGO-DO-VENDEDOR que não pertence à chave primária. Para eliminarmos esta anomalia devemos criar a entidade VENDEDOR, com o atributo NOME-DO-VENDEDOR e tendo como chave primária o atributo CODIGO-DO-VENDEDOR.

Encontramos ainda o conjunto de atributos formados por ENDEREÇO, CIDADE, UF, CGC e INSCRIÇÃO-ESTADUAL que dependem transitivamente do atributo CLIENTE. Neste caso, devemos criar a entidade CLIENTE que conterá os atributos ENDEREÇO, CIDADE, UF, CGC e INSCRIÇÃO-ESTADUAL. Para chave primária desta entidade vamos criar um atributo chamado CODIGO-DO-CLIENTE que funcionará melhor como chave primária do que NOME-DO-CLIENTE, deixando este último como simples atributo da entidade CLIENTE.

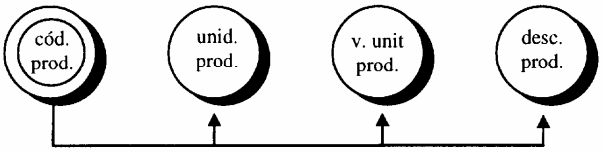
PEDIDO



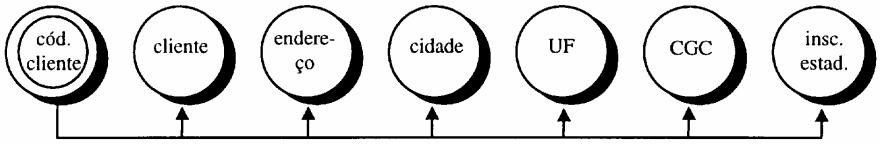
ITEM-DO-PEDIDO



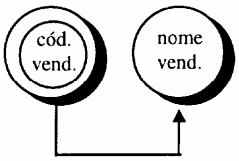
PRODUTO



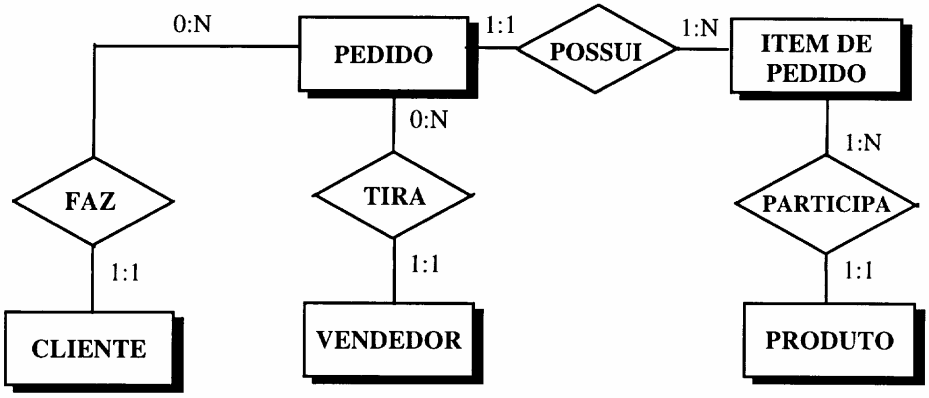
CLIENTE



VENDEDOR



Representação no ER



Um PEDIDO só é feito por um e somente um CLIENTE e um CLIENTE pode fazer de zero (clientes que devem ser contatados mais freqüentemente pelos vendedores) até N elementos de PEDIDO. Um PEDIDO só é tirado por um e somente um VENDEDOR e um VENDEDOR pode tirar de zero (vendedores que devem ser reciclados em termos de treinamento, para aumentar o poder de venda) a N elementos de PEDIDO.

12.9 - Forma Normal de BOYCE / CODD (FNBC)

As definições da 2FN e 3FN, desenvolvidas por Codd, não cobriam certos casos. Esta inadequação foi apontada por Raymond Boyce em 1974. Os casos não cobertos pelas definições de Codd somente ocorrem quando três condições aparecem juntas:

- a entidade tenha várias chaves candidatas;
- estas chaves candidatas sejam concatenadas (mais de um atributo);
- as chaves concatenadas compartilham pelo menos um atributo comum.

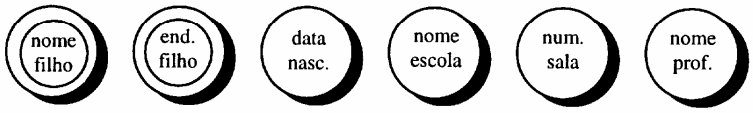
Na verdade, a FNBC é uma extensão da 3FN, que não resolvia certas anomalias presentes na informação contida em uma entidade. O problema foi observado porque a 2FN e a 3FN só tratavam dos casos de dependência parcial e transitiva de atributos fora de qualquer chave, porém quando o

atributo observado estiver contido em uma chave (primária ou candidata), ele não é captado pela verificação da 2FN e 3FN.

A definição da FNBC é a seguinte: uma entidade está na FNBC se e somente se todos os determinantes forem chaves candidatas. Notem que esta definição é em termos de chaves candidatas e não sobre chaves primárias.

Considere a seguinte entidade FILHO:

FILHO



Por hipótese, vamos assumir que um professor possa estar associado a mais de uma escola e uma sala. Sob esta suposição, tanto a chave (candidata) concatenada NOME-DA-ESCOLA + SALA-DA-ESCOLA bem como NOME-DA-ESCOLA + NOME-DO-PROFESSOR podem ser determinantes. Logo esta entidade atende às três condições relacionadas anteriormente:

- as chaves candidatas para a entidade FILHO são: NOME-DO-FILHO + ENDEREÇO-DO-FILHO, NOME-DO-FILHO + NUMERO-DA-SALA e NOME-DO-FILHO + NOME-DO-PROFESSOR;
- todas as três chaves apresentam mais de um atributo (concatenados);
- todas as três chaves compartilham um mesmo atributo: NOME-DO-FILHO.

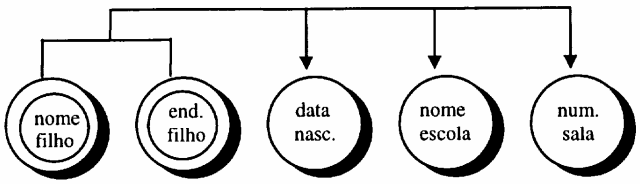
Neste exemplo, NOME-DO-PROFESSOR não é completamente dependente funcional do NUMERO-DA-SALA, nem NUMERO-DA-SALA é completamente dependente funcional do NOME-DO-PROFESSOR. Neste caso, NOME-DO-PROFESSOR é realmente completamente dependente funcional da chave candidata concatenada NOME-DO-FILHO + NUMERO-DA-SALA ou NUMERO-DA-SALA é completamente dependente funcional da chave candidata concatenada NOME-DO-FILHO + NOME-DO-PROFESSOR.

Ao se aplicar FNBC, a entidade FILHO deve ser dividida em duas entidades, uma que contém todos os atributos que descrevem o FILHO, e uma

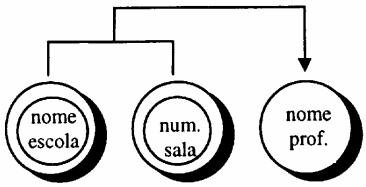
segunda que contém os atributos que designam um professor em uma particular escola e número de sala.

FNBC

FILHO



SALA



12.10 - Quarta Forma Normal (4FN)

Na grande maioria dos casos, as entidades normalizadas até a 3FN são fáceis de entender, atualizar e de se recuperar dados. Mas às vezes podem surgir problemas com relação a algum atributo não chave, que recebe valores múltiplos para um mesmo valor de chave. Esta nova dependência recebe o nome de dependência multivalorada que existe somente se a entidade contiver no mínimo três atributos.

Uma entidade que esteja na 3FN também está na 4FN, se ela não contiver mais do que um fato multivalorado a respeito da entidade descrita. Esta dependência não é o mesmo que uma associação M:N entre atributos, geralmente descrita desta forma em algumas literaturas.

Ex: Dada a entidade hipotética a seguir:

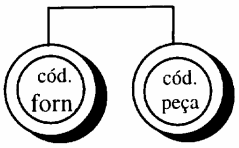
CÓDIGO FORNECEDOR	CÓDIGO PEÇA	CÓDIGO COMPRADOR
1111	BA3	113
1111	CJ10	113
1111	88A	435
1111	BA3	537

Tabela 1

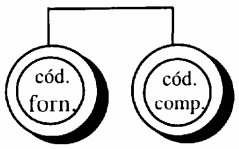
Como podemos observar, esta entidade tenta conter dois fatos multivalorados: as diversas peças compradas e os diversos compradores. Com isso apresenta uma dependência multivalorada entre CÓDIGO-FORNECEDOR e o CÓDIGO-PEÇA e entre CÓDIGO-FORNECEDOR e o CÓDIGO-COMPRADOR. Embora esteja na 3FN, ao conter mais de um fato multivalor, torna sua atualização muito difícil, bem como a possibilidade de problemas relativos ao espaço físico de armazenamento poderem ocorrer, causados pela ocupação desnecessária de área de memória (primária ou secundária), podendo acarretar situações críticas em termos de necessidade de mais espaço para outras aplicações.

Para passarmos a entidade acima para a 4FN, é necessária a realização de uma divisão da entidade original, em duas outras, ambas herdando a chave CÓDIGO-FORNECEDOR e concatenado, em cada nova entidade, com os atributos CÓDIGO-PEÇA e CÓDIGO-COMPRADOR.

FORNECEDOR-PEÇA



FORNECEDOR-COMPRADOR



12.11 - Quinta Forma Normal (5FN)

Esta última forma normal trata do conceito de dependência de junção, quando a noção de normalização é aplicada à decomposição, devido a uma operação de projeção, e aplicada na reconstrução devido a uma junção.

A 5FN trata de casos bastante particulares, que ocorrem na modelagem de dados, que são os relacionamentos múltiplos (ternários, quaternários,...., n-ários). Ela fala que um registro está na sua 5FN, quando o conteúdo deste mesmo registro não puder ser reconstruído (junção) a partir de outros registros menores, extraídos deste registro principal. Ou seja, se ao particionar um registro, e sua junção posterior não conseguir recuperar as informações contidas no registro original, então este registro está na 5FN.

Vamos ilustrar o uso da 5FN utilizando um exemplo de relacionamento ternário, apresentado no livro do Prof. Waldemar Setzer [18].

Ex: Uma empresa constrói equipamentos complexos. A partir de desenhos de projeto desses equipamentos, são feitos documentos de requisições de materiais, necessários para a construção do equipamento; toda a requisição de um material dá origem a um ou mais pedidos de compra. A modelagem deste exemplo, irá mostrar quais materiais de que requisições geraram quais pedidos. Na figura 12.2 é apresentado este relacionamento ternário.

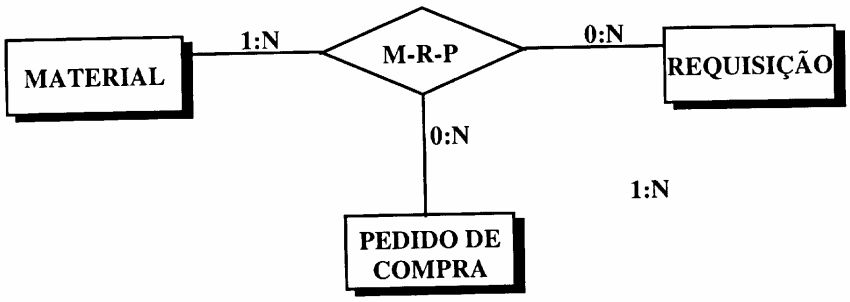


Figura 12.2

A tabela 2, representante do relacionamento ternário M-R-P, poderia conter os seguintes dados:

MATERIAL	PEDIDO DE COMPRA	REQUISIÇÃO
ROTOR 1BW	PC 0792	R1292
ROTOR 1BW	PC0992	R3192
CI 102	PC0792	R3192
ROTOR 1BW	PC0792	R3192

Tabela 2

Utilizando uma soma de visualização da dependência de junção, apresentada por James Bradley[4], obtemos o seguinte gráfico de dependência de junção, mostrado na figura 12.3.

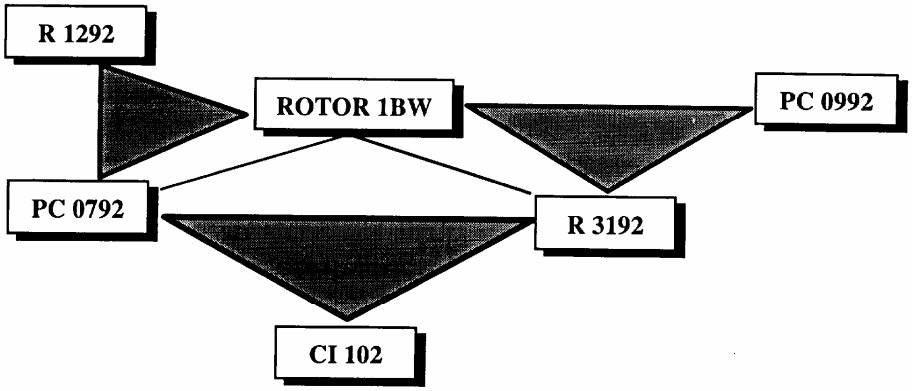


Figura 12.3

Uma pergunta surge sobre este problema: é possível substituir o relacionamento ternário por relacionamentos binários, como os apresentados na figura 12.4.

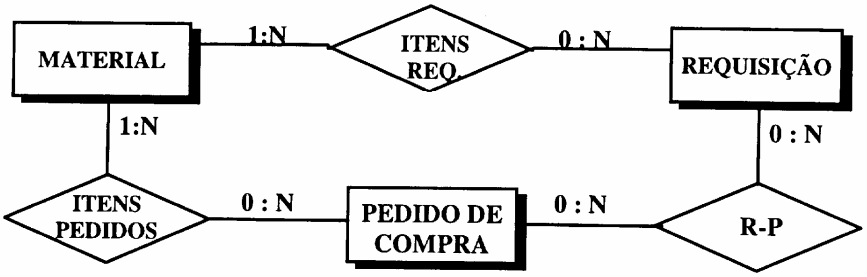


Figura 12.4

Como resposta, podemos dizer que geralmente não é possível criar esta decomposição sem perda de informação, armazenada no relacionamento ternário.

Realizando uma projeção na tabela anterior, chegamos às entidades apresentadas na figura 12.5.

ENTIDADE 1

MATERIAL	PEDIDO DE COMPRA
ROTOR 1BW	PC 0792
ROTOR 1BW	PC0992
CI 102	PC0792

ENTIDADE 2

PEDIDO DE COMPRA	REQUISIÇÃO
PC 0792	R1292
PC0992	R3192
PC0792	R3192

ENTIDADE 3

MATERIAL	REQUISIÇÃO
ROTOR 1BW	R1292
ROTOR 1BW	R3192
CI 102	R3192

Figura 12.5

Se realizarmos, agora, um processo de junção destas três entidades, teremos:

- Inicialmente, vamos juntar a entidade 1 com a entidade 2, através do campo pedido de compra. Obtemos então a entidade 4, mostrada na figura 12.6;

ENTIDADE 4

MATERIAL	PEDIDO DE COMPRA	REQUISIÇÃO
ROTOR 1BW	PC 0792	R1292
ROTOR 1BW	PC0992	R3192
CI 102	PC0792	R3192
ROTOR 1BW	PC0792	R3192
CI 102	PC0792	R1292

Figura 12.6

- Podemos observar que o registro apontado pela seta não existia na tabela original, ou seja, foi criado pela junção das tabelas parciais. Devemos juntar a entidade 4, resultante da primeira junção, com a entidade 3, através dos campos material e requisição. Após esta última operação de junção, obtemos a entidade 5, mostrada na figura 12.7.

ENTIDADE 5

MATERIAL	PEDIDO DE COMPRA	REQUISIÇÃO
ROTOR 1BW	PC 0792	R1292
ROTOR 1BW	PC0992	R3192
CI 102	PC0792	R3192
ROTOR 1BW	PC0792	R3192

Figura 12.7

Como se pode notar, ao se juntar as três entidades, fruto da decomposição da entidade original, as informações destas foram preservadas. Isto significa que o relacionamento M-R-P não está na 5FN, sendo necessário decompô-lo em relacionamentos binários, os quais estarão na 5FN.

A definição da 5FN diz que: uma relação de 4FN estará em 5FN, quando seu conteúdo não puder ser reconstruído (existir perda de informação) a partir das diversas relações menores que não possuam a mesma chave primária. Esta forma normal trata especificamente dos casos de perda de informação, quando da decomposição de relacionamentos múltiplos.

Com a 5FN, algumas redundâncias podem ser retiradas, como a informação de que o "ROTOR 1BW" está presente na requisição "R3192", será armazenada uma única vez, a qual na forma não normalizada pode ser repetida inúmeras vezes.

12.12 - Roteiro de Aplicação da Normalização

- Entidade ou documento não normalizado, apresentando grupos repetitivos e certas anomalias de atualização.

- aplicação da 1FN
  - Decompor a entidade em uma ou mais entidades, sem grupos repetitivos;
  - Destacar um ou mais atributos como chave primária da(s) nova(s) entidade(s), e este será concatenado com a chave primária da entidade original.;
  - Estabelecer o relacionamento e a cardinalidade entre a(s) nova(s) entidade(s) gerada(s) e a entidade geradora;
  - Verificar a questão da variação temporal de certos atributos e criar relacionamentos 1:N entre a entidade original e a entidade criada por questões de histórico.

- ENTIDADES NA 1FN

- aplicação da 2FN
  - Para entidades que contenham chaves primárias concatenadas, destacar os atributos que tenham dependência parcial em relação à chave primária concatenada;

- Criar uma nova entidade que conterá estes atributos, e que terá como chave primária o(s) atributo(s) do(s) qual(quais) se tenha dependência parcial;
- Serão criadas tantas entidades quanto forem os atributos da chave primária concatenada, que gerem dependência parcial;
- Estabelecer o relacionamento e a cardinalidade entre a(s) nova(s) entidade(s) gerada(s) e a entidade geradora.

➲ ENTIDADES NA 2FN

♦ aplicação da 3FN

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

➲ ENTIDADES NA 3FN

♦ aplicação da FNBC

- Só aplicável em entidades que possuam chaves primárias e candidatas concatenadas;
- Verificar se alguma chave candidata concatenada é um determinante, e em caso afirmativo, criar uma entidade com os que dependam funcionalmente deste determinante e cuja chave primária é o próprio determinante.

➲ ENTIDADES NA FNBC

♦ aplicação da 4FN

- Para se normalizar em 4FN, a entidade tem que estar (obrigatoriamente) na 3FN;
- Verificar se a entidade possui atributos que não sejam participantes da chave primária e que sejam multivalorados e independentes em relação a um mesmo valor da chave primária;

- Retirar estes atributos não chaves e multivalorados, criando novas entidades individuais para cada um deles, herdando a chave primária da entidade desmembrada.

➲ ENTIDADES NA 4FN

♦ aplicação da 5FN
- Aplicada em elementos que estejam na 4FN;
- A ocorrência deste tipo de forma normal está vinculada aos relacionamentos múltiplos (ternários, etc.) ou entidades que possuam chave primária concatenada com três ou mais atributos;
- Verificar se é possível reconstruir o conteúdo do elemento original a partir de elementos decompostos desta;
- Se não for possível, o elemento observado não está na 5FN, caso contrário os elementos decompostos representam um elemento na 5FN.

➲ ENTIDADES NA FORMA NORMAL FINAL

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

12.13 - Desnormalização

Parece um tanto quanto incoerente, em um capítulo sobre normalização, apresentar um item falando sobre desnormalização. Porém os processos de síntese de entidades vistos até aqui levam à criação de novas entidades e relacionamentos.

Os novos elementos criados podem trazer prejuízos na hora de serem implementados em um SGDB. Devido as características de construção física de certos bancos de dados, algumas entidades e relacionamentos devem ser desnormalizados para que o SGBD tenha um melhor desempenho.

Hoje em dia, existe um grande debate sobre as chamadas semânticas da normalização, sua utilidade e falicilidade em relação à implementação física em um sistema computacional.

Vários estudos e algumas considerações estão sendo realizados, e se chega até a utilização de banco de dados relacionais não normalizados [18], por apresentarem maior ligação com a realidade e por terem vínculos matemáticos mais amenizados.

Outro aspecto da normalização é que todas as definições sobre as formas normais após a 1FN, ainda não foram exaustivamente examinadas, propiciando assim grandes controvérsias. A redução das anomalias de atualização devido às formas normais de alta ordem sofre os ataques óbvios dos grandes problemas (físicos) de atualização, pois as relações estão excessivamente normalizadas e com isso uma simples alteração pode encadear um efeito cascata bastante profundo no banco de dados, ocasionando um aumento bastante significativo no tempo.

Infelizmente, os argumentos que podem viabilizar o processo de desnormalização sofrem de uma deficiência e aderência matemática.

Pesquisas recentes indicam que as estruturas desnormalizadas (apelidadas de forma normal não primeira) têm um atrativo matemático similar ao que foi o da normalização. Estas pesquisas estão provendo recentes definições sobre álgebra relacional desnormalizada e extensões à linguagem SQL para a manipulação de relações desnormalizadas.

Ao se optar pela desnormalização, deve-se levar em conta o custo da redundância de dados e as anomalias de atualização decorrentes.

Chegou-se à conclusão também que o espírito da normalização contradiz vários princípios importantes, relativos à modelagem semântica e à construção de bases de dados em SGBD orientados por objeto. Este assunto, como já foi dito anteriormente, fica para um próximo encontro entre nós e vocês, caros leitores.

12.14 - Considerações Finais Sobre Normalização

Antes de qualquer conclusão, podemos observar que as formas normais nada mais são do que restrições de integridade, e à medida que se alimenta este grau de normalização, tornam-se cada vez mais restritivas.

Dependendo do SGBD relacional utilizado, essas restrições podem se tornar benéficas ou não.

A forma de atuação da normalização no ciclo de vida de um projeto de bases de dados pode ser mais satisfatória no desenvolvimento (*botton - up*) de modelos preliminares, a partir da normalização da documentação existente no ambiente analisado, bem como de arquivos utilizados em alguns processos automatizados neste ambiente.

No caso do desenvolvimento *top - dowm*, no qual um modelo de dados é criado a partir da visualização da realidade, a normalização serve para realizar um aprimoramento deste modelo, tornando-o menos redundante e inconsistente. No caso desta visão, a normalização torna-se um poderoso aliado da implementação física do modelo.

Por experiência, podemos afirmar que a construção de um modelo de dados já leva naturalmente ao desenvolvimento de entidades e relacionamentos na 3FN, ficando as demais (FNBC, 4FN e 5FN) para melhorias e otimizações.

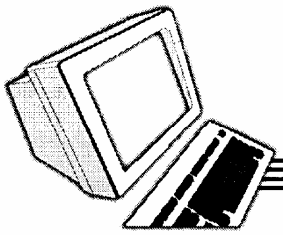
A criação de modelos de dados, partindo-se da normalização de documentos e arquivos pura e simplesmente, não é o mais indicado, pois na verdade estaremos observando o problema e não dando uma solução para ele. Neste caso, estaremos projetando estruturas de dados que se baseiam na situação atual (muitas vezes caótica) e que certamente não vão atender às necessidades reais do ambiente em análise. Ao passo que, se partirmos para a criação do modelo de dados com entidades e relacionamentos aderentes à realidade em estudo (mundo real), vamos naturalmente desenvolver uma base de dados ligada à visão da realidade e como conseqüência iremos solucionar o problema de informação.

A aplicação da modelagem de dados, ao longo da nossa vida profissional, tem sido bastante gratificante, mostrando principalmente, que a técnica de normalização é uma ferramenta muito útil como apoio ao desenvolvimento do modelo de dados. Seja ela aplicada como levantamento inicial (documentos e arquivos) bem como otimizadora do modelo de dados, tendo em vista certas restrições quanto à implementação física nos bancos de dados conhecidos.

Todas as idéias sobre eficiência da normalização passam necessariamente sobre tempo e espaço físico, em função, principalmente, das consultas efetuadas pelos usuários bem como a quantidade de bytes necessários para guardar as informações.

Nota-se, através da observação, que o projeto do modelo conceitual nem sempre pode ser derivado para o modelo físico final. Com isso, é de grande importância que o responsável pela modelagem (analista, AD, etc.) não conheça só a teoria iniciada por Peter Chen, mas também tenha bons conhecimentos a respeito do ambiente de banco de dados utilizado pelo local em análise.

Capítulo



O Modelo Lógico Relacional

13

Criado por Edgar F. Codd, nos anos 70 [6], começou a ser realmente utilizado nas empresas a partir de 1987. A abordagem relacional está baseada no princípio de que as informações em uma base de dados podem ser consideradas como relações matemáticas e que estão representadas de maneira uniforme, através do uso de tabelas bidimensionais. Este princípio coloca os dados (entidades e relacionamentos) dirigidos para estruturas mais simples de armazenar dados, que são as tabelas, e nas quais a visão do usuário é privilegiada.

Definição Clássica: São conjuntos de dados vistos segundo um conjunto de TABELAS (figura 13.1) e as operações sobre elas (tabelas) são feitas por linguagens que manipulam a álgebra relacional, não sendo procedurais, ou seja, manipulando conjuntos de uma só vez.

Produto

CDProd	Descrição	Quant-Est

Compra

cdP	CDCO	Quant.	Preço

Comprador

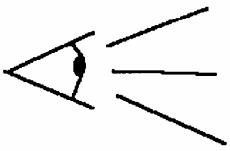
CDCO	Endereço	Histórico

Exemplos: DB2, INGRES, ORACLE, PROGRESS, XDB, etc.....

Figura 13.1

O conceito principal vem da teoria dos conjuntos (álgebra relacional) atrelado à idéia de que não é relevante ao usuário saber **onde os dados estão nem como os dados estão** (transparência). Os usuários manipulam objetos dispostos em linhas e colunas das tabelas (figura 13.2) . O usuário, para lidar com estes objetos, conta com um conjunto de operadores e funções de alto nível, constantes na álgebra relacional.

VISÃO LÓGICA DE DADOS



SAFRA	TIPO	PRODT.	MARCA
86	BRANCO	01	CLOS DE NOBLES
87	TINTO	01	CLOS DE NOBLES
86	BRANCO	05	FORESTIER RIESL.
85	TINTO	03	GRANJA UNIÃO
87	BRANCO	07	ALMADEN CORD.

COMO O USUÁRIO VÊ SEUS DADOS

Figura 13.2

13.1 - Principais Vantagens da Abordagem Relacional

- Independência total dos dados;
- Visão múltipla dos dados;
- Melhor comunicação entre CPD e usuário;
- Redução acentuada na atividade de desenvolvimento de aplicações e o tempo gasto em manutenção;

- Melhoria na segurança dos dados;
- Mais agilidade na questão gerencial da informação ligada ao processo decisório da organização.

13.2 - As 12 Regras de Codd

Codd, ao definir o modelo relacional, estabeleceu um conjunto de 12 regras para a determinação de um banco de dados ser realmente relacional. Segundo estas regras, discute-se a fidelidade de um SGBD ao modelo relacional [7]. Raros são os bancos de dados que se enquadrem em mais do que 10 destas regras.

As regras de Codd são:

1 - Toda informação num banco de dados relacional é apresentada a nível lógico por valores em tabelas;

2 - Todo dado em um banco de dados relacional tem a garantia de ser logicamente acessível, recorrendo-se a uma combinação do nome da tabela, um valor de chave e o nome da coluna;

3 - Tratamento sistemático de valores nulos (ausência de dado);

4 - O dicionário de dados (catálogo) relacional ativo é baseado no modelo relacional;

5 - O SGBD relacional deve ter uma linguagem para definição, detalhamento e manipulação dos dados;

6 - Tratamento das atualizações de visões dos dados;

7 - Tratamento de alto nível para inserção, atualização e eliminação de dados;

8 - Independência dos dados físicos (mudança na memória e no método de acesso);

9 - Independência de dados lógicos (mudanças de qualquer tipo nas tabelas básicas, ex: divisão de uma tabela por linha ou coluna);

10 - Independência das restrições de integridade;

11 - Independência de distribuição;

12 - Não subversão das regras de integridade ou restrições quando se utiliza uma linguagem de baixo nível.

13.3 - Chaves e Índices

- **Chave** designa o conceito de item de busca, ou seja, um dado que será empregado nas consultas à base de dados. É um conceito lógico da aplicação;
- **Índice** é um recurso físico visando otimizar a recuperação de uma informação, via um método de acesso. Seu objetivo principal está relacionado com a performance de um sistema.

Uma chave pode ser utilizada como índice, mas um índice não é necessariamente uma chave. A forma de criação do índice depende do ambiente relacional.

Ex. lista invertida (ADABAS®), árvore B+ (DB2®)

13.4 - O Conceito de Chave no Modelo Relacional

➢ Chave PRIMÁRIA (primary key): É o atributo de uma tabela que identifica univocamente uma tupla.

Uma chave primária não tem nenhuma ligação com o conceito de ordenação e com o acesso à tabela, pois declarar um atributo como primário e acessar a tabela por outro atributo, serve para manter duas restrições de integridade determinadas por Codd.

➢ Chave SECUNDÁRIA (secundary key): Serve para definir uma segunda chave primária. Identifica sempre um item de busca, através do qual desejamos recuperar uma informação ou um conjunto de informações. É o atributo de uma tabela que identifica um subconjunto de tuplas onde este subconjunto pode ter apenas uma tupla. No ambiente tradicional, só posso acessar um registro se declarar que aquele campo é chave. No ambiente relacional, uma tabela é acessível por qualquer campo (atributo) independente deste ser declarado como chave ou não.

Usamos normalmente a declaração de chave secundária para agilizar o acesso àquela tabela naquele campo (atributo).

Para cada chave secundária declarada, o ambiente relacional cria um índice (estratégia de acesso). Qualquer campo pode ser chave secundária.

- Chave CANDIDATA: Uma tabela relacional pode possuir alternativas de identificador único, ou seja, várias colunas ou concatenações diferentes de colunas podem ter esta propriedade. Estes identificadores são candidatos à chave primária, como um e somente um será escolhido como chave primária, o restante passa a ser considerado como chave alternativa.
- Chave ESTRANGEIRA (foreign key): As chaves estrangeiras constituem um conceito de vital importância no modelo relacional. São os elos de ligação entre as tabelas.

Quando dizemos que duas tabelas estão relacionadas através de atributos comuns, devemos observar que provavelmente esta coluna em uma das tabelas é uma chave primária. Na outra tabela, este atributo irá caracterizar o que é denominado de chave estrangeira, propiciando, assim, uma ligação lógica (relacionamento) entre as tabelas.

O CONCEITO DE CHAVE NO MODELO RELACIONAL

Cod. Projeto	Nome do Projeto	Tipo do Projeto	Departamento
SFT	Sist. Faturamento	001-I	ANÁLISE
CAP	Contas a Pagar	002-N	PROGRAMAÇÃO
ADP	Modulo de Cadastro	004-E	ANÁLISE
COB-2	Cobranca Filial RIO	002-N	DIRETORIA SISTEMAS
PAG	Folha de Pagamento	007-E	PROGRAMAÇÃO

Chave Primária

Chave Estrangeira

Chave Estrangeira

Chave Primária

Chave Primária

Verba	Nome do Departamento
16.000	ANÁLISE
22.000	PROGRAMAÇÃO
6.000	DIRETORIA SISTEMAS
2.500	OPERAÇÃO
1.000	O & M

Tipo	Descrição
001-I	PROTOTIPO
002-N	NOVO SISTEMA
005-I	ESTUDO
004-E	EXPANSÃO
007-E	ADEQUAÇÃO

13.5 - Regras de Integridade do Modelo Relacional

1- Integridade de Identidade (ou entidade)

A chave primária não pode conter um valor nulo (NULL). O NULL não é o valor 0 (zero) nem o caractere branco, é simplesmente a não-existência de conteúdo neste campo.

2- Integridade Referencial

Se uma determinada tabela A possui uma chave estrangeira, a qual é chave primária em uma outra tabela B, então ela deve:

- ser igual a um valor de chave primária existente em B; ou
- ser NULA (NULL).

Não pode existir na chave estrangeira, um valor que não exista na tabela na qual ela é chave primária.

As regras de integridade do modelo relacional representam a garantia de que as tabelas guardam informações compatíveis. São de extrema importância para a confiabilidade das informações contidas no banco de dados.

13.6 - Características do Modelo Relacional

1 - Uma tabela é acessível por qualquer campo (atributo) independente se este é declarado como chave ou não;

2 - O relacionamento entre conjunto de dados (tabelas) não existe fisicamente, pois este relacionamento é apenas lógico e representado através das chaves estrangeiras;

3 - Utilização de linguagens auto-contidas e não procedurais;

4 - Os ambientes relacionais possuem um otimizador estratégico para escolher o melhor caminho para recuperação dos dados.

13.7 - Derivação do Modelo E-R para o Modelo Relacional

Nesta etapa, o desenvolvedor irá passar as visões do modelo conceitual para o modelo lógico relacional, no qual os dados são vistos como estruturas de dados voltadas para as características do modelo lógico escolhido, visando a implementação do banco de dados.

Regras de Conversão

Existem regras precisas que não dão margem a erros, sendo que uma vez projetado o ER, as tabelas que representam o ER num nível mais baixo podem ser obtidas de forma clara. É evidente que nesta passagem devem ser observadas as características do SGBD que será utilizado, pois existem diferenças que podem auxiliar ou dificultar algumas soluções propostas. Alguns SGBD's necessitam de adaptações específicas.

➢ **Mapeamento das ENTIDADES**: Toda entidade torna-se uma tabela carregando todos os atributos (definidos para a entidade). Cada atributo vira um campo da tabela criada. Cada uma das chaves gera "estruturas de acesso". A chave primária e as chaves candidatas são projetadas para não permitirem ocorrências múltiplas e nem admitem nulos.

As chaves das entidades fracas são formadas pelos atributos indicados, precedidos pelos atributos que compõem a chave primária das entidades da qual ela é fraca.

Você vê um retângulo, o que fazer? Transformá-lo em uma tabela, não tem como errar.

Por que não tem como errar? Porque estamos falando de um modelo conceitual previamente elaborado e validado junto com o usuário e os processos que vão trabalhar em cima desses dados.

Ex. Processo: dado um banco de dados **empregado** e **departamento.** O usuário olha para o banco e diz: Quero um relatório que **imprima o número de empregados que cada departamento tem**, aí você valida: esses dados respondem ao relatório que o usuário quer? **sim** - está legal; **não** - tenho de ajustar o banco de dados pois não vai responder à necessidade do usuário. A validação é sempre com o usuário, ele confirma e vê se o meu banco responde.

➢ **Mapeamento dos ATRIBUTOS**: Os atributos das entidades e relacionamentos (que possuam atributos) devem ser gerados na ordem que minimize o consumo de espaço de armazenamento e torne mais eficiente a recuperação. Devem ser exploradas todas as características de SGBD em uso. Para tanto, deve ser considerado se os campos têm ou não a especificação de extensão em bytes, se existe localização no interior do registro que propicie vantagens na recuperação e se existe compactação de espaços não ocupados.

➢ **Mapeamento dos RELACIONAMENTOS**

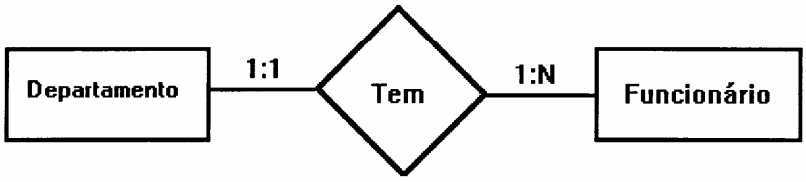
As alternativas possíveis são divididas em dois grandes grupos:

- **Navegação incorporada**: trabalha diretamente com o conceito de chave estrangeira;
- **Navegação disjunta**: trabalha sem a modificação das definições dos registros já existentes, criando novos registros (entidades) diferentes dos existentes, que têm a finalidade de propiciar a navegação.

Neste livro, iremos utilizar as alternativas do primeiro grupo (navegação incorporada), por ser mais simples e de uso mais comum hoje em dia. A segunda alternativa será utilizada na questão dos relacionamentos N:M.

➢ **Relacionamento - 1:N** (envolvendo entidades distintas)

A entidade (tabela) cuja conectividade é N carrega o identificador da entidade (tabela) cuja conectividade é 1 (chave estrangeira), e os atributos do relacionamento, se houver (é praticamente impossível aparecer em relacionamentos 1:N e 1:1 algum atributo, pois neste nível já se consegue enxergar quem é o dono do dado). Ou seja, quem está com o 'N' do lado carrega a chave do outro.



Departamento

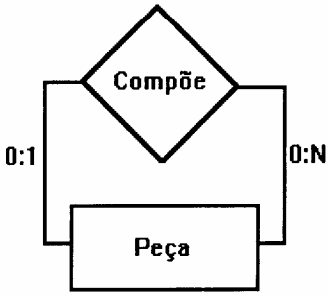
Cod.	Nome do Departamento
SFT	ANÁLISE
COB-2	PROGRAMAÇÃO
ADP	DIRETORIA SISTEMAS

Funcionário

Mat.	Nome	Etc.	CDEP
4534	Soraia Mattos	...	SFT
6547	Breno Medeiros	...	ADP
7734	Gustavo Borges	...	COB-2
1198	Ana Ferreira	...	SFT
3289	Telma Ribeiro	...	ADP

➢ **Relacionamento - 1:N** (envolvendo auto-relacionamento)

Incluir a chave primária da entidade na própria entidade como chave estrangeira, gerando uma estrutura de acesso a partir desta chave estrangeira.



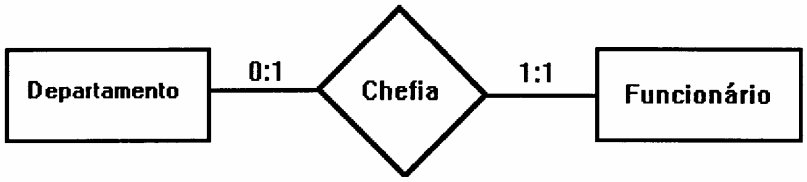
Peça

Cód.	Nome	Etc.	Códch
4534	Correia	...	
6547	Parafuso	...	
7734	Freio	...	6547
1198	Carburador	...	6547

Compõe

➢ **Relacionamento - 1:1** (envolvendo entidades distintas)

As entidades (tabelas) envolvidas neste relacionamento carregarão o identificador da outra (uma ou outra ou ambas) conforme a conveniência do projeto (de acordo com o acesso a essas tabelas).



Departamento

Cod.	Nome do Departamento
SFT	ANÁLISE
COB-2	PROGRAMAÇÃO
ADP	DIRETORIA SISTEMAS

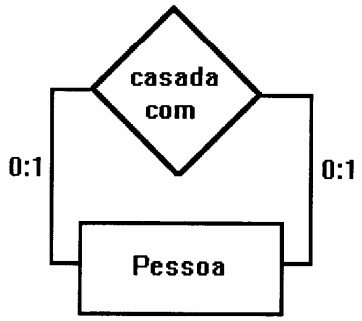
Funcionário

Mat.	Nome	Etc.	CDEP
4534	Soraia Mattos	...	SFT
6547	Breno Medeiros	...	
7734	Gustavo Borges	...	COB-2
1198	Ana Ferreira	...	
3289	Telma Ribeiro	...	ADP

Segundo a nossa definição, carregar o identificador da outra (uma ou outra ou ambas). Se no acesso, as operações que manipulam esse banco de dados são operações que manipulam muito o conjunto de DEPARTAMENTO, então seria conveniente colocar a matrícula do funcionário no conjunto DEPARTAMENTO e, se for FUNCIONÁRIO o mais operado colocamos o Código do departamento em Funcionário, se manipula os dois coloca-se nos dois.

➢ **Relacionamento - 1:1** (envolvendo auto-relacionamento)

Incluir a chave primária da entidade na própria entidade (chave estrangeira) e gerar uma estrutura de acesso para ela.



Papel de Marido

Cód.	Nome	Códcj
P2	Mauricio	
P3	Celso	
P1	Susana	P2
P4	Edna	P3
P5	Carlos	

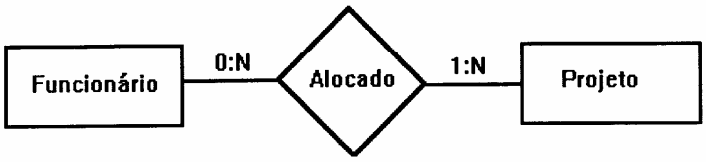
Papel de Esposa

Cód.	Nome	Códcj
P2	Mauricio	P1
P3	Celso	P4
P1	Susana	
P4	Edna	
P5	Carlos	

➢ **Relacionamento - M:N** (envolvendo entidades distintas e auto-relacionamento)

O relacionamento torna-se uma tabela carregando os atributos (se houver) e os identificadores das tabelas (entidades) que ele relaciona.

Esse é o único caso em que um relacionamento torna-se uma tabela. (figuras na próxima página)



Funcionário

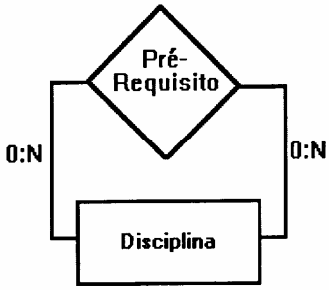
Mat.	Nome	Etc.
4534	Soraia Mattos	...
6547	Breno Medeiros	...
7734	Gustavo Borges	...
1198	Ana Ferreira	...
3289	Telma Ribeiro	...

Alocado

MAT	CODP	DTALOC
4534	PRIB	...
1198	PRIB	...
4534	PLV	05/05/94
6547	PRIB	...
7734	PPIR	17/02/92
3289	PPIR	...
7734	PRIB	04/05/94

Projeto

Cod.	Nome do Projeto
PRIB	Ribeira
PLV	Linha Vermelha
PPIR	Piratininga
PBG	Guanabara



Disciplina

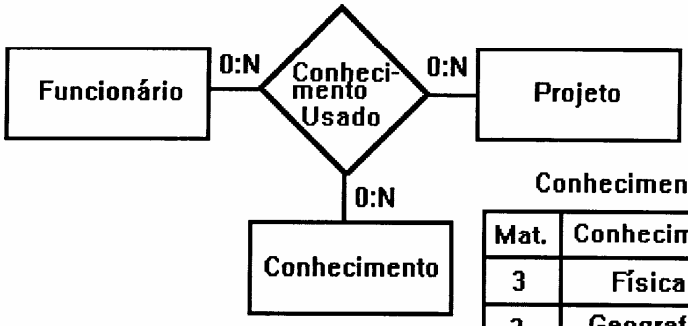
Cód. Disciplina	Nome Disciplina
666	Química I
123	Cálculo III
324	Física I
888	Cálculo II
491	Física II

Pré-Requisito

Cód.	CPré
123	888
491	324
491	888
666	324

➢ Relacionamento múltiplo

O relacionamento é mapeado em uma tabela e são geradas tantas estruturas de acesso quanto for o grau do relacionamento. A chave primária de cada uma das entidades associadas gera uma estrutura de acesso. A chave desta nova tabela é a concatenação das chaves estrangeiras.



Conhecimento Usado

Mat.	Conhecimento	CProj.
3	Física	P1
3	Geografia	P28
3	Cálculo	P4
4	Física	P28
4	Desenho	P9
8	Inglês	P4
8	Física	P9
9	Geografia	P28

➢ **As generalizações**

Ex: dado o conjunto 'funcionário', existe uma variação para este, tem um ponto básico para os funcionários que são engenheiros pois os mesmos têm informações adicionais, estes dados adicionais no subconjunto 'engenheiro', para o vendedor existem outros dados que não seriam semelhantes aos de engenheiro.

O artifício, pois não posso ter campos com tamanho variável, é criar subconjuntos para os casos em que as informações variam.

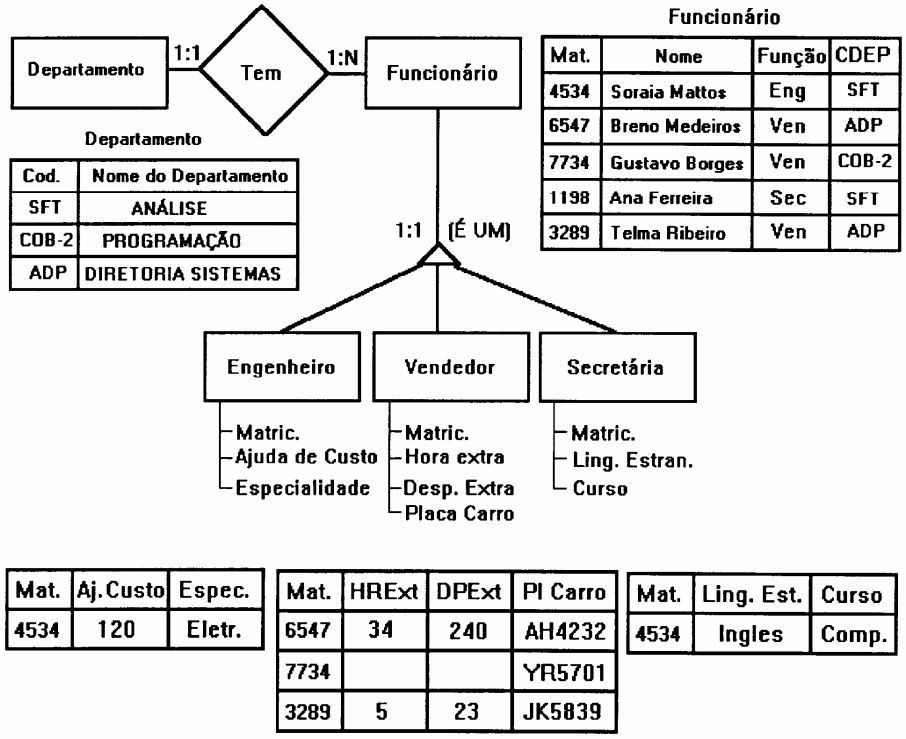
Um elemento de 'funcionário' só pode ter em um e somente um subconjunto.

As informações dos engenheiros serão completadas pelo subconjunto 'engenheiro'.

Se não tivesse atributo diferente para os outros, todos os que não fossem engenheiros ou vendedores só teriam seus dados no conjunto 'funcionário'.

Não é possível, um empregado cuja função é engenheiro, atuar como vendedor, porque os elementos não podem se sobrepor.

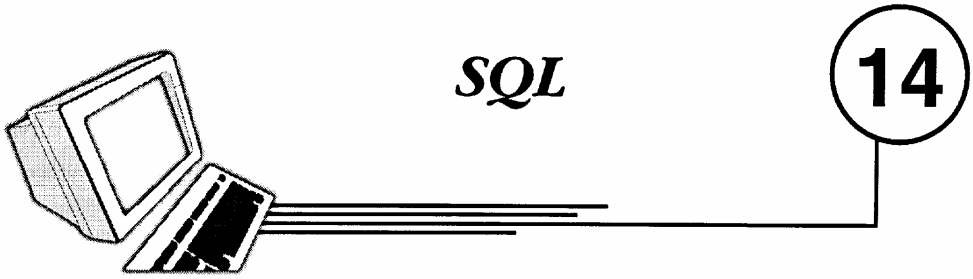
Os subconjuntos tornam-se tabelas carregando o identificador do conjunto ao qual pertencem.



A garantia de não-sobreposição dos subconjuntos é uma restrição de integridade que deve ser expressa na linguagem de acesso ao banco de dados (ex: SQL).

O conjunto 'funcionário' vira uma tabela (regra padrão) e os subconjuntos serão transformados em outras tabelas, carregando a chave primária matrícula.

Capítulo



14. 1 - A Importância da Linguagem SQL

O nome "SQL" significa "Structured Query Language" - Linguagem Estruturada de Pesquisa. Essa linguagem, de grande utilização, teve seus fundamentos no modelo relacional de Codd (1970). Sua primeira versão recebeu o nome de SEQUEL ("Structured English Query Language"), sendo definida por D. D. CHAMBERLIN, entre outros, em 1974, nos laboratórios de pesquisa da IBM (Califórnia). Em 1975, foi implementado um protótipo de aplicação dessa nova linguagem. Entre 1976 e 1977, o SEQUEL foi revisado e ampliado, e teve seu nome alterado para "SQL" por razões jurídicas.

Com esta revisão foi posto em prática um projeto ambicioso da IBM chamado System R. Novas alterações foram introduzidas na SQL, graças às idéias apresentadas pelos diversos usuários do ambiente.

Devido ao sucesso dessa nova forma de consulta e manipulação de dados, dentro de um ambiente de banco de dados, a utilização da SQL foi se tornando cada vez maior. Com isso uma grande quantidade de SGBD's foi tendo como linguagem básica a SQL - SQL/DS e DB2 da IBM, ORACLE da Oracle Corporation, RDB da Digital, SYBASE da Sybase INC, e Microsoft® SQL Server™, entre outros.

A SQL se tornou um padrão de fato, no mundo dos ambientes de banco de dados relacionados. Bastava agora se tornar de direito. Então, em 1982, o American National Standard Institute (ANSI) tornou a SQL padrão oficial de linguagem em ambiente relacional.

Infelizmente, como todo padrão que se preze, existem hoje vários dialetos SQL, cada um, evidentemente, tentando ser mais padronizado que o

outro. Neste capítulo, iremos seguir o padrão ANSI da SQL, tentando ser o mais isento possível de implementações específicas de cada fabricante.

Como vimos no capítulo anterior, o modelo relacional é constituído basicamente de tabelas, cada qual contendo linhas (registros, tuplas) e colunas. Os registros na tabela não são ordenados e sua localização se faz por meio de um campo-chave, ou seja, um campo que assume o papel de chave primária da tabela. É por intermédio dessa chave que se identifica uma, e somente uma, ocorrência do valor contido no campo.

Uma das razões da popularidade dos sistemas relacionais é a sua facilidade de manipulação e entendimento.

A linguagem SQL foi desenvolvida especialmente para o ambiente relacional, podendo ser adaptada a qualquer ambiente não relacional.

14.2 - A Linguagem SQL

A idéia original da SQL só previa seu uso de forma interativa. Após sofrer alguns acréscimos, ela passou também a ter capacidade de ser utilizada em linguagens hospedeiras, tais como: COBOL, FORTRAN, "C", etc.

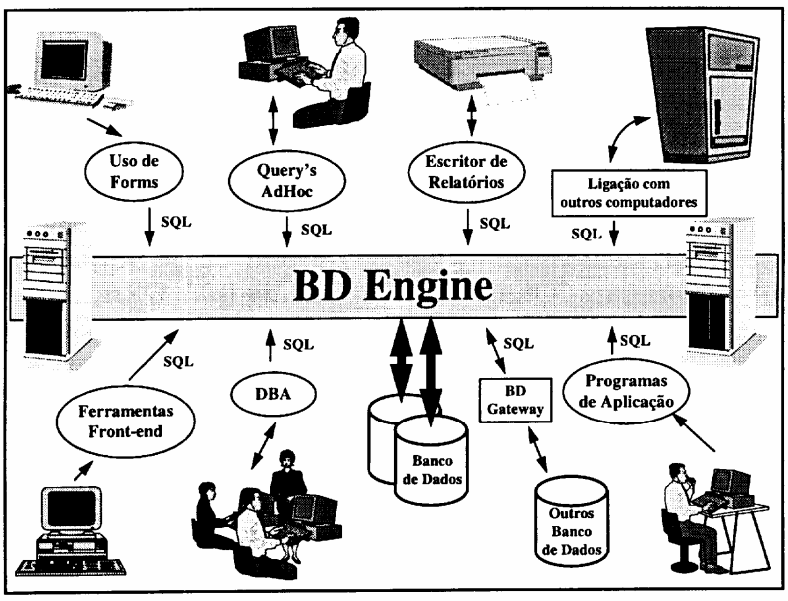


Figura 14.1

Atualmente, a linguagem SQL assume um papel muito importante nos stemas de gerenciamento de banco de dados, podendo ter muitos enfoques, ›mo apresenta a figura 14.1:

- **Linguagem interativa de consulta** *(query AdHoc)*- Por meio de comandos SQL, os usuários podem montar consultas poderosas sem a necessidade da criação de um programa, podendo utilizar *Forms* ou ferramentas de montagem de relatório;

- **Linguagem de programação para acesso a banco de dados** - Comandos SQL embutidos em programas de aplicação que acessam os dados armazenados;

- **Linguagem de administração de banco de dados** - O responsável pela administração do banco de dados (DBA) pode utilizar comandos SQL para realizar suas tarefas;

- **Linguagem cliente/servidor** - Os programas (cliente) dos computadores pessoais usam comandos SQL para se comunicarem por meio de uma rede local, compartilhando os dados armazenados em um único local (servidor). A arquitetura cliente/servidor minimiza o tráfego de dados pela rede;

- **Linguagem para banco de dados distribuído** - A SQL auxilia na distribuição dos dados por meio de vários nós conectados ao sistema de computação. Auxilia também na comunicação de dados com outros sistemas;

- **Caminho de acesso a outros bancos de dados em diferentes máquinas** - A SQL auxilia na conversão entre diferentes produtos de banco de dados colocados em diferentes máquinas (de micro até mainframe).

Por ser uma linguagem de numerosas aplicações, a SQL pode anipular objetos de diferentes classes (figura 14.2) entre as funções de um ;BD:

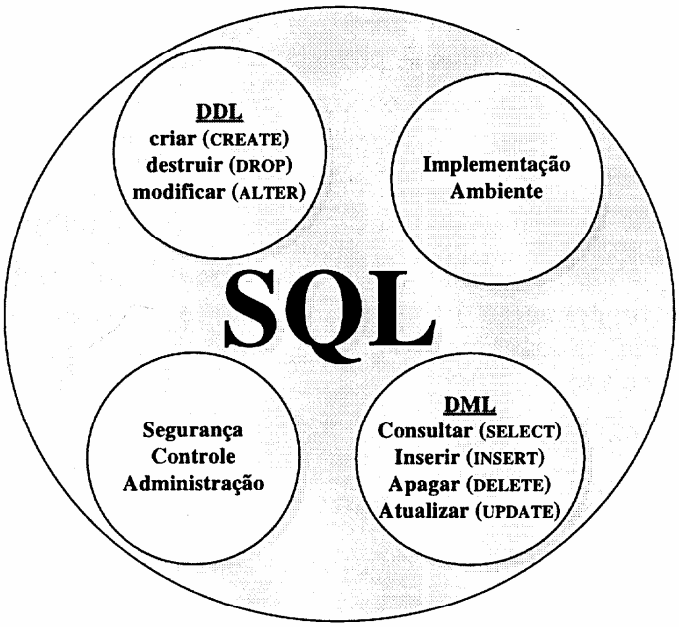


Figura 14.2

- **Definição de dados (DDL)** - permite ao usuário a definição da estrutura e organização dos dados armazenados, e as relações que existem entre eles;
- **Manipulação de dados (DML)** - permite ao usuário ou a um programa de aplicação, a inclusão, remoção, seleção ou atualização de dados previamente armazenados no banco;
- **Controle de acesso** - protege os dados de manipulações não autorizadas;
- **Compartilhamento de dados** - coordena o compartilhamento dos dados por usuários concorrentes, sem contudo interferir na ação de cada um deles;
- **Integridade dos dados** - auxilia no processo de definição da integridade dos dados, protegendo contra corrupções, inconsistências e falhas do sistema de computação.

14.3 - Vantagens e Desvantagens da Linguagem SQL

Com o uso e a padronização da SQL, algumas vantagens são diretas:

- **Independência de fabricante** - A SQL é oferecida em praticamente todos os SGBD's, e os que ainda não têm estão se encaminhando para lá. Com isso posso mudar de SGBD sem me preocupar com o novo que vai chegar;
- **Portabilidade entre computadores** - A SQL pode ser utilizada desde um computador pessoal, passando por uma estação de trabalho, até um computador de grande porte;
- **Redução dos custos com treinamento** - Baseado no item anterior, as aplicações podem se movimentar de um ambiente para o outro sem que seja necessária uma reciclagem da equipe de desenvolvimento;
- **Inglês estruturado de alto nível** - A SQL é formada por um conjunto bem simples de sentenças em inglês, oferecendo um rápido e fácil entendimento;
- **Consulta interativa** - A SQL provê um acesso rápido aos dados, fornecendo respostas ao usuário, a questões complexas, em minutos ou segundos;
- **Múltiplas visões dos dados** - A SQL permite ao criador do banco de dados levar diferentes visões dos dados a diferentes usuários;
- **Definição dinâmica dos dados** – Por meio da SQL, podem-se alterar, expandir ou incluir, dinamicamente, as estruturas dos dados armazenados com a máxima flexibilidade;

Apesar de todas essas vantagens, algumas críticas são dirigidas à SQL:

- A padronização leva a uma, natural, inibição da criatividade, pois quem desenvolve aplicações fica preso a soluções padronizadas, não podendo sofrer melhorias ou alterações;
- A SQL está longe de ser uma linguagem relacional ideal - Segundo C. J. Date, em seu livro "Relational Database: selected Writing (Addison - Werley, 1986)", algumas críticas são feitas à linguagem SQL:

- Falta de ortogonalidade nas expressões, funções embutidas, variáveis indicadoras, referência a dados correntes, constante NULL, conjuntos vazios, etc.;
- Definição formal da linguagem após sua criação;
- Discordância com as linguagens hospedeiras;
- Falta de algumas funções;
- Erros (valores nulos, índices únicos, cláusula FROM, etc.);
- Não dá suporte a alguns aspectos do modelo relacional (atribuição de relação, join explícito, domínios, etc.):

Mesmo enfrentando alguns problemas e críticas, a linguagem SQL veio para ficar, auxiliando de forma bastante profunda a vida dos usuários e analistas no trabalho de manipulação dos dados armazenados em um banco de dados relacional. E é sobre esse auxílio que este capítulo irá tratar, mostrando comandos e funcionalidades da SQL, por meio de exemplos práticos. Não iremos mostrar todos os comandos, principalmente os que foram definidos para serem utilizados dentro de uma linguagem hospedeira (cursor); apresentaremos os comandos para criação, atualização, alteração, pesquisa e eliminação de tabelas dentro de um ambiente relacional típico.

Além dos comandos da SQL padrão ANSI, vamos apresentar a sintaxe de comandos SQL efetuados no SGBD ORACLE da Oracle Corporation e MS-SQL Server 7.0, um produto da Microsoft®.

14.4 - O Exemplo

Todo o nosso percurso pela linguagem SQL será efetuado com base no exemplo de modelo de dados apresentado na figura 14.3, criado no capítulo 13 sobre Normalização.

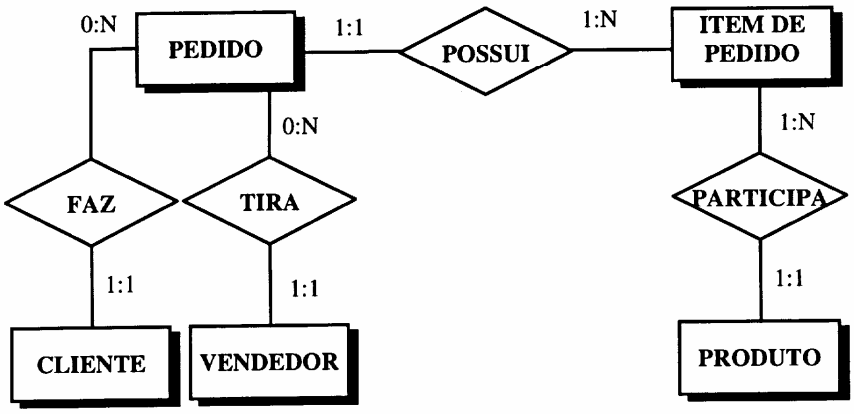


Figura 14.3

Na figura 14.4, são apresentadas as tabelas referentes ao modelo da figura 14.3.

TABELA CLIENTE

Cód. do cliente	Nome do cliente	Endereço	Cidade	CEP	UF	CGC	IE
720	Ana	Rua 17 n. 19	Niterói	24358310	RJ	12113231/0001-34	2134
870	Flávio	Av. Pres. Vargas 10	São Paulo	22763931	SP	22534126/9387-9	4631
110	Jorge	Rua Caiapo 13	Curitiba	30078500	PR	14512764/9834-9	
222	Lúcia	Rua Itabira 123 Loja 9	Belo Horizonte	22124391	MG	28315213/9348-8	2985
830	Maurício	Av. Paulista 1236 sl/2345	São Paulo	3012683	SP	32816985/7465-6	9343
130	Edmar	Rua da Praia sn/	Salvador	30079300	BA	23463284/234-9	7121
410	Rodolfo	Largo da Lapa 27 sobrado	Rio de Janeiro	30078900	RJ	12835128/2346-9	7431
20	Beth	Av. Climério n. 45	São Paulo	25679300	SP	32485126/7326-8	9280
157	Paulo	Tv. Moraes c/3	Londrina		PR	32848223/324-2	1923
180	Lívio	Av. Beira Mar n. 1256	Florianópolis	30077500	SC	12736571/2347-4	
260	Susana	Rua Lopes Mendes 12	Niterói	30046500	RJ	21763571/232-9	2530
290	Renato	Rua Meireles n. 123 bl.2 sl.345	São Paulo	30225900	SP	13276571/1231-4	1820
390	Sebastião	Rua da Igreja n. 10	Uberaba	30438700	MG	32176547/213-3	9071
234	José	Quadra 3 bl. 3 sl. 1003	Brasília	22841650	DF	21763576/1232-3	2931

Figura 14.4

TABELA VENDEDOR

Código do vendedor	Nome do vendedor	Salário Fixo	Faixa de Comissão
209	José	1.800,00	C
111	Carlos	2.490,00	A
11	João	2.780,00	C
240	Antônio	9.500,00	C
720	Felipe	4.600,00	A
213	Jonas	2.300,00	A
101	João	2.650,00	C
310	Josias	870,00	B
250	Maurício	2.930,00	B

TABELA PEDIDO

Número do Pedido	Prazo de Entrega	Código do Cliente	Código do Vendedor
121	20	410	209
97	20	720	101
101	15	720	101
137	20	720	720
148	20	720	101
189	15	870	213
104	30	110	101
203	30	830	250
98	20	410	209
143	30	20	111
105	15	180	240
111	20	260	240
103	20	260	11
91	20	260	11
138	20	260	11
108	15	290	310
119	30	390	250
127	10	410	11

TABELA ITEM_DO_PEDIDO

Número do pedido	Código do produto	Quantidade
121	25	10
121	31	35
97	77	20
101	31	9

Figura 14.4 (continuação)

TABELA ITEM_DO_PEDIDO (continuação)

Número do pedido	Código do produto	Quantidade
101	78	18
101	13	5
98	77	5
148	45	8
148	31	7
148	77	3
148	25	10
148	78	30
104	53	32
203	31	6
189	78	45
143	31	20
143	78	10
105	78	10
111	25	10
111	78	70
103	53	37
91	77	40
138	22	10
138	77	35
138	53	18
108	13	17
119	77	40
119	13	6
119	22	10
119	53	43
137	13	8

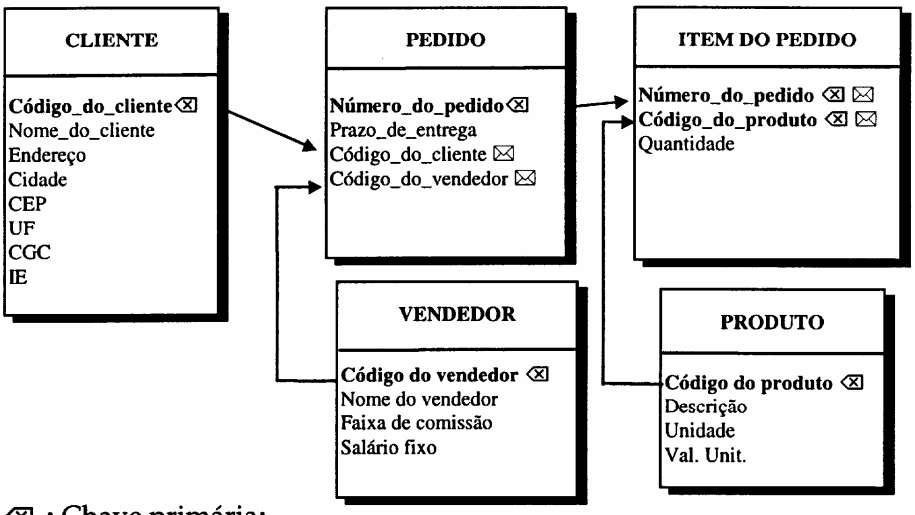
TABELA PRODUTO

Código do produto	*Unidade do* produto	*Descrição do* produto	*Valor* unitário
25	Kg	Queijo	0,97
31	BAR	Chocolate	0,87
78	L	Vinho	2,00
22	M	Linho	0,11
30	SAC	Açúcar	0,30
53	M	Linha	1,80
13	G	Ouro	6,18
45	M	Madeira	0,25
87	M	Cano	1,97
77	M	Papel	1,05

Figura 14.4 (continuação)

As informações armazenadas nas tabelas da figura 14.4 serão utilizadas pelos comandos SQL, apresentados ao longo deste capítulo.

14.4.1 - Visão Gráfica do Exemplo



⌫ : Chave primária;

⌧ : Chave estrangeira.

14.4.2 - Criação e Distribuição de Tabelas

Para podermos criar e inserir as tabelas de uma aplicação em Banco de Dados, dependendo do ambiente de SGBD que estivermos utilizando, criar o DATABASE, ou seja criar um banco de dados no qual estarão residentes as tabelas de nosso sistema.

Esta operação, no Microsoft® SQL Server™, por exemplo, é realizada normalmente por equipes de suporte, e consiste em dois passos, no mínimo, que pela ordem são:

a) Inicializar os arquivos em que serão armazenados os DATABASES de nossas aplicações (devices);

Esta é uma criação de nomes físicos e lógicos e determinação do tamanho da área em meio magnético desse device; no Microsoft®-SQL

Server™, o comando utilizado é o DISK INIT, entretanto, não sendo objeto de nosso estudo.

b) Criar os databases nos devices já criados anteriormente.

Para isto iremos usar no MS-SQL Server o comando CREATE DATABASE com a seguinte sintaxe:

```
CREATE DATABASE database_name
[ON {DEFAULT| database_device} [= size]
        [,database_device [= size] ]...]
[LOG ON database_device [= size]
        [, database_device [= size] ]...]
[FOR LOAD]
```

Um exemplo:

1. **CREATE DATABASE** vendas

Cria o database vendas no device default com tamanho default de 2 MB.

2. **CREATE DATABASE** vendas **ON** default = 256

Cria o Database Vendas no device default com 256 Mb.

3. **CREATE DATABASE** vendas **ON** *default* = 50, novosdados = 25

Cria o Database Vendas e aloca 50 Mb no device *default*, e 25 Mb no device *NovosDados*.

4. **CREATE DATABASE** vendas **ON** library_dev1 = 10 LOG ON librlog_dev2 = 4

Cria o DataBase Vendas e aloca 10 Mb em *library_dev1* e coloca 4 Mb para log de transações num device separado chamado librlog_dev2.

Uma vez que já criamos o DataBase, ou seja, o nosso banco de dados da aplicação, podemos então partir para criar as nossas tabelas.

O comando **CREATE TABLE** cria a tabela solicitada e obedece à seguinte forma:

Na linguagem SQL padrão

```
CREATE TABLE <tabela>
            (<descrição das colunas>);
            (<descrição das chaves>);
```

Em que:

<tabela> - É o nome da nova tabela a ser criada;

<descrição das colunas> - É uma lista de colunas (campos) e seus respectivos tipos de dados (smallint, char, money, varchar, integer, decimal, float, real, date, time, timestamp, logical).

<descrição das chaves> - É a lista de colunas que são tratadas como chave estrangeira.

Alguns campos podem receber o valor NULL (nulo) e o campo definido como chave primária, além de não poder receber NULL, deve ser um campo UNIQUE (sem repetições - chave primária). Para o banco de dados da figura acima temos os seguintes comandos:

```
CREATE TABLE CLIENTE
(
Código_cliente    smallint    not null unique,
nome_cliente      char(20),
endereço          char(30),
cidade            char(15),
CEP               char(8),
UF                char(2),
CGC               char(20),
IE                char(20)
);

CREATE TABLE PEDIDO
(
num_pedido        int         not null    unique,
prazo_entrega     smallint    not null,
código_cliente    smallint    not null,
código_vendedor   smallint    not null,
FOREIGN KEY       (código_cliente)
                        REFERENCES CLIENTE,
```

```
FOREIGN KEY        (código_vendedor)
                        REFERENCES VENDEDOR
);

CREATE TABLE ITEM_DO_PEDIDO
(
num_pedido         int          not null unique,
código_produto     smallint     not null unique,
quantidade         decimal,
FOREIGN KEY        (num_pedido)
                   REFERENCES PEDIDO,
FOREIGN KEY        (código_produto)
                   REFERENCES PRODUTO
);
```

Observe-se que a cláusula **REFERENCE** estabelece a restrição de integridade referencial entre as tabelas. Porém, observe sempre que só podemos incluir essas restrições se as tabelas referidas na cláusula **REFERENCE** já foram criadas antes desta.

Como em nosso exemplo a tabela PRODUTO ainda não foi criada, teríamos um erro ao executar esses comandos. Para criar a tabela ITEM DE PEDIDO, deveríamos criá-la após a tabela PRODUTO.

Da mesma forma, a tabela PEDIDO somente pode ser criada com a restrição de integridade referencial se for criada após a criação de CLIENTE e VENDEDOR.

É muito normal realizar a criação das tabelas sem especificar a cláusula REFERENCE posteriormente ao processo de criação, utilizar-se o comando **ALTER TABLE** para realizar a inserção das restrições de integridade referencial, como mostraremos quando da apresentação desse comando.

```
CREATE TABLE VENDEDOR
 (
 código_vendedor          smallint     not null,
 nome_vendedor            char(20),
 salário_fixo             money,
 faixa_comissão           char(1)
 PRIMARY KEY  (codigo_vendedor)
 );
```

Note que a chave primária já está definida juntamente com o registro da tabela. A criação do índice, que por razões óbvias deve acontecer após a tabela, naturalmente é um comando totalmente independente do primeiro create, que serviu para criar a tabela e suas característica básicas.

```
CREATE TABLE PRODUTO
(
código_produto          smallint not null unique,
unidade                 char(3),
descrição_              produto char(30),
val_unit                money
);
```

Para eliminar uma tabela criada, é utilizado o comando **DROP**:

forma: **DROP TABLE <tabela>;**

Ex.: DROP TABLE PEDIDO

- elimina a tabela de pedidos que foi previamente criada, seus dados e suas referências a outras tabelas.

14.4.3 - Extraindo Dados de uma Tabela: Comando SELECT

Uma das operações mais comuns, realizadas sobre um banco de dados, é examinar (selecionar) as informações armazenadas.

Essas operações são realizadas por meio do comando **SELECT**.

Neste item, iremos mostrar várias situações de utilização do comando **SELECT**.

O comando **SELECT** tem palavras-chaves em um comando básico:

a) **SELECT** Especifica as colunas da tabela que queremos selecionar;

b) **FROM** – Especifica as tabelas;

c) **WHERE** – Especifica as linhas.

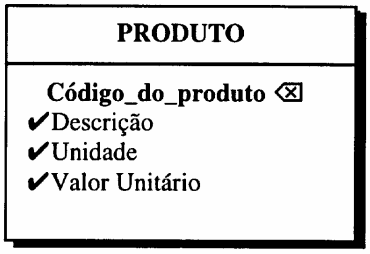
A) Selecionando Colunas Específicas da Tabela

- Sintaxe:

```
select <nome(s) da(s) coluna(s)>
    from <tabela>;
```

- Problema:
  - Listar todos os produtos com respectivas descrições, unidades e valores unitários.

    - Diagrama gráfico:



Sintaxe

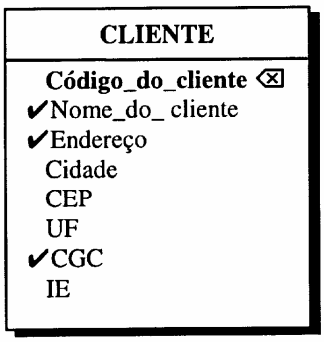
```
SELECT descrição, unidade, valor_unitário
FROM produto;
```

- A execução desse comando neste formato irá listar todas as linhas da tabela produto.
- **SELECT** sem **WHERE** lista todas as linhas de uma tabela.
- Resultado:

DESCRIÇÃO	UNIDADE	VAL UNIT
Queijo	Kg	0,97
Chocolate	BAR	0,87
Vinho	L	2,00
Linho	M	0,11
Açúcar	SAC	0,30
Linha	M	1,80
Ouro	G	6,18
Madeira	M	0,25
Cano	M	1,97
Papel	M	1,05

- Problema:
  - Listar da tabela **CLIENTE** :
  - o CGC, o nome do cliente e seu endereço.

- Diagrama gráfico:



Neste caso, cumpre destacar que estamos realizando a seleção de colunas específicas da tabela e apresentando o resultado com a disposição dos campos diferente da ordem em que se encontram-na tabela.

Sintaxe

```
SELECT CGC, nome_cliente, endereço
       FROM cliente;
```

- Resultado:

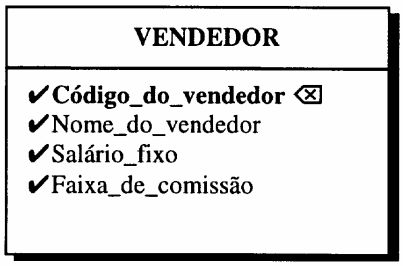
CGC	NOME DO CLIENTE	ENDEREÇO
12113231/0001-34	Ana	Rua 17 n. 19
22534126/9387-9	Flávio	Av. Pres. Vargas 10
14512764/98349-9	Jorge	Rua Caiapo 13
28315213/9348-8	Lúcia	Rua Itabira 123 Loja 9
32816985/7465-6	Maurício	Av. Paulista 1236 sl/2345
23463284/234-9	Edmar	Rua da Praia sn/
12835128/2346-9	Rodolfo	Largo da Lapa 27 sobrado
32485126/7326-8	Beth	Av. Climério n. 45
32848223/324-2	Paulo	Tv. Moraes c/3
1273657/2347-4	Lívio	Av. Beira Mar n. 1256
21763571/232-9	Susana	Rua Lopes Mendes 12
13276571/1231-4	Renato	Rua Meireles n. 123 bl.2 sl.345
32176547/213-3	Sebastião	Rua da Igreja n. 10
2176357/1232-3	José	Quadra 3 bl. 3 sl. 1003

B) Selecionando todas as Colunas da Tabela

- Sintaxe:

```
SELECT *
    FROM <tabela>;
```

- Problema:
  - Listar todo o conteúdo de vendedor.
    - Diagrama gráfico:



    - Sintaxe ANSI[1]

```
SELECT *
        FROM vendedor;
```

    - Resultado:

CÓDIGO VENDEDOR	NOME VENDEDOR	SALÁRIO FIXO	FAIXA COMISSÃO
209	José	1.800,00	C
111	Carlos	2.490,00	A
11	João	2.780,00	C
240	Antônio	9.500,00	C
720	Felipe	4.600,00	A
213	Jonas	2.300,00	A
101	João	2.650,00	C

[1] *A utilização do comando SELECT sem a cláusula WHERE causa uma desnecessária carga nos recursos de sistema. Por esta razão, em MS-SQL e ORACLE utilize sempre a cláusula WHERE.*

CÓDIGO VENDEDOR	NOME VENDEDOR	SALÁRIO FIXO	FAIXA COMISSÃO
310	Josias	870,00	B
250	Maurício	2.930,00	B

C) Alterando o Heading (Cabeçalho) da coluna

Por default, o heading (nome da coluna criado no database) apresentado na saída do SELECT é o nome da coluna na tabela. (Ver comando CREATE)

O SQL permite que se apresente a saída de um SELECT com cabeçalhos de colunas ao nosso gosto.

Sintaxe:

```
SELECT cabeçalho da coluna = nome da coluna, [, nome
da coluna]
FROM nome da tabela
```

Exemplo:

```
SELECT numero = codigo_vendedor,
nome= nome_vendedor
rendimentos = salario_fixo,
comissão = faixa_comissão
FROM vendedor
```

Resultado :

NUMERO	NOME	RENDIMENTOS	COMISSÃO
209	José	1.800,00	C
111	Carlos	2.490,00	A
11	João	2.780,00	C
240	Antônio	9.500,00	C
720	Felipe	4.600,00	A
213	Jonas	2.300,00	A
101	João	2.650,00	C
310	Josias	870,00	B
250	Maurício	2.930,00	B

D) Manipulando dados numéricos: Operadores Aritméticos

Operadores aritméticos podem ser usados sobre qualquer coluna numérica, incluindo colunas de tipo de dado *int, smallint, tinyint, float, real, money, or smallmoney*[2].

Os operadores Aritméticos são:

Símbolo	Operação	Pode ser usado com (*MS-SQL Server 7.0*)
+	Adição	*Int, smallint,tinyint, numeric,decimal, float, real, money and smallmoney*
-	Subtração	*Int, smallint,tinyint, numeric,decimal, float, real, money and smallmoney*
/	Divisão	*Int, smallint,tinyint, numeric,decimal, float, real, money and smallmoney*
*	Multiplicação	*Int, smallint,tinyint, numeric,decimal, float, real, money and smallmoney*
%	Módulo	*Int,smallint e tinyint*

Exemplo:

```
SELECT nome_vendedor,
salario_fixo = (salario_fixo * 2)
FROM vendedor
```

NOME_VENDEDOR	SALARIO_FIXO
José	3,600.00
Carlos	4,980.00
João	5,560.00
Antônio	19,000.00
Felipe	9,200.00
Jonas	4,600.00
João	5,300.00
Josias	1,740.00
Maurício	5,860.00

O resultado apresenta os salários com valores multiplicados por dois.

[2] *Datatypes de MS-SQL Server 7.0*

E) Selecionando somente algumas linhas da Tabela:

A cláusula WHERE em um comando SELECT especifica quais linhas queremos obter, baseada em condições de seleção.

Chamamos isto de observar uma seleção horizontal de informações.

♦ Sintaxe básica:

```
SELECT <nome(s) da(s) coluna(s)>
FROM <tabela>
WHERE  <condições de seleção>;
```

Comparações na Cláusula WHERE

♦ **WHERE** <nome da coluna> <operador> <valor>

E.1) Operadores de Comparação

- = ⇨ Igual
- <> ou != ⇨ diferente
- < ⇨ menor do que
- > ⇨ maior do que
- >= ⇨ maior ou igual do que
- !> ⇨ não maior
- !< ⇨ não menor
- <= ⇨ menor ou igual do que

Quando a coluna é do tipo caractere, o <valor> deve estar entre aspas (');

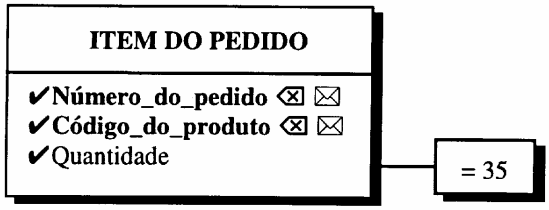
Ex.: 'PARAFUSO'

Na linguagem SQL, existe a diferenciação entre maiúsculas e minúsculas em alguns SGBDs, logo 'PARAFUSO' é diferente de 'parafuso'.

♦ Problema:

– Listar o num_pedido, o código_produto e a quantidade dos itens do pedido com a quantidade igual a 35 da tabela item_do_pedido.

- Diagrama gráfico:



- Sintaxe:

```
SELECT num_pedido, código_produto, quantidade
FROM item_do_pedido
WHERE quantidade = 35;
```

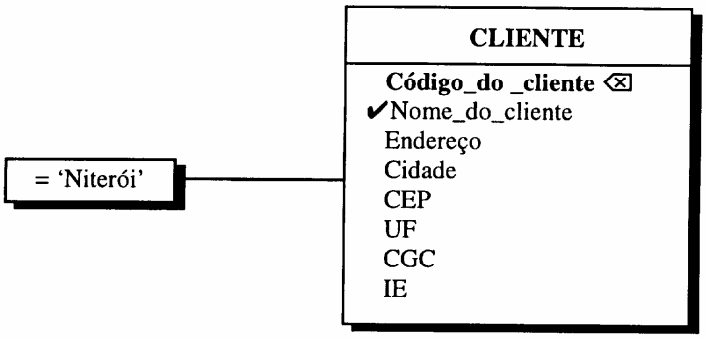
- Resultado:

NÚMERO DO PEDIDO	CÓDIGO DO PRODUTO	QUANTIDADE
121	31	35
138	77	35

♦ Problema:

– Quais os clientes que moram em Niterói?

- Diagrama gráfico:



Sintaxe:

```
SELECT nome_cliente
FROM cliente
WHERE cidade = 'Niterói';
```

- Resultado:

NOME CLIENTE
Ana
Susana

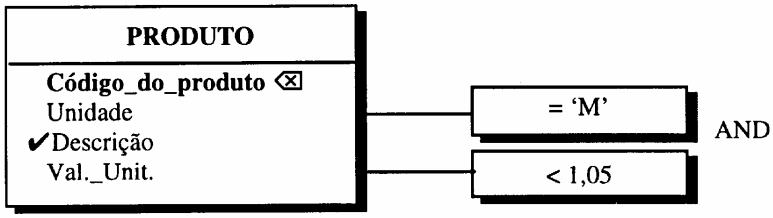
E.2) Operadores Lógicos

- AND ⇨ "e" lógico
- OR ⇨ "ou" lógico
- NOT ⇨ negação

♦ Problema:

– Listar os produtos que tenham unidade igual a 'M' e valor unitário igual a R$ 1,05 da tabela produto.

- Diagrama gráfico:



- Sintaxe:

```
SELECT descrição_produto
FROM produto
WHERE unidade = 'M' AND val_unit = 1.05;
```

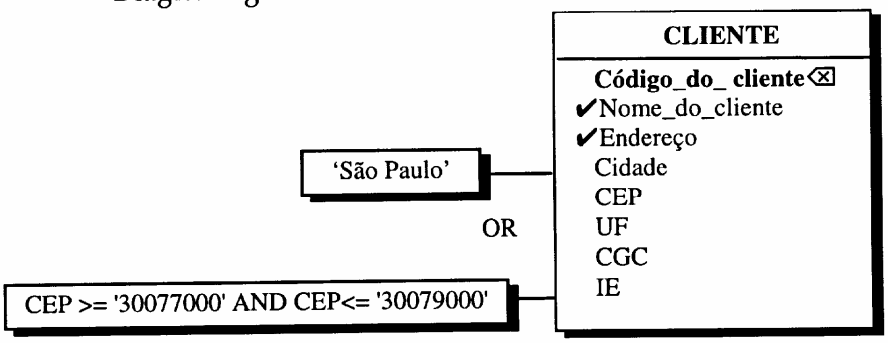
- Resultado:

Descrição_produto
Papel

♦ Problema:

– Liste os clientes e seus respectivos endereços, que moram em 'SÃO PAULO' ou estejam na faixa de CEP entre '30077000' e '30079000'.

- Diagrama gráfico:



- Sintaxe:

```
SELECT nome_cliente, endereço
FROM cliente
        WHERE (CEP >= '30077000' AND CEP <= '30079000')
      OR cidade = 'São Paulo';
```

- Resultado:

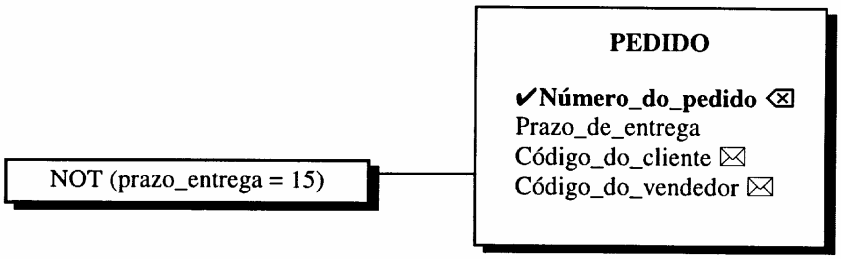
NOME CLIENTE	ENDEREÇO CLIENTE
Flávio	Av. Pres. Vargas 10
Jorge	Rua Caiapo 13
Maurício	Av. Paulista 1236 sl/2345
Rodolfo	Largo da Lapa 27 sobrado
Beth	Av. Climério n. 45
Lívio	Av. Beira Mar n. 1256
Renato	Rua Meireles n. 123 bl.2 sl.345

A utilização dos parênteses é fundamental para a construção correta da sentença, pois sem eles as consultas podem ser analisadas de forma errada, devido à prioridade do operador AND ser maior que a prioridade do operador OR.

♦ Problema:

– Mostrar todos os pedidos que não tenham prazo de entrega igual a 15 dias.

- Diagrama gráfico:



- Sintaxe:

```
SELECT num_pedido
FROM pedido
WHERE  NOT (prazo_entrega = 15);
```

Ou podemos alternativamente utilizar um operador de comparação <> que irá realizar a mesma operação de seleção.

- Sintaxe2 :

```
SELECT num_pedido
FROM pedido
WHERE (prazo_entrega <> 15);
```

- Resultado:

Num_pedido
121
97
137
148
104
203
98
143
111
103
91
138
119
127

E.3) Operadores Between e NOT Between

- Sintaxe:

– **WHERE** <nome da coluna> **BETWEEN** <valor1> AND <valor2>
– **WHERE** <nome da coluna> **NOT BETWEEN** <valor1> AND <valor2>

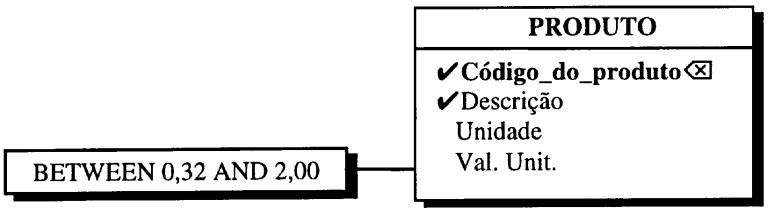
Este operador propicia a pesquisa por uma determinada coluna e selecionando as linhas cujo valor da coluna esteja dentro de uma faixa determinada de valores, sem a necessidade dos operadores >=, <= e AND. Tanto o VALOR1 quanto o VALOR2 têm de ser do mesmo tipo de dado da coluna.

[3]Quando executado sobre colunas do tipo *char, varchar, text, datetime e smalldatetime* devemos colocar estes valores entre aspas.

♦ Problema:

– Listar o código e a descrição dos produtos que tenham o valor unitário na faixa de R$ 0,32 até R$ 2,00.

- Diagrama gráfico:



- Sintaxe:

```
SELECT codigo_produto, descrição_produto
FROM produto
WHERE val_unit between 0.32 and 2.00;
```

- Resultado:

CÓDIGO DO PRODUTO	DESCRIÇÃO
25	Queijo
31	Chocolate

[3]*Referente a MS-SQL Server ™ 7.0.*

CÓDIGO DO PRODUTO	DESCRIÇÃO
78	Vinho
53	Linha
87	Cano
77	Papel

E.4) Operadores baseados em string de caracteres LIKE e NOT LIKE

- Sintaxe:

– **WHERE** <nome da coluna> **LIKE** <valor>;
– **WHERE** <nome da coluna> **NOT LIKE** <valor>;

Os operadores LIKE e NOT LIKE só trabalham sobre colunas que sejam do tipo CHAR. Eles têm praticamente o mesmo funcionamento que os operadores = e < > , porém o poder desses operadores está na utilização dos símbolos (%) e (_) que podem fazer o papel de "curinga":

- % - substitui uma palavra
- _ - substitui um caractere

Ex.: Like 'LÁPIS %' pode enxergar os seguintes registros:

- 'LÁPIS PRETO',
- 'LÁPIS CERA',
- 'LÁPIS BORRACHA'

Ou seja, todos os registros que contenham 'LÁPIS' seguido de qualquer palavra ou conjunto de caracteres.

LIKE 'BROCA N_' pode enxergar os seguintes registros:

- 'BROCA N1',
- 'BROCA N9',
- 'BROCA N3'

LIKE '%ão_' pode enxergar qualquer nome que termine em "*ão*".

LIKE '[CM]%' permite enxergar qualquer nome que comece com 'C ' ou com 'M'.

LIKE '[C-X]%' permite enxergar qualquer nome que comece com 'C ' até 'X.

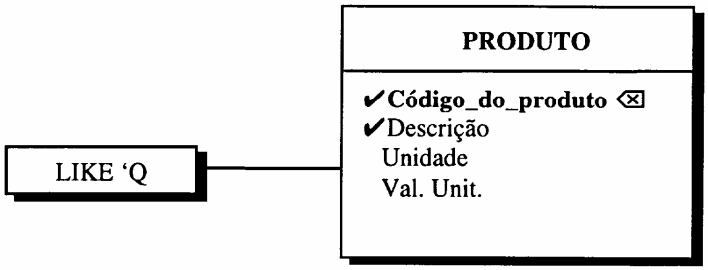
LIKE 'M[^o]%' permite enxergar qualquer nome que comece com 'M ' e não tenha a letra 'o' como segunda letra.

Vamos ver mais alguns exemplos de utilização da cláusula LIKE.

♦ Problema:

– Listar todos os produtos que tenham o seu nome começando por Q.

• Diagrama gráfico:



• Sintaxe:

```
SELECT Código_produto, descrição_produto
FROM produto
WHERE descrição_produto LIKE 'Q_';
```

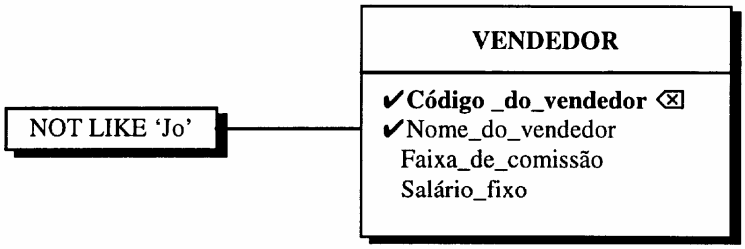
• Resultado:

CÓDIGO DO PRODUTO	DESCRIÇÃO
25	Queijo

♦ Problema:

– Listar os vendedores que não começam por 'Jo'.

- Diagrama gráfico:



Sintaxe ANSI

```
SELECT Código_vendedor, nome_vendedor
FROM vendedor
WHERE nome_vendedor NOT LIKE 'Jo%';
```

Sintaxe Microsoft® SQL Server™

```
SELECT Código_vendedor, nome_vendedor
FROM vendedor
WHERE nome_vendedor LIKE '[^Jo]%';
```

- Resultado **NOT LIKE**:

CÓDIGO VENDEDOR	NOME VENDEDOR
111	Carlos
240	Antônio
720	Felipe
250	Maurício

E.5) Operadores baseados em listas IN e NOT IN

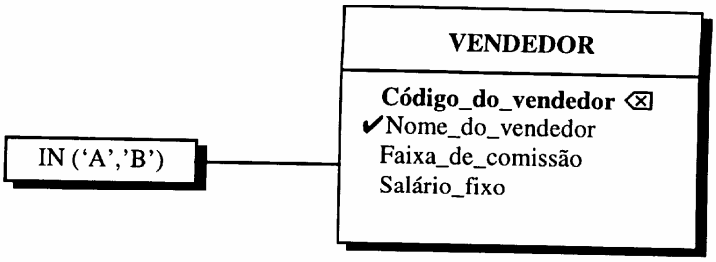
- **WHERE** <nome da coluna> **IN** <valores>;
- **WHERE** <nome da coluna> **NOT IN** <valores>;

Esses operadores pesquisam registros que estão ou não contidos no conjunto de <valores> fornecido. Estes operadores minimizam o uso dos operadores =, <>, AND e OR.

♦ Problema:

- Listar os vendedores que são da faixa de comissão A e B.

- Diagrama gráfico:



Sintaxe:

```
SELECT nome_vendedor
FROM vendedor
WHERE faixa_comissão IN ('A', 'B');
```

- Resultado:

NOME VENDEDOR
Carlos
Felipe
Jonas
Josias
Maurício

E.6) Operadores baseados em valores desconhecidos :IS NULL e IS NOT NULL

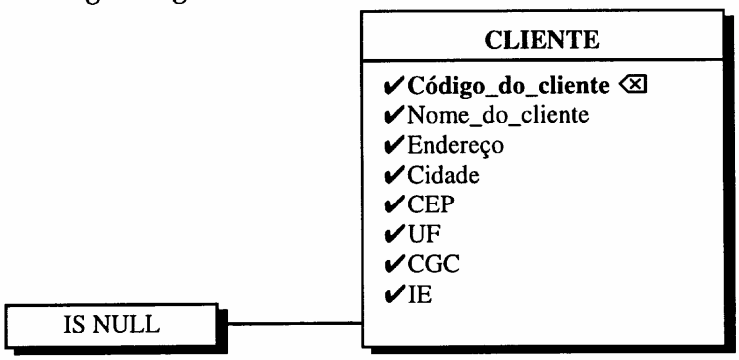
- **WHERE** <nome da coluna> **IS NULL**;
- **WHERE** <nome da coluna> **IS NOT NULL**;

A utilização do valor nulo (NULL) é muito problemática, pois cada implementação da linguagem pode adotar qualquer representação para o valor nulo.

O resultado da aplicação destes operadores permite o tratamento de valores nulos em colunas de uma tabela, selecionando as linhas correspondentes.

- Problema:
  - Mostrar os clientes que não tenham inscrição estadual.

- Diagrama gráfico:



- Sintaxe:

```
SELECT *
     FROM cliente
          WHERE IE IS NULL;
```

- Resultado:

Código Cliente	Nome Cliente	Endereço	Cidade	Cep	UF	CGC	IE
110	Jorge	R. Caiapo 13	Curitiba	30078500	PR	145127645/983493-9	
180	Lívio	Av. Beira Mar 1256	Florianópolis	30077500	SC	127365713/2347-4	

F) Ordenando os Dados Selecionados

Quando se realiza uma seleção, os dados recuperados não estão ordenados. Os dados são recuperados pela ordem em que se encontram dispostos fisicamente na tabela do SGBD.

A SQL prevê a cláusula ORDER BY para realizar uma ordenação dos dados selecionados.

♦ Sintaxe básica:

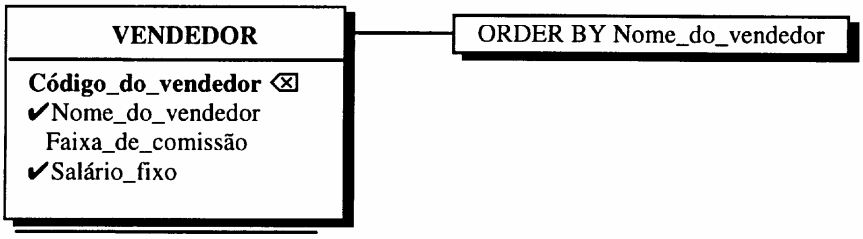
```
SELECT <nome da(s) coluna(s)>
FROM <tabela>
WHERE <condição(ões)> >
ORDER BY <nome da(s) coluna(s)>   | ASC
         ou                       | DESC
ORDER BY <número da coluna>
```

A informação <número da coluna> se refere à posição relativa das colunas quando for apresentado o resultado da consulta, e não à posição na tabela original, contada da esquerda para a direita. As palavras ASC e DESC significam, respectivamente, ascendente e descendente. A forma ascendente de ordenação é assumida como padrão.

♦ Problema:

- Mostrar em ordem alfabética a lista de vendedores e seus respectivos salários fixos.

• Diagrama gráfico:



• Sintaxe:

```
SELECT nome_vendedor, salário_fixo
FROM vendedor
ORDER BY nome_vendedor;
```

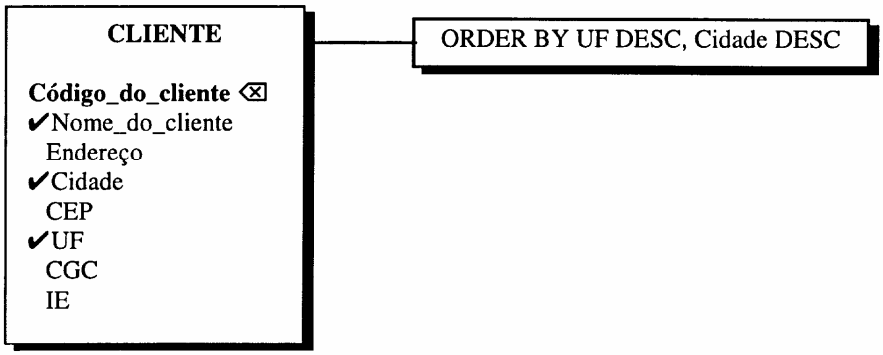
• Resultado:

NOME VENDEDOR	SALÁRIO FIXO
Antônio	9.500,00
Carlos	2.490,00
Felipe	4.600,00
João	2.780,00
João	2.650,00
Jonas	2.300,00
José	1.800,00
Josias	870,00
Maurício	2.930,00

♦ Problema:

- Listar os nomes, cidades e estados de todos os clientes, ordenados por estado e cidade de forma descendente.

- Diagrama gráfico:



- Sintaxe:

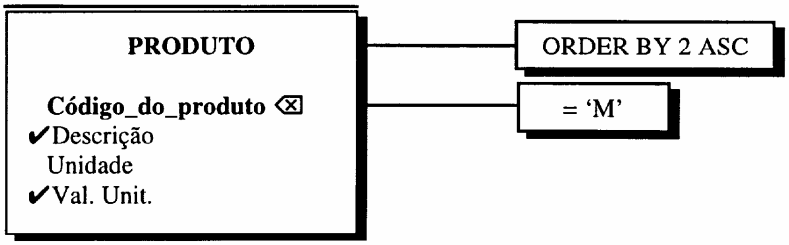
```
SELECT nome_cliente, cidade, UF
FROM cliente
ORDER BY UF DESC, cidade DESC;
```

- Resultado:

NOME CLIENTE	CIDADE	UF
Flávio	São Paulo	SP
Maurício	São Paulo	SP
Beth	São Paulo	SP
Renato	São Paulo	SP
Lívio	Florianópolis	SC
Rodolfo	Rio de Janeiro	RJ
Ana	Niterói	RJ
Susana	Niterói	RJ
Paulo	Londrina	PR
Jorge	Curitiba	PR
Sebastião	Uberaba	MG
Lúcia	Belo Horizonte	MG
José	Brasília	DF
Edmar	Salvador	BA

- Problema:
  - Mostrar a descrição e o valor unitário de todos os produtos que tenham a unidade 'KG', em ordem de valor unitário ascendente.

- Diagrama gráfico:



- Sintaxe:

```
SELECT descrição, val_unit
FROM produto
WHERE unidade = 'M'
ORDER BY 2 ASC;
```

- Resultado:

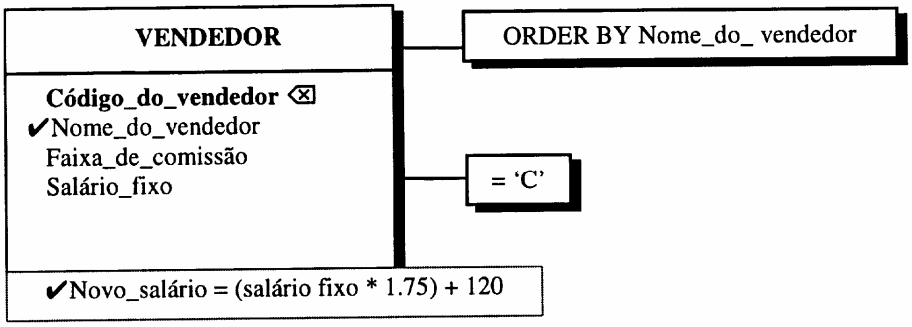
DESCRIÇÃO	VALOR UNITÁRIO
Linho	0,11
Madeira	0,25
Papel	1,05
Linha	1,80
Cano	1,97

E) Realizando Cálculos com Informação Selecionada

Com a linguagem SQL pode-se criar um campo que não pertença à tabela original, e seja fruto de cálculo sobre alguns campos da tabela.

- Problema:
  - Mostrar o novo salário fixo dos vendedores, de faixa de comissão 'C', calculado com base no reajuste de 75% acrescido de R$ 120,00 de bonificação. Ordenar pelo nome do vendedor.

- Diagrama gráfico:



- Sintaxe:

```
SELECT nome_vendedor,
novo_salário = (salário_fixo * 1.75) + 120
FROM vendedor
WHERE faixa comissão = 'C'
ORDER BY nome_vendedor;
```

- Resultado:

NOME VENDEDOR	NOVO SALÁRIO
Antônio	16.745,00
João	4.985,00
João	4.757,50
José	3270,00

G) Utilizando Funções de Agregação sobre Conjuntos

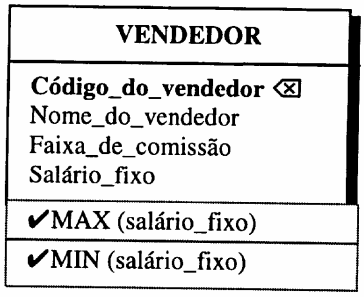
As funções de agregação resultam sempre em uma nova coluna no resultado da pesquisa.

G.1) Buscando Máximos e Mínimos (MAX, MIN)

♦ Problema:

– Mostrar o menor e o maior salários da tabela vendedor.

- Diagrama gráfico:



- Sintaxe:

```
SELECT MIN(salário_fixo), MAX(salário_fixo)
FROM vendedor;
```

- Resultado:

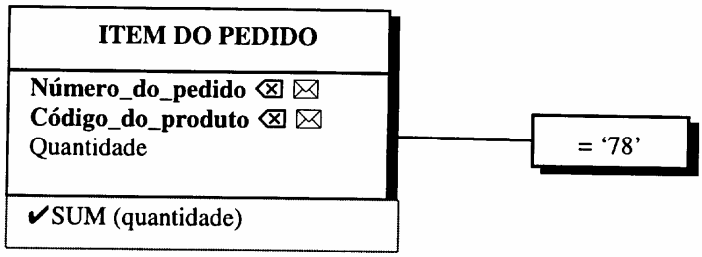
MIN(salário fixo)	MAX(salário fixo)
870,00	9.500,00

F.2) Totalizando dos valores de Colunas (SUM)

♦ Problema:

– Mostrar a quantidade total pedida para o produto 'VINHO' de código '78' na tabela item_de_pedido.

- Diagrama gráfico:



- Sintaxe:

```
SELECT SUM(quantidade),
FROM item_pedido
WHERE código_produto = '78';
```

- Resultado:

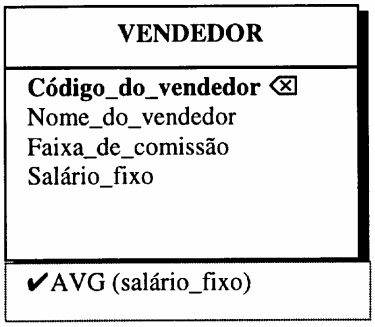
SUM(Quantidade)
183

F.3) Calculando Médias (AVG)

♦ Problema:

– Qual a média dos salários fixos dos vendedores?

- Diagrama gráfico:



- Sintaxe:

```
SELECT AVG(salário_fixo),
FROM vendedor;
```

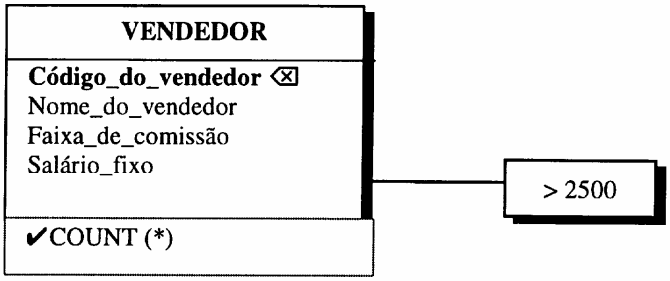
- Resultado:

AVG(salário fixo)
3.324,44

F.4) Contando os Registros (COUNT)

- Problema:
  - Quantos vendedores ganham acima de R$ 2.500,00 de salário fixo?

- Diagrama gráfico:



Sintaxe:

```
SELECT COUNT(*),
        FROM vendedor
        WHERE salário_fixo >2500;
```

O comando COUNT, quando utilizado sem a cláusula WHERE, realiza a contagem das linhas da tabela.

- Resultado:

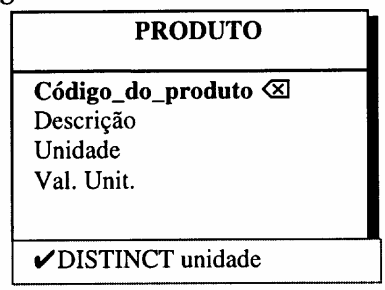
COUNT (*)
5

F.5) Utilizando a Cláusula DISTINCT

Normalmente, vários registros dentro de uma tabela podem conter os mesmos valores, com exceção da chave primária. Com isso, muitas consultas podem trazer informações erradas. A cláusula DISTINCT, aplicada em uma consulta, foi criada para não permitir que certas redundâncias, obviamente necessárias, causem problemas.A cláusula DISTINCT elimina repetições de valores em relação a uma coluna.

- Problema:
  - Quais as unidades de produtos, diferentes, na tabela produto?

- Diagrama gráfico:



- Sintaxe:

```
SELECT DISTINCT unidade,
FROM produto;
```

- Resultado:

UNIDADE
Kg
BAR
L
M
SAC
G

Importante: Com a utilização de DISTINC não se classificam os dados de saída[4].

F.6) Agrupando Informações Selecionadas (GROUP BY e HAVING)

A função de agregação por si própria produz um número simples para uma tabela.

A cláusula organiza esse sumário de dados em grupos, produzindo informação sumarizada para os grupos definidos na tabela objeto de seleção.

A cláusula HAVING realiza as restrições das linhas resultantes da mesma forma que a cláusula WHERE o faz em um SELECT.

---

[4] *Especificação para MS-SQL Server 7.0.*

Podemos igualmente continuar com a cláusula WHERE selecionando as condições da seleção.

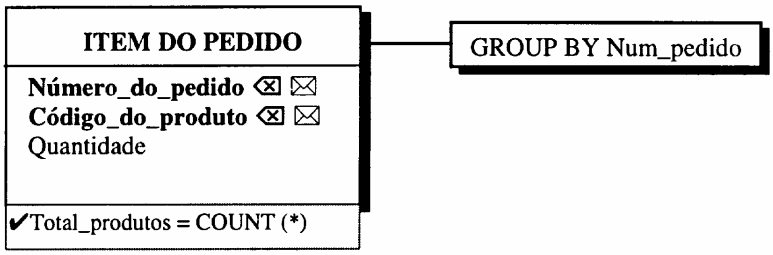
- Forma:

```
SELECT <nome da(s) coluna(s)>
FROM <tabela>
WHERE condição(ões)>
GROUP BY <nome da(s) coluna(s)>;
HAVING <condição(ões)>;
```

- Problema:

  – Listar o número de produtos que cada pedido contém.

  - Diagrama gráfico:



  - Sintaxe:

```
SELECT num_pedido,
total_produtos = COUNT(*)
FROM item_de_pedido
GROUP BY num_pedido;
```

Inicialmente, os registros são ordenados de forma ascendente por número do pedido.

Num segundo passo, é aplicada a operação COUNT(*) para cada grupo de registros que tenha o mesmo número de pedido.

Após a operação de contagem de cada grupo, o resultado da consulta utilizando a cláusula GROUP BY é apresentado.

- Resultado:

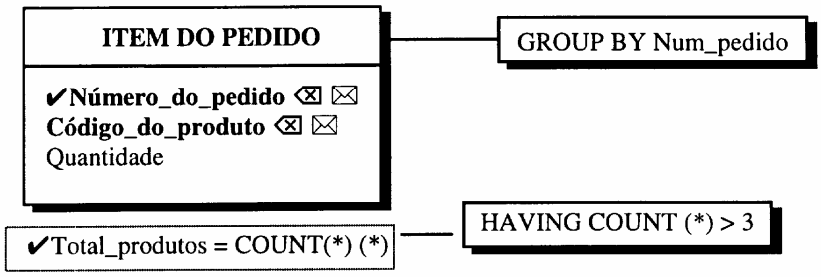
Num_pedido	Total_Produtos
91	1
97	1
98	1
101	3
103	1
104	1
105	1
108	1
111	2
119	4
121	2
138	3
143	2
148	5
189	1
203	1

Geralmente, a cláusula GROUP BY é utilizada em conjunto com as operações COUNT e AVG.

F.7) Utilizando com HAVING

- Problema:
  - Listar os pedidos que têm mais do que três produtos.

- Diagrama gráfico:



- Sintaxe:

```
SELECT num_pedido,total_produtos = COUNT(*)
FROM item_pedido
GROUP BY num_pedido;
HAVING COUNT(*) >3;
```

- Resultado:

NÚMERO DO PEDIDO	TOTAL DE PRODUTOS
119	4
148	5

A cláusula GROUP BY pode ser utilizada em conjunto com qualquer outra cláusula que já estudamos até este ponto.

G) Recuperando Dados de Várias Tabelas (JOINS)

Até agora viemos trabalhando com a recuperação de dados sobre uma única tabela, mas o conceito de banco de dados reúne, evidentemente, várias tabelas diferentes.

Para que possamos recuperar informações de um banco de dados, temos, muitas vezes, a necessidade de acessar simultaneamente várias tabelas relacionadas entre si. Algumas dessas consultas necessitam realizar uma junção (JOIN) entre tabelas, para poder extrair dessa junção as informações necessárias para a consulta formulada.

G.1) O Conceito de Qualificadores de Nome

O qualificador de nome consiste no nome da tabela seguido de um ponto e o nome da coluna na tabela, por exemplo:

- O qualificador de nome para a coluna DESCRIÇÃO da tabela PRODUTO será
  - PRODUTO.descrição

Os qualificadores de nome são utilizados em uma consulta para efetivar a junção (JOIN) entre tabelas, uma vez que o relacionamento entre tabelas é realizado por meio de chaves estrangeiras, como vimos no capítulo 6,

e isto implica na existência de colunas com mesmo nome em tabelas diferentes.

Existem duas sintaxes que vamos considerar em nosso estudo.[5]

A sintaxe ANSI SQL e a sintaxe do MS-SQL Server para a implementação de joins.

Sintaxe ANSI SQL

```
SELECT <nome_da_tabela.nome_da_coluna
      [nome_da_tabela.nome_da_coluna…..]>
      FROM    {nome_da_tabela    [tipo    de    join]
      nome_da_tabela ON condição de pesquisa
      WHERE [condição de pesquisa…..]
```

Sintaxe Microsoft®-SQL Server™

```
SELECT < nome_da_tabela.nome_da_coluna
[nome_da_tabela.nome_da_coluna…..]>
FROM <nome_da_tabela, nome_da_tabela>
WHERE <nome_da_tabela.nome_da_coluna [operador
de join] nome_da_tabela.nome_da_coluna
```

Na sintaxe ANSI, criamos uma tabela de join, *"uma join_table"*, para as linhas selecionadas conforme a cláusula WHERE. Podemos escolher o tipo de *Join* a realizar, colocando a palavra que o identifica (tipo de join).

Quando usamos INNER JOIN como tipo de join, serão incluídas somente as linhas que satisfazem a condição do join.

Quando usamos CROSS JOIN, incluímos cada uma da combinações de todas as linhas entre as tabelas.

E finalmente, quando usamos OUTER JOIN, incluímos as linhas que satisfazem a condição de JOIN e as linhas restantes de uma das tabelas do JOIN.

Na sintaxe MS-SQL Server, são comparadas as tabelas por uma coluna específica para cada tabela (chave estrangeira), linha por linha, e são listadas as linhas em que a comparação é verdadeira.

---

[5] *Aqui vale a pena destacar que existe um padrão de sintaxe:ANSI e um padrão de sintaxe MS-SQL Server para a execução de joins, sendo que não podemos usar os dois simultaneamente.*

Nos joins do MS-SQL Server, a cláusula FROM lista as tabelas envolvidas no JOIN e a cláusula WHERE especifica que linhas devem ser incluídas no conjunto resultante. Na cláusula WHERE, o operador de join é utilizado entre os componentes a serem juntados.

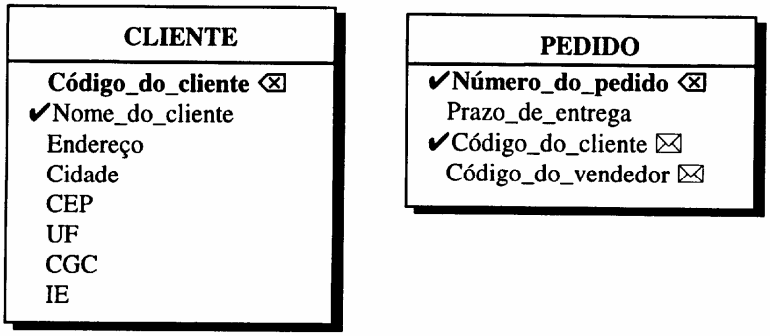
Os seguintes operadores são utilizados como *join operators* no MS-SQL Server:

Símbolo	Significado
=	Igual a
>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual
<>	Diferente

♦ Problema:

Ver os pedidos de cada cliente.

Diagrama gráfico:



G.2) Inner Joins

- Sintaxe ANSI SQL

```
SELECT Cliente.nome_cliente,
       pedido.cod_cliente,
       pedido.num_pedido
```

```
FROM cliente INNER JOIN pedido
ON cliente.codigo_do_cliente
= pedido.codigo_do_cliente
```

- Sintaxe Microsoft®-SQL Server™

```
SELECT Cliente.nome_cliente,
       pedido.cod_cliente,
       pedido.num_pedido
FROM cliente, pedido
WHERE cliente.codigo_do_cliente
= pedido.codigo_do_cliente
```

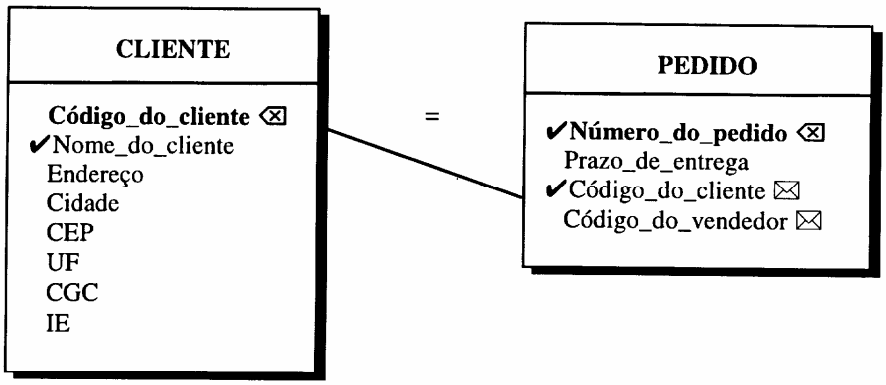
- Resultado:

Nome_Cliente	Pedido.Codigo_do_cliente	Pedido.num_pedido
Ana	720	97
Ana	720	101
Ana	720	137
Ana	720	148
Flávio	870	189
Jorge	110	104
Maurício	830	203
Rodolfo	410	121
Rodolfo	410	98
Rodolfo	410	127
Beth	20	143
Lívio	180	105
Susana	260	111
Susana	260	103
Susana	260	91
Susana	260	138
Renato	290	108
Sebastião	390	119

Nesta junção, são apresentados os pedidos de cada cliente, pois a condição de "*join*" restringe e qualifica a junção dos dados entre as tabelas. A equação apresentada na cláusula WHERE é chamada de EQUAÇÃO DE JUNÇÃO.

G.3) Cross Join ou Produto Cartesiano

- Problema:
  - Juntar Clientes com Pedidos.
- Diagrama gráfico:



- Sintaxe ANSI SQL:

```
SELECT nome_cliente,
       pedido.cod_cliente,
       num_pedido
       FROM cliente CROSS JOIN pedido
```

- Sintaxe MS- SQL Server :

```
SELECT nome_cliente,
       pedido.cod_cliente,
       num_pedido
       FROM cliente, pedido
```

- Resultado:

Nome_cliente	Pedido.codigo_do_cliente	Pedido.num_pedido
Ana	720	97
Ana	260	111
Ana	870	54
Ana	390	119
Ana	260	103
Ana	830	203
Ana	410	121

Nome_cliente	Pedido.codigo_do_cliente	Pedido.num_pedido
Ana	110	104
Ana	180	105
Ana	720	83
Ana	290	108
Ana	410	89
Flávio	720	97
Flávio	260	111
Flávio	870	54
Flávio	390	119
Flávio	260	103
Flávio	830	203
Flávio	410	121
Flávio	110	104
Flávio	720	83
Flávio	290	108
Flávio	410	89
Jorge	720	97
Jorge	260	111
Jorge	870	54
Jorge	390	119
Jorge	260	103
Jorge	830	203
Jorge	410	121
Jorge	110	104
Jorge	180	105
Jorge	720	83
Jorge	290	108
Jorge	410	89
Lúcia	720	97
Lúcia	260	111
Lúcia	870	54
Lúcia	390	119
Lúcia	260	103
⋮	⋮	⋮
⋮	⋮	⋮

Podemos observar que não existe muito proveito do resultado desse tipo de JOIN, excetuando-se quando queremos fazer referência cruzada entre duas tabelas e suas linhas todas.

G.4) Outer Joins

É a seleção em que são restritas as linhas que interessam em uma tabela, mas são consideradas todas as linhas de outra tabela.

Ou seja, queremos ver quais linhas de uma tabela estão relacionadas com a outra tabela e quais linhas não estão.

Poderíamos dizer, exemplificando no mundo real, que queremos ver quais clientes tem pedidos e quais não têm nenhum pedido.

É de muita utilidade quando queremos verificar se existem membros órfãos em um banco de dados, ou seja, chave primária e chave estrangeira sem sincronia ou simetria.

Um OUTER JOIN somente pode ser realizado entre duas tabelas, não mais que duas tabelas.

A sintaxe SQL ANSI possui três tipos de qualificadores para o OUTER JOIN:

LEFT OUTER JOIN – São incluídas todas as linhas da tabela do primeiro nome de tabela (a tabela mais à esquerda da expressão).

RIGHT OUTER JOIN – São incluídas todas as linhas da tabela do segundo nome de tabela da expressão (tabelas mais à direita da expressão).

FULL OUTER JOIN – São incluídas as linhas que não satisfazem a expressão tanto da primeira tabela quanto da segunda tabela.

O MS-SQL Server utiliza operadores para outer join:

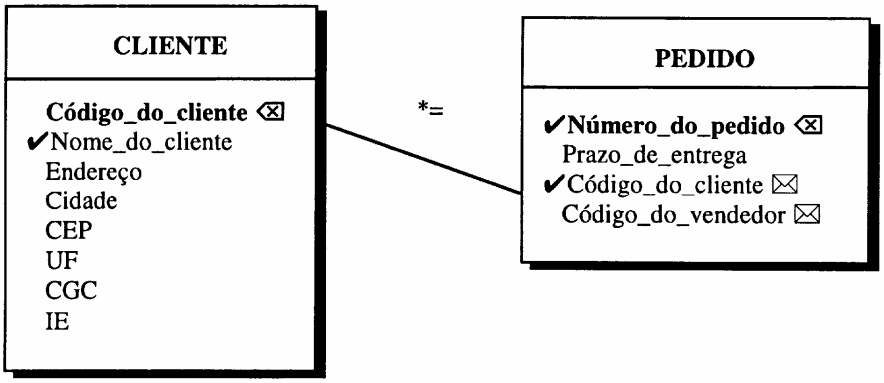
*= Inclui todas as linhas da primeira tabela da expressão

=*Inclui todas as linhas da segunda tabela da expressão

Vamos ver um exemplo de OUTER JOIN para entenderemos melhor a sua funcionalidade.

- Problema: Quais são os clientes que têm pedido e os que não têm pedido.

- Diagrama gráfico:



- Sintaxe ANSI SQL

```
SELECT nome_cliente,
       pedido.cod_cliente,
       num_pedido
       FROM cliente LEFT OUTER JOIN pedido
       ON cliente.codigo_do_cliente =
             Pedido.codigo_do_cliente
```

- Sintaxe Microsoft®- SQL Server ™:

```
SELECT nome_cliente,
       pedido.cod_cliente,
       num_pedido
       FROM cliente, pedido
       WHERE cliente.codigo_do_cliente *=
             Pedido.codigo_do_cliente
```

Resultado

Nome_Cliente	Pedido.Codigo_do_cliente	Pedido.num_pedido
Ana	720	97
Ana	720	101
Ana	720	137
Ana	720	148
Flávio	870	189
Jorge	110	104
Maurício	830	203
Rodolfo	410	121
Rodolfo	410	98

Nome_Cliente	Pedido.Codigo_ do_cliente	Pedido.num_pedido
Rodolfo	410	127
Beth	20	143
Lívio	180	105
Susana	260	111
Susana	260	103
Susana	260	91
Susana	260	138
Renato	290	108
Sebastião	390	119
Lúcia	NULL	NULL
Edmar	NULL	NULL
Paulo	NULL	NULL
José	NULL	NULL

Como os clientes Lúcia, Edmar, Paulo e José não têm nenhuma linha da tabela pedindo a informação de seu código, essas informações são apresentadas como NULL.

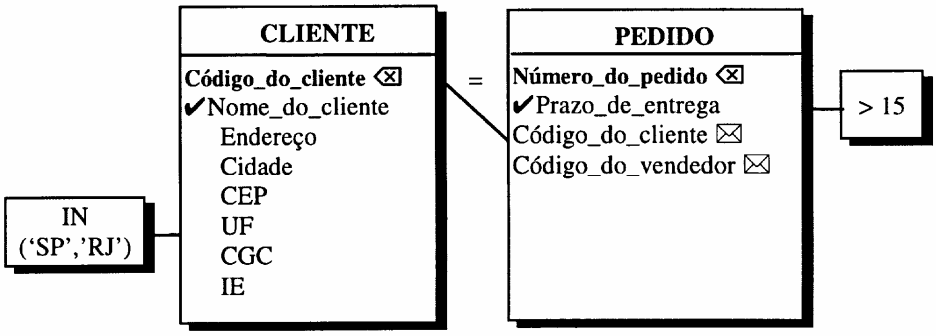
Por este motivo não devemos utilizar NULL na condição de seleção, pois teremos de utilizar os resultados mais imprevisíveis possíveis ou imaginários.

Podemos utilizar as cláusulas LIKE, NOT LIKE, IN, NOT IN, NULL, NOT NULL e misturá-las com os operadores AND, OR e NOT, dentro de uma cláusula WHERE na junção entre tabelas.

Vamos estudar comandos de seleção de dados com esses operadores.

- Problema:
  - Quais clientes têm prazo de entrega superior a 15 dias e pertencem aos estados de São Paulo ('SP') ou Rio de Janeiro ('RJ')?

- Diagrama gráfico:



- Sintaxe SQL ANSI:

```
SELECT Cliente.nome_cliente,
         pedido.cod_cliente,
         pedido.num_pedido
         FROM cliente INNER JOIN pedido
       ON cliente.codigo_do_cliente
                   = pedido.codigo_do_cliente
        WHERE UF IN ('SP', 'RJ') AND
        prazo_entrega > 15
```

- Sintaxe Microsoft®-SQL Server™

```
SELECT Cliente.nome_cliente,
         pedido.cod_cliente,
         pedido.num_pedido
         FROM cliente, pedido
       WHERE cliente.codigo_do_cliente
                 = pedido.codigo_do_cliente
                AND UF IN ('SP', 'RJ') AND
        prazo_entrega > 15
```

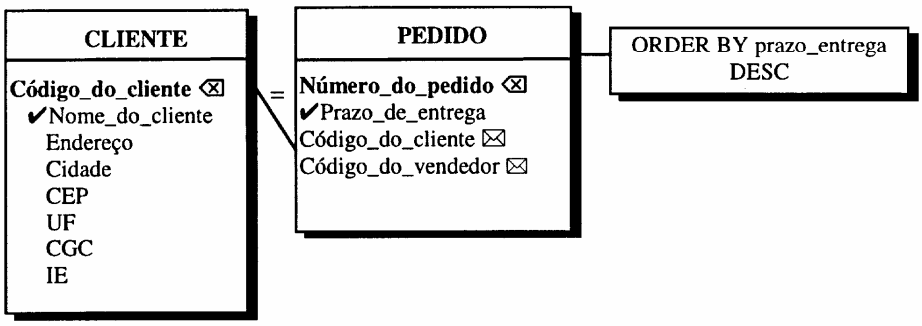
- Resultado:

NOME CLIENTE	UF	PRAZO ENTREGA
Ana	RJ	20
Maurício	SP	30
Rodolfo	RJ	20
Beth	SP	30
Susana	RJ	20

Problema:

- Mostrar os clientes e seus respectivos prazos de entrega, ordenados do maior para o menor.

- Diagrama gráfico:



- Sintaxe SQL ANSI

```
SELECT nome_cliente, prazo_entrega
FROM cliente, pedido
ON cliente.cod_cliente = pedido.cod_cliente
ORDER BY prazo_entrega desc;
```

- Sintaxe Microsoft®-SQL Server™:

```
SELECT nome_cliente, prazo_entrega
FROM cliente, pedido
WHERE cliente.cod_cliente = pedido.cod_cliente
ORDER BY prazo_entrega desc;
```

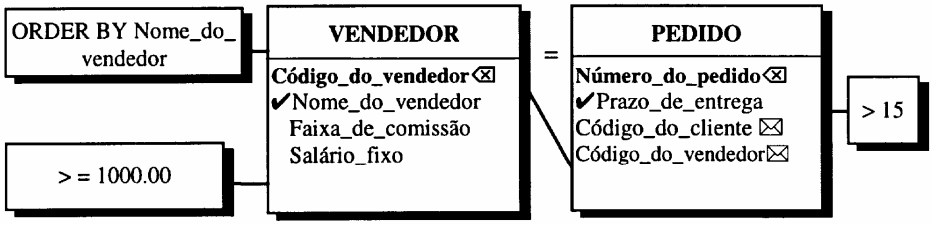
- Resultado:

NOME CLIENTE	PRAZO ENTREGA
Jorge	30
Maurício	30
Beth	30
Sebastião	30
Rodolfo	20
Ana	20
Susana	20
Ana	15
Flávio	15
Lívio	15
Renato	15
Rodolfo	10

Para que não seja necessário escrever todo o nome da tabela nas qualificações de nome, podemos utilizar ALIASES (sinônimos) definidos na própria consulta. A definição dos ALIASES é feita na cláusula FROM e utilizada normalmente nas outras cláusulas (Where, order by, group by, having, select).

- Problema:
  - Apresentar os vendedores (ordenados) que emitiram pedidos com prazos de entrega superiores a 15 dias e tenham salários fixos iguais ou superiores a R$ 1.000,00.

- Diagrama gráfico:



- Sintaxe:

```
SELECT nome_vendedor, prazo entrega
FROM vendedor V, pedido P
WHERE salário_fixo >= 1000.00 AND
      prazo_entrega > 15 AND
      V.cod_vendedor = P.cod_vendedor
ORDER BY nome_vendedor;
```

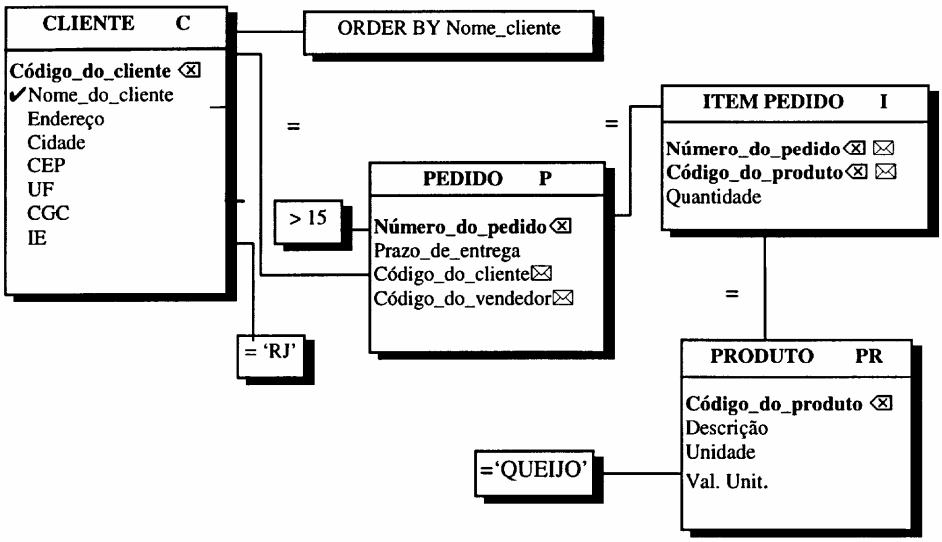
- Resultado:

NOME VENDEDOR	PRAZO ENTREGA
Antonio	20
Carlos	30
João	20
José	20
Maurício	30

G.2) Juntando mais de duas Tabelas

- Problema:
  - Mostre os clientes (ordenados) que têm prazo de entrega maior que 15 dias para o produto 'QUEIJO' e sejam do Rio de Janeiro.

- Diagrama gráfico:



Novamente vamos apresentar a sintaxe ANSI e a sintaxe específica do MS-SQL Server.

- Sintaxe ANSI

```
SELECT Cliente.nome_cliente,
  FROM cliente INNER JOIN pedido
ON cliente.codigo_do_cliente  = pedido.codigo_do_cliente
            INNER JOIN item_de_pedido
ON pedido.num_pedido = item-de_pedido.num_pedido
            INNER JOIN produto
ON item_de_pedido.cod_produto=produto.cod_produto
 WHERE Pedido.prazo_entrega > 15 AND
       Produto.Descricao ='queijo' AND
            Cliente.UF = 'RJ'
            ORDER BY Cliente.nome_cliente
```

- Sintaxe Microsoft®-SQL Server™:

```
SELECT nome_cliente
FROM cliente , pedido , item_pedido , produto
WHERE Cliente.cod_cliente = Pedido.cod_cliente
AND  Pedido.num_pedido = Item_de_pedido.num_pedido
AND Item_de_pedido.cod_produto = Produto.cod_produto
AND Pedido.prazo_entrega >15                          AND
Produto.descrição = 'Queijo'
AND Cliente.UF = 'RJ'
ORDER BY Cliente.nome_cliente;
```

- Resultado:

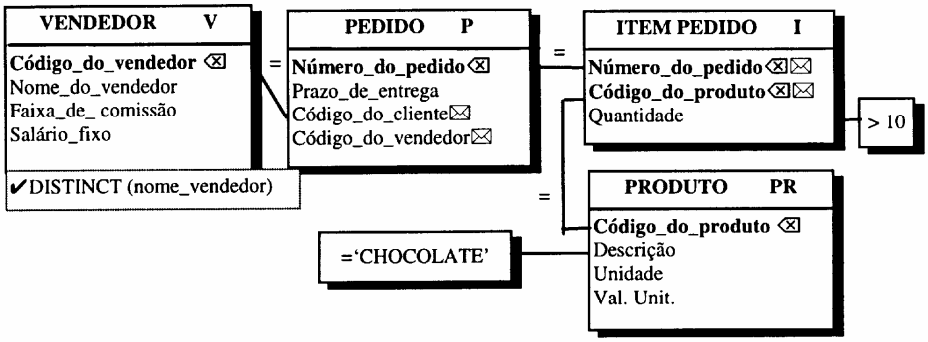
NOME CLIENTE
Ana
Rodolfo
Susana

Vamos a mais um exemplo:

♦ Problema:

– Mostre todos os vendedores que venderam chocolate em quantidade superior a 10 Kg.

- Diagrama gráfico:



- Sintaxe Microsoft®-SQL Server™:

```
SELECT DISTINCT nome_vendedor
FROM vendedor ,pedido , item_pedido , produto
WHERE Cliente.cod_vendedor = Pedido.cod_vendedor
AND  Pedido.num_pedido = Item_de_pedido.num_pedido
```

```
AND Item_de_pedido.cod_produto =Produto.cod_produto
AND  quantidade >10
AND  descrição = 'Chocolate';
```

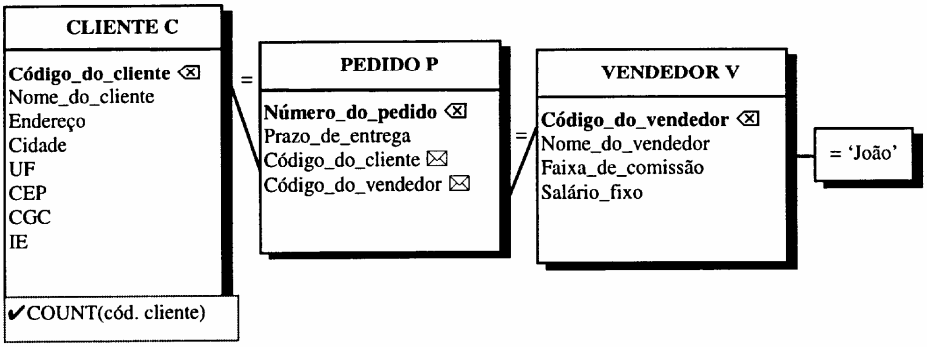
- Resultado:

NOME VENDEDOR
José
Carlos

♦ Problema:

– Quantos clientes fizeram pedido com o vendedor João?

- Diagrama gráfico:



- Sintaxe Microsoft®-SQL Server™

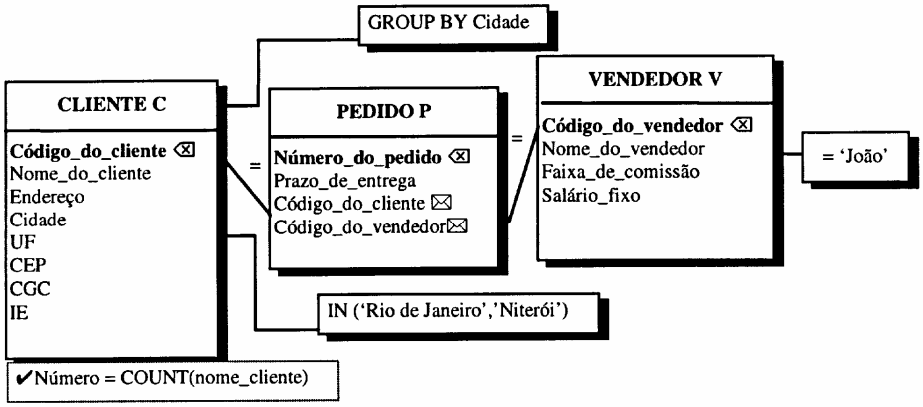
```
SELECT COUNT (cod_cliente)
FROM cliente , pedido ,vendedor
WHERE Cliente.cod_cliente = Pedido.cod_cliente AND
Pedido.cod_vendedor = Vendedor.cod_vendedor
AND Vendedor.nome_vendedor = 'João';
```

- Resultado:

COUNT (CÓDIGO CLIENTE)
4

- Problema:
  - Quantos clientes da cidade do Rio de Janeiro e de Niterói tiveram seus pedidos tirados com o vendedor João?

- Diagrama gráfico:



- Sintaxe Microsoft®-SQL Server™:

```
SELECT cidade, número = COUNT (nome_cliente)
FROM cliente , pedido , vendedor
WHERE nome_vendedor = 'João'
AND CIDADE IN ('Rio de Janeiro', 'Niterói')
AND Vendedor.cod_Vendedor = Pedido.cod_Vendedor AND
Pedido.cod_cliente = Cliente.cod_cliente
GROUP BY cidade;
```

- Resultado:

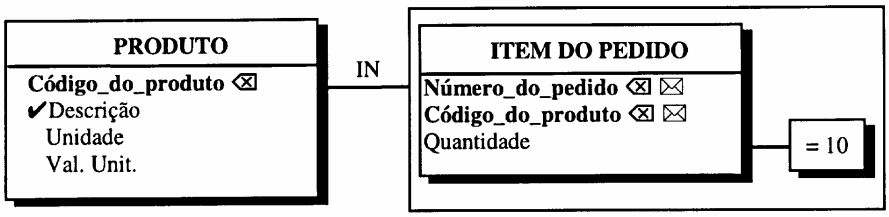
CIDADE	NÚMERO
Niterói	2
Rio de Janeiro	1

G.3) Utilizando Consultas Encadeadas (Subqueries)

Que é uma subquery? Em linhas gerais, é quando o resultado de uma consulta é utilizado por outra consulta, de forma encadeada e contida no mesmo comando SQL.

- Problema – **Utilizando IN**
  - Que produtos participam de qualquer pedido cuja quantidade seja 10?

- Diagrama gráfico:



- Sintaxe:

```
SELECT descrição
        FROM produto
        WHERE cod_produto IN
       (SELECT cod_produto
        FROM item_pedido
        WHERE quantidade = 10)
```

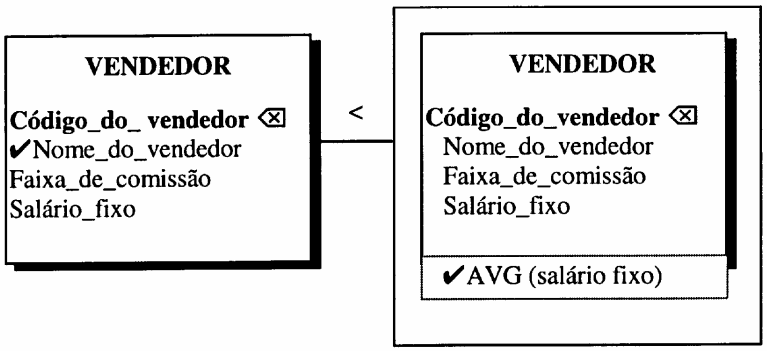
- Resultado:

DESCRIÇÃO
Queijo
Vinho
Linho

♦ Problema – **Utilizando AVG**

– Quais vendedores ganham um salário fixo abaixo da média?

- Diagrama gráfico:



- Sintaxe:

```
SELECT nome_vendedor
        FROM vendedor
        WHERE salário_fixo <
        (SELECT AVG(salário_fixo)
        FROM vendedor);
```

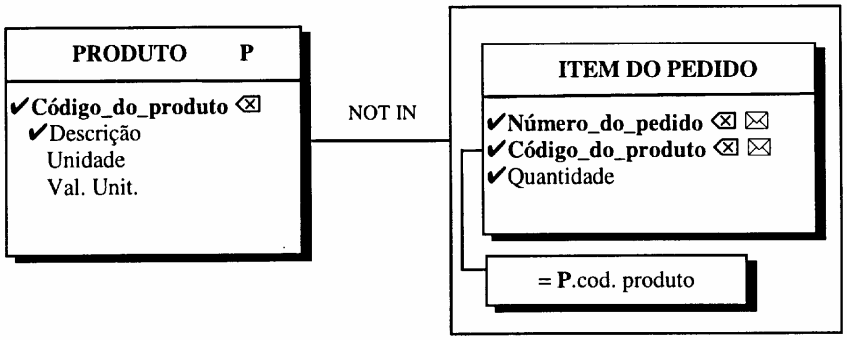
- Resultado:

Nome_Vendedor
José
Carlos
João
Jonas
Josias
Maurício

♦ Problema: - **Utilizando NOT IN**

– Quais os produtos que não estão presentes em nenhum pedido?

- Diagrama gráfico:



- Sintaxe:

```
SELECT cód_produto, descrição
FROM produto
WHERE cod.produto NOT IN
(SELECT * FROM item_pedido
WHERE item_de_pedido.cod_produto =
Produto.cod_produto)
```

- Resultado:

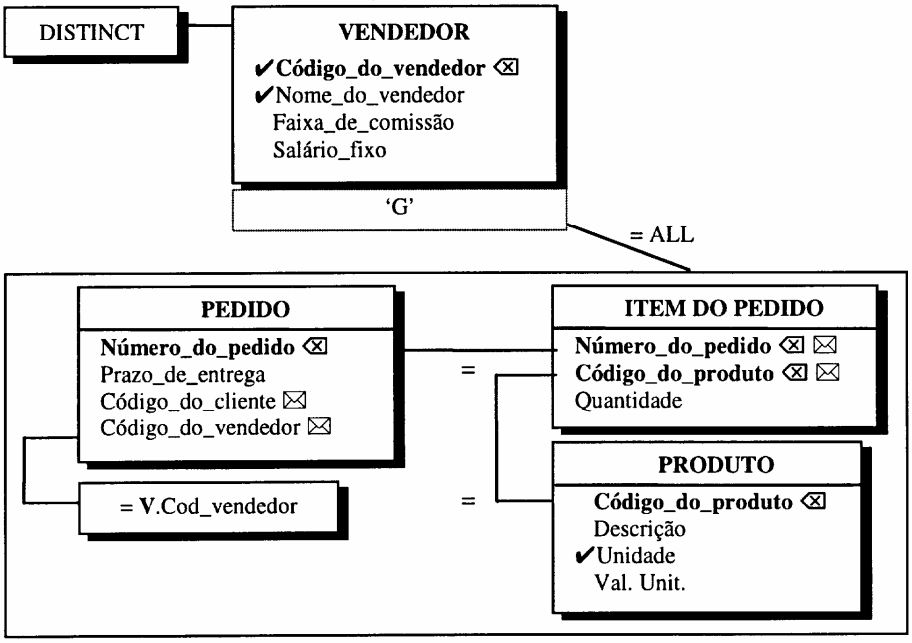
CÓDIGO PRODUTO	DESCRIÇÃO
87	Carro

Na consulta anterior, a "subquery" não é executada diretamente de uma vez só; ela é executada para cada valor do registro do produto, ou seja, para cada execução da "subquery" é utilizado um valor do código do produto dentro do produto, o qual é comparado, via condição de subquery, com vários valores em itens do pedido.

♦ Problema:

– Quais os vendedores que só venderam produtos por grama ('G')?

- Diagrama gráfico:



- Sintaxe ANSI:[6]

```
SELECT DISTINCT cod_vendedor, nome_vendedor
FROM vendedor
WHERE  unidade ALL =('G')
SELECT unidade
FROM pedido, item_pedido, produto
WHERE Pedido.num_pedido =item_de_pedido.num_pedido
AND Item_de_pedido.cod_produto = Produto.cod_produto
AND Produto.cod_vendedor = Vendedor.cod_vendedor);
```

- Resultado:

CÓDIGO VENDEDOR	NOME VENDEDOR
720	Felipe
310	Josias

Interpretando o comando aplicado:

Selecione todas as ocorrências da tabela vendedores que estão associadas à tabela pedidos, e cujas ocorrências da tabela item de pedido, relacionada com a tabela pedidos e com a tabela produtos, são de produtos com unidade 'G ', ou seja, Grama.

Condição que satisfaz a query:

Todos os itens de pedido de um pedido têm unidade G.

Mostrar o nome dos vendedores associados a esses pedidos.

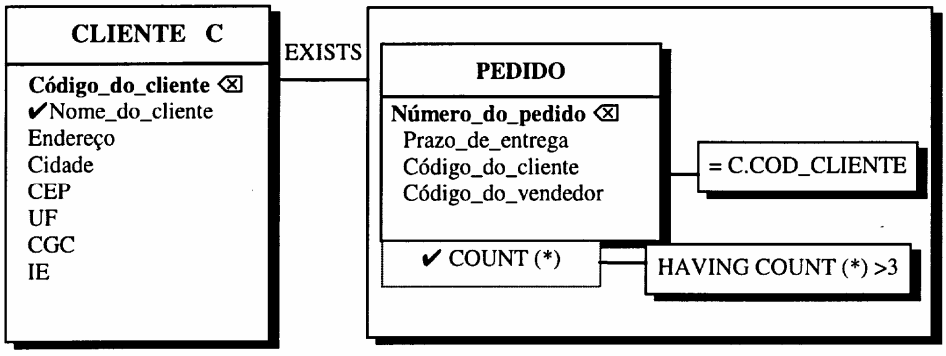
Sabemos que esse comando é complicado de entender, por isto tentamos uma forma de explicá-lo em uma linguagem mais natural, entretanto sabemos, caro leitor que mesmo assim talvez não tenhamos conseguido ser claros completamente.

♦ Problema: - **Subquery testando a existência**

– Quais clientes estão presentes em mais de três pedidos?

[6] *Não encontramos sintaxe específica neste caso para MS-SQL Server.*

- Diagrama gráfico:



- Sintaxe:

```
SELECT nome_cliente
FROM cliente
WHERE EXIST
(SELECT COUNT(*)
FROM pedido
WHERE cod_cliente = Cliente.cod_cliente
HAVING COUNT(*) >3);
```

- Resultado:

NOME CLIENTE
Ana
Susana

Entendendo o comando:

Sempre analisamos a query entre parênteses primeiro.

Neste Select, estamos contando (Count) os pedidos de cada cliente e selecionando os clientes que têm mais de três Pedidos

A query principal, por assim dizer, somente lista o nome dos clientes que EXISTEM nesse conjunto obtido (EXIST).

Destaque-se aqui a diferença entre o comando select com EXIST e com NOT IN (IN):

[7]Quando utilizamos EXIST, não colocamos nenhum nome de coluna antes da cláusula, e quando utilizamos IN ou NOT IN, somos obrigados a colocar o nome da coluna antes.

WHERE coluna IN (NOT IN) – retorna uma lista de zero ou mais valores.

WHERE EXIST (NOT EXIST) – retorna um valor falso ou verdadeiro.

- Problema - **Criar uma nova tabela com o resultado de um SELECT.**

Quando usamos um comando Select com a cláusula INTO, definimos uma tabela e colocamos os dados resultantes da query dentro dela.

Se a tabela já existir com o mesmo nome dado no comando, esse comando irá falhar.

Sintaxe Básica MS-SQL Server

SELECT <lista de colunas>

INTO <nova tabela>

FROM < lista de tabelas>

WHERE <condições de seleção>

No MS-SQL Server a cláusula INTO criará uma nova tabela permanente somente se a opção **select into/bulkcopy** estiver setada, caso contrário estaremos criando uma tabela temporária.

Se as colunas selecionadas na lista de colunas não possuírem nomes (SELECT *), as novas colunas criadas na nova tabela também não possuirão nomes e somente poderão ser selecionadas por meio de um SELECT * FROM <nome da tabela>.

---

[7] *Não encontramos referência sobre a utilização dessa cláusula no MS-SQL Server de forma diferenciada.*

14.4.4 - Inserindo, Modificando e Apagando Registros

A) Adicionando Registro à Tabela

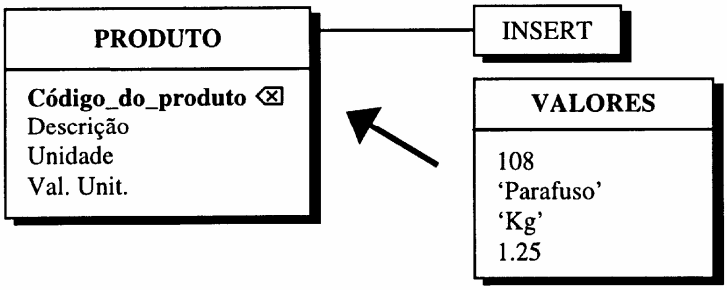
- Forma:

```
INSERT INTO <nome da tabela>
          <nome da(s) coluna(s)>)
          VALUES (<valores>);
```

- Problema:

  – Adicionar o produto 'parafuso' à tabela produto.

  - Diagrama gráfico:



Sintaxe:

```
INSERT into produto
       VALUES (108,
       'Parafuso',
       'Kg',
       1.25);
```

A cláusula VALUE especifica os dados que você deseja inserir na tabela. A cláusula VALUE é uma palavra requerida para introduzir no comando uma lista de valores para cada coluna.

Se não especificados os nomes de colunas, essa lista de valores deverá estar na ordem das colunas definidas no comando **CREATE TABLE.**

B) Adicionando Registros usando um SELECT

- Formato:

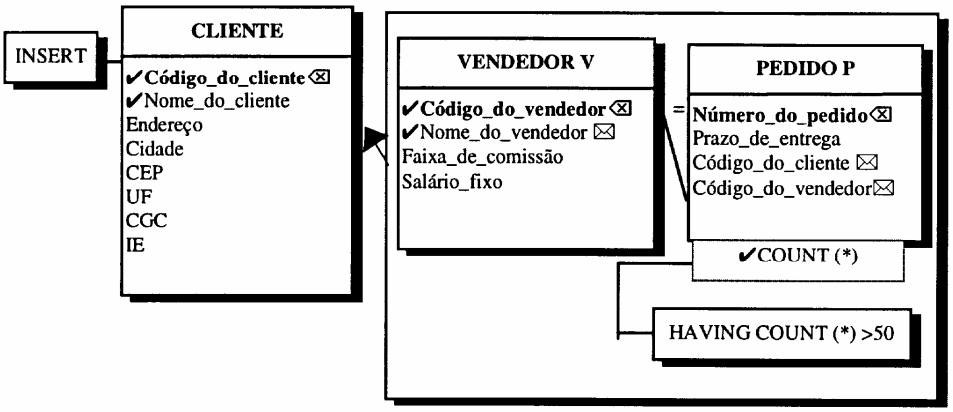
```
INSERT INTO <nome da tabela>
     (<nome da(s) coluna(s)>)
```

```
SELECT <nome da(s) coluna(s)>
FROM <nome da tabela>
WHERE <condição>;
```

♦ Problema:

– Cadastrar como cliente os vendedores que emitiram mais de 50 pedidos. Usar para código de cliente o mesmo código de vendedor.

- Diagrama gráfico:



- Sintaxe ANSI 92:

```
INSERT into cliente (cod_cliente, nome_cliente)
SELECT  cod_vendedor, nome_vendedor, COUNT(*)
FROM  vendedor , pedido
WHERE Vendedor.cod_vendedor = Pedido.cod_vendedor
HAVING COUNT(*) > 50;
```

- Sintaxe MS-SQL Server™:

```
INSERT cliente (cod_cliente, nome_cliente)
SELECT  cod_vendedor, nome_vendedor, COUNT(*)
FROM   vendedor , pedido
WHERE Vendedor.cod_vendedor = Pedido.cod_vendedor
HAVING COUNT(*) > 50;
```

A diferença de sintaxe se resume somente à existência da cláusula `INTO` no SQL ANSI 92 e cumpre destacar que ORACLE exige a cláusula `INTO`.

C) Atualizando um Registro - UPDATE

A atualização de dados em linhas existentes na tabela permite que :

Especifique-se uma determinada coluna e altere-se seu valor.

Seja indicada uma linha específica ou uma condição de identificação de linhas para que sejam alterados valores de determinadas colunas.

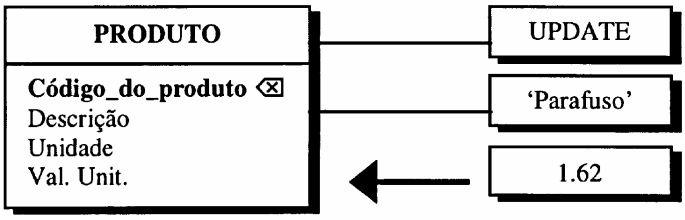
♦ Formato:

UPDATE <nome da tabela>
SET <nome da(s) coluna(s)> = valor
WHERE <condição>;

♦ Problema:

– Alterar o valor unitário do produto 'parafuso' de R$ 1.25 para R$ 1.62.

• Diagrama gráfico:



• Sintaxe:

```
UPDATE produto
        SET val_unit = 1.62
        WHERE descrição = 'Parafuso';
```

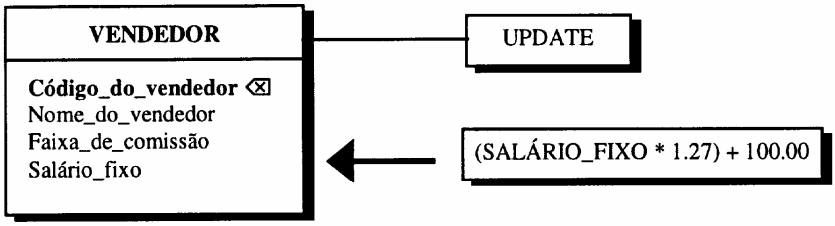
Resultado na tabela:

COD_PRODUTO	DESCRIÇÃO	UNIDADE	VAL_UNIT
108	parafuso	Kg	1.62

♦ Problema:

– Atualizar o salário fixo de todos os vendedores em 27% mais uma bonificação de R$ 100,00.

- Diagrama gráfico:



Sintaxe:

```
UPDATE vendedor
   SET salário_fixo = (salário_fixo * 1.27) + 100.00;
```

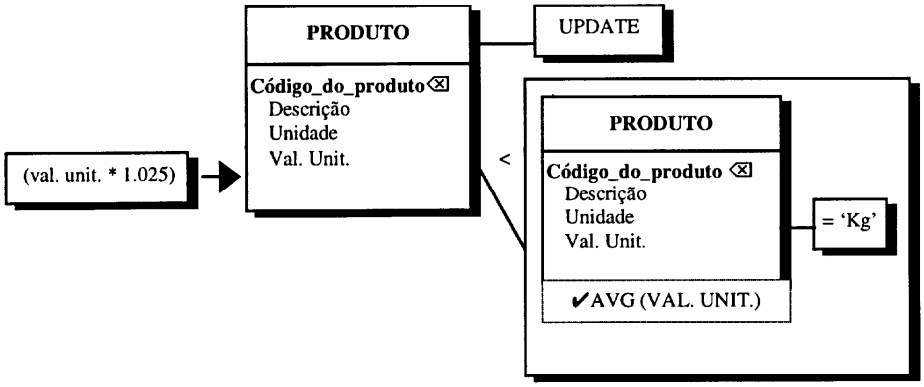
- O resultado desse comando faz com que todos os vendedores tenham o mesmo valor de salário fixo.

- A sintaxe é idêntica para Microsoft® SQL SERVER™.

♦ Problema:

– Acrescentar 2,5% ao preço unitário dos produtos que estejam abaixo da média dos preços, para aqueles comprados a Quilo.

- Diagrama gráfico:



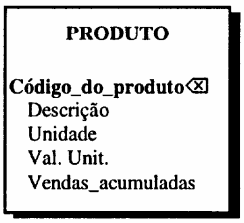
- Sintaxe:

```
UPDATE produto
       SET val_unit = val_unit * 1.025
       WHERE val_unit <
       SELECT AVG(val_unit)
       FROM produto
       WHERE unidade = 'Kg');
```

D) Alterando Registros com dados de outra Tabela

Para explicar esse comando, vamos supor que nossa tabela de produtos possua um campo de vendas acumuladas do ano.

Diagrama Gráfico



- Problema:

Atualizar as vendas acumuladas do ano para cada produto.

```
UPDATE Produto
SET Vendas_acumuladas = 0
UPDATE Produto
SET Vendas_acumuladas = ( SELECT SUM (quantidade)
FROM Item_de_pedido)
```

- Resultado na tabela

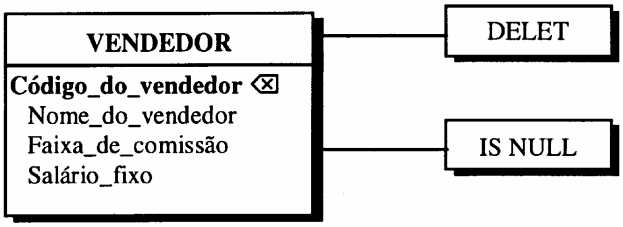
Código do produto	Unidade do produto	Descrição do produto	Valor unitário	Vendas Acumuladas
25	Kg	Queijo	0,97	30
31	BAR	Chocolate	0,87	57
78	L	Vinho	2,00	193
22	M	Linho	0,11	20
30	SAC	Açúcar	0,30	0
53	M	Linha	1,80	82
13	G	Ouro	6,18	36
45	M	Madeira	0,25	8
87	M	Cano	1,97	0
77	M	Papel	1,05	143

D) Apagando Registros da Tabela

- Formato:

```
DELETE FROM <nome da tabela>
    WHERE <condição>;
```

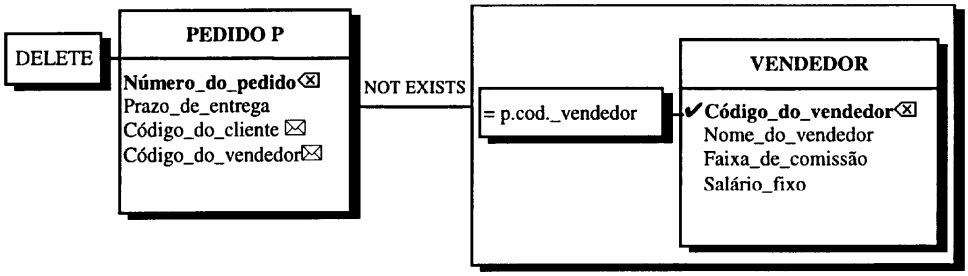
- Problema:
  - Apagar todos os vendedores com faixa de comissão nula.
  - Diagrama gráfico:



  - Sintaxe:

```
DELETE FROM vendedor
       WHERE faixa_comissão IS NULL;
```

- Problema:
  - Apagar todos os registros de pedidos realizados por vendedores fantasmas (operação caça-fantasma).
  - Diagrama gráfico:



  - Sintaxe:

```
DELETE FROM pedido
WHERE NOT EXIST
(SELECT vendedor
FROM vendedor
WHERE cod_vendedor = Pedido.cod_vendedor);
```

D.1) Apagando Registros da Tabela com Base em Dados de Outra Tabela

- Problema:

  - Apagar todos os registros de item de pedidos realizados para produtos que possuam "lh" no nome.

  - Sintaxe ANSI 92:

```
DELETE FROM item_de_pedido
WHERE pedido.cod_produto IN
(SELECT * FROM produto, item_de_pedido
WHERE
produto.cod_produto = item_de_pedido.cod_produto
AND produto.descricao LIKE %lh%)
```

  - Sintaxe Micrososft®-SQL Server™:

```
DELETE FROM item_de_pedido
FROM produto P, item_de_pedido I
WHERE P.cod_produto =  I.cod_produto
AND produto.descricao LIKE %lh%)
```

14.4.5 - Utilizando Views

Uma VIEW é um caminho alternativo para visualizarmos dados derivados de uma ou mais tabelas em um banco de dados. Um usuário pode necessitar ver partes selecionadas de dados com nome, departamento e supervisor, porém não visualizar salário. VIEWS também podem ser utilizadas para informação calculada ou derivada, como preço total de um item de pedido que é calculado por meio da multiplicação de quantidade e preço unitário.

As tabelas criadas em um banco de dados relacional têm existência física dentro do sistema de computação. Muitas vezes é necessário criar tabelas que não ocupem espaço físico, mas que possam ser utilizadas como as tabelas normais. Essas são chamadas de VIEWS (tabelas virtuais).

Como as TABELAS REAIS, as VIEWS devem ser criadas:

- Formato:

```
CREATE VIEW <nome da VIEW>
(<nome da(s) coluna(s)>) AS
```

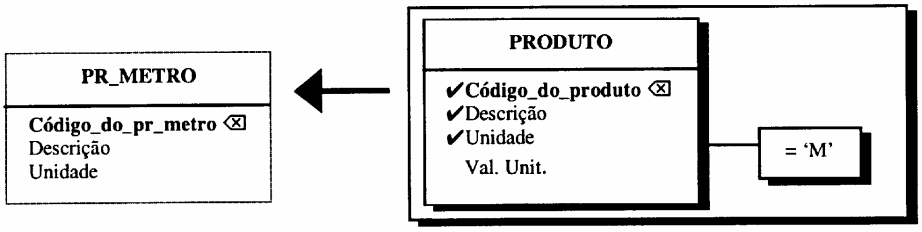
```
SELECT <nome da(s) coluna(s)>
FROM <nome da tabela>
WHERE <condição>
[WITH CHECK OPTION];
```

As VIEWS são utilizadas para se ter uma particular visão de uma tabela, para que não seja necessária a utilização do conjunto como um todo.

Restrições de Views.

Não utilize `SELECT INTO`, `ORDER BY`, `COMPUTE`, `COMPUTE BY` ou `UNION`

- Problema:
  - Criar uma VIEW que contenha só os produtos cuja medida seja metro.
  - Diagrama Gráfico:

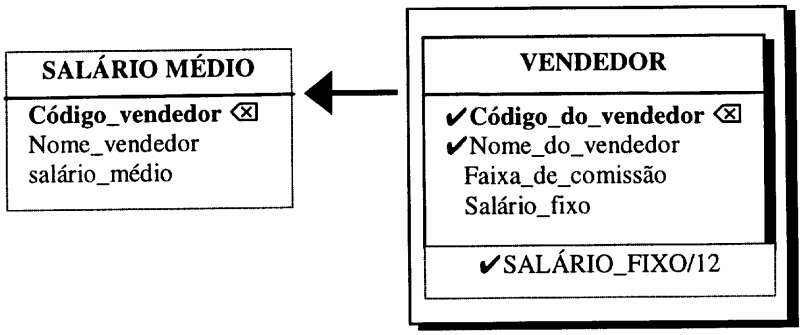


  - Sintaxe:

```
CREATE VIEW PR_metro (cod_PR_metro,
 descrição, unidade) AS
SELECT cod_produto, descrição, unidade
FROM produto
WHERE unidade = 'M';
```

- Problema:
  - Criar uma VIEW contendo o código do vendedor, o seu nome e o salário fixo médio no ano.

- Diagrama gráfico:



- Sintaxe:

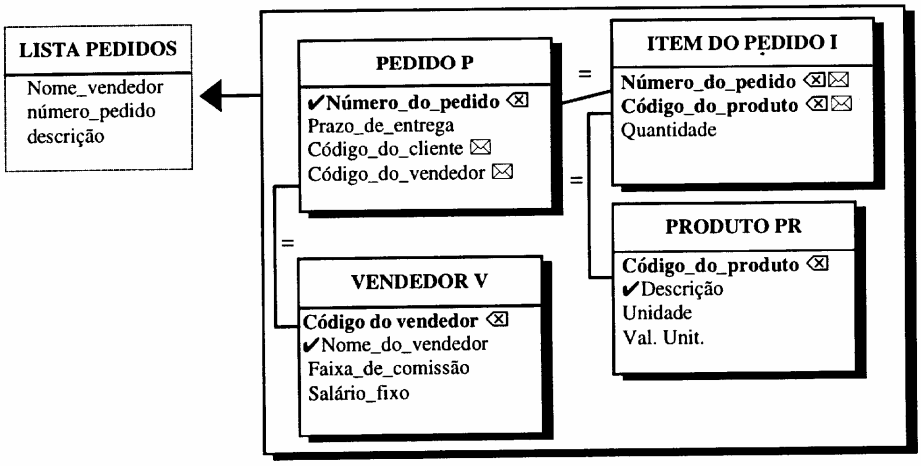
```
CREATE VIEW salário_medio (cod_vendedor,
  nome_vendedor, salário_medio) AS
SELECT cod_vendedor, nome_vendedor, salario_fixo/12
FROM vendedor;
```

14.4.6 - Criando uma View por meio de um Join

♦ Problema:

- Criar uma VIEW contendo os vendedores, seus pedidos efetuados e os respectivos produtos.

- Diagrama gráfico:



Sintaxe:

```
CREATE VIEW lista_pedidos AS
    SELECT nome_vendedor, num_pedido, descrição
    FROM vendedor V, pedido P, item_pedido I, produto PR
    WHERE V.cod_vendedor = P.cod_vendedor and
    P.num_pedido = I.num_pedido and
    I.cod_produto = PR.cod_produto;
```

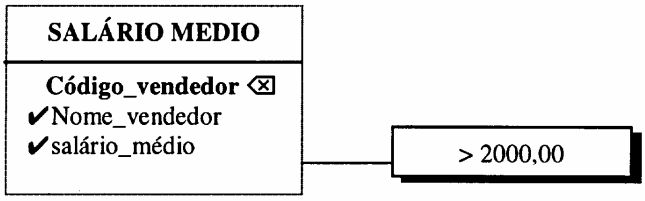
As VIEWS criadas passam a existir no banco de dados como se fossem tabelas reais. As VIEWS são voláteis, desaparecendo no final da sessão de trabalho. Depois de criadas, elas podem ser utilizadas em todas as funções da linguagem SQL (listar, inserir, modificar, apagar, etc.).

14.4.7 - Utilizando uma View

A) Listando

- Problema:
  - Com base na VIEW SALÁRIO_MEDIO, mostrar os vendedores que possuem média salarial superior a R$2.000,00.

- Diagrama gráfico:



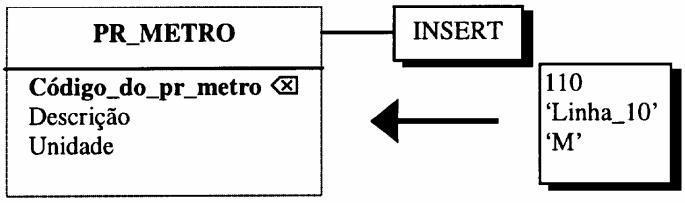
- Sintaxe:

```
SELECT nome_vendedor, salario_medio
       FROM salario_medio
       WHERE salario_medio > 2000.00;
```

B) Inserindo

- Problema:
  - Inserir o registro: 110, Linha_10, M; na VIEW PR_METRO.

- Diagrama gráfico:



- Sintaxe:

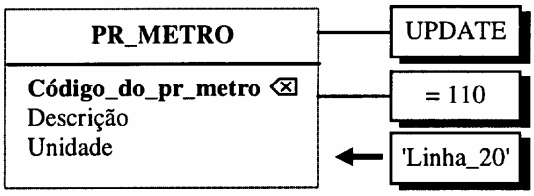
```
INSERT INTO pr_metro
      VALUES (110, 'Linha_10', 'M');
```

C) Modificando

♦ Problema:

– Alterar a descrição de 'Linha_10' para 'Linha_20' no código 110 da VIEW PR_METRO.

- Diagrama gráfico:



- Sintaxe:

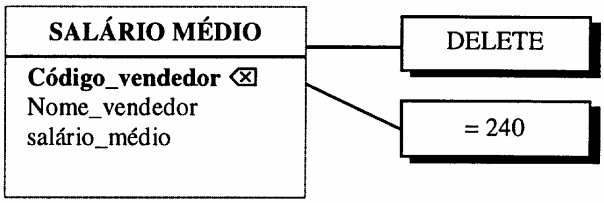
```
UPDATE pr_metro
       SET descrição = 'Linha_20'
       WHERE cod_pr_metro = 110;
```

D) Apagando

♦ Problema:

– Apagar da VIEW salario_medio o registro de código do vendedor igual a 240.

- Diagrama gráfico:



- Sintaxe:

```
DELETE FROM salario_medio
       WHERE cod_vendedor = 240;
```

E) Eliminando uma View

♦ Formato:

```
DROP VIEW <nome da VIEW>;
```

♦ Problema:

– Eliminar a **VIEW** salário_médio;

- Sintaxe:

```
DROP VIEW salário_médio;
```

14.4.8 - Garantindo os Privilégios de Acesso - Grant e Revoke

Muitos sistemas de banco de dados relacionais podem ser acessados por diversos usuários. Cada usuário tem uma determinada necessidade em relação aos dados armazenados. De acordo com o projeto do banco de dados, alguns usuários só podem consultar alguns dados, outros podem atualizar, outros podem inserir, etc. Para que o dado fique protegido do uso indevido de qualquer usuário, a linguagem SQL permite a definição dos privilégios que cada um pode ter em relação às tabelas criadas no banco de dados.

Os privilégios garantem a segurança e a integridade dos dados, bem como a responsabilidade de cada usuário sobre seus dados específicos.

A) O Comando Grant (garantir)

Quando uma tabela/view é criada, o nome do usuário que a criou é anexado, internamente, ao nome da tabela.

Por exemplo: se a tabela produto foi criada pelo usuário Felipe, então internamente ela será conhecida como Felipe.produto.

O criador da tabela/view tem total privilégio sobre a tabela criada, podendo disponibilizar qualquer privilégio para outros usuários pelo comando GRANT:

- Sintaxe:

```
GRANT {ALL |Lista de privilégios}
ON {nome da tabela/view [lista de colunas]}
TO {PUBLIB |lista de usuários}
[WITH GRANT OPTION]
```

A palavra **ALL** falta um verbo quando qualquer previlégio é aplicável ao objeto, ou então especificamos qual o previlégio que está sendo dado (SELECT, UPDATE, etc.).

A cláusula **ON** especifica a tabela ou view e suas colunas para as quais está sendo dado o previlégio.

**** - Importante: Somente pode ser utilizada uma tabela ou view por comando.**

As diferentes versões de SQL dos SGBDs relacionais incluem diferentes conjuntos de previlégios. SELECT, UPDATE, DELETE e INSERT estão sempre presentes em todos esses conjuntos, indiferentemente do SGBD.

Os previlégios podem ser especificados para algumas colunas , porém devem ser todas da mesma tabela. Se não for especificada nenhuma coluna, os previlégios valem para todas.

A cláusula opcional WITH GRANT OPTION permite que quando se dá o previlégio a um usuário, ele passe esse previlégio para os outros usuários.

Lista de opções de privilégios:

Select ➪ pode executar uma consulta sobre a tabela

Insert	➪ pode executar uma inserção sobre a tabela
Delete	➪ pode apagar registros da tabela
Update	➪ pode modificar registros na tabela
All privileges/all	➪ pode executar qualquer operação sobre a tabela
<usuário>	➪ nome do usuário que vai receber os privilégios. Tem de ser um nome cadastrado dentro do ambiente.
PUBLIC	➪ Concede os privilégios especificados a todos os usuários do ambiente.

Exemplos:

GRANT SELECT ON Produto TO Maurício; - permite só consultas ao usuário Maurício sobre a tabela produto;

GRANT Select, insert, update on pedido to tele_mark; - concede ao usuário tele_mark (entrada de pedidos), os privilégios de seleção, inserção e alteração sobre a tabela PEDIDO;

GRANT All privileges on cliente to public; - permite todos os privilégios a todos os usuários sobre a tabela cliente;

GRANT Select on cliente to Felipe, Mauricio; - concede aos usuários Maurício e Felipe, o privilégio de seleção sobre a tabela CLIENTE.

- Problema:
  - Disponibilizar para seleção, a view salário_medio a todos os usuários.
  - Sintaxe:

```
GRANT Select ON salario_medio TO public;
```

Podemos, por meio do comando GRANT, disponibilizar acessos só a alguns campos da tabela/view. Vamos a um exemplo:

- Problema:

  - Disponibilizar para seleção, só os campos código de vendedor e nome do vendedor da tabela vendedor a todos os usuários.

  - Sintaxe:

```
GRANT Select (cod_vendedor, nome_vendedor)
ON vendedor
TO public;
```

Podemos passar nossa concessão de privilégios a outros usuários por meio da cláusula WITH GRANT OPTION, como explicamos anteriormente

- Problema:

  - Conceder ao usuário FELIPE o poder de permitir a concessão de todos os privilégios a outros usuários sobre a tabela PEDIDO.

  - Sintaxe:

```
GRANT ALL
ON pedido
TO Felipe
WITH GRANT OPTION
```

B) O Comando Revoke (revogação)

Da mesma forma que o criador da tabela pode garantir (GRANT) os privilégios de acesso aos outros usuários, ele pode revogar esses privilégios por meio do comando REVOKE:

- Formato:

```
REVOKE [ lista de privilégios ]
ON [nome da tabela/view]
FROM [lista de usuários];
```

- Problema:

  - Retirar o privilégio de seleção sobre a tabela produto do usuário Maurício.

  - Sintaxe:

```
REVOKE select
ON produto
FROM Mauricio;
```

- Problema:
  - Revogar todos os privilégios concedidos a todos os usuários sobre a tabela cliente.
  - Sintaxe:

```
REVOKE select
ON cliente
FROM public;
```

- Problema:
  - Retirar os privilégios de atualização e inserção concedidos ao usuário tele_mark sobre a tabela pedido.
  - Sintaxe:

```
REVOKE insert, update
ON pedido
FROM tele_mark
```

14.4.9 - Trabalhando com Índices

Índice é uma estrutura que permite rápido acesso às linhas de uma tabela, com base nos valores de uma ou mais colunas. O índice é simplesmente uma outra tabela no banco de dados, na qual estão armazenados valores e ponteiros, arrumados de forma ascendente ou descendente. O SGBD utiliza os índices para pesquisar rapidamente um determinado valor dentro do banco de dados. Por intermédio dos ponteiros se localiza em qual linha o valor desejado está armazenado.

As tabelas de índices são utilizadas internamente pelo SGBD, ficando totalmente transparente ao usuário sua utilização.

Um Check List para Criação de Índices

Quando vamos criar índices em colunas, devemos considerar :

Criar índice sobre coisas que vamos pesquisar com freqüência.

Indexe suas chaves estrangeiras quando precisar de "joins" mais eficientes e performantes.

As colunas que são regularmente utilizadas em joins devem ser indexadas porque o sistema pode performar esses joins de forma muito mais rápida, e tempo de resposta é algo vital no desenvolvimento de projetos físicos de bancos de dados.

Crie índice sempre em colunas que são pesquisas por um intervalo de valores.

E por fim, crie índice sempre em colunas que são utilizadas em cláusulas WHERE.

Quando não criar índices:

Se a expectativa de retorno de linhas de uma consulta for muita alta índices não devem ser utilizados. Por exemplo: se uma coluna tem somente dois valores: Masculino e Feminino, uma consulta a essa coluna sempre vai retornar uma alta quantidade de linhas. Como é um índice não agrupado, poderá ocupar muito espaço e pode não ser utilizado para consultas.

O conceito de índice Cluster ou agrupado é quando os valores constantes na coluna podem ser agrupados em pequenos conjuntos de pesquisa, facilitando a pesquisa na área de índices da tabela. Quando o número de valores diferentes que pode aparecer na coluna é pequeno, o índice é não agrupado. Mas nosso objetivo aqui é somente tentar dar um conceito; não vamos nos aprofundar demais, pois estes aspectos fazem parte de um estudo todo específico para Administradores de Bancos de Dados.

Índices agrupados ou "clustered" são dispostos na mesma ordem física das linhas da tabela, e índices Não Agrupados ou nonclustered são organizados pela ordem lógica das linhas da tabela.

Vamos ver como se criam índices.

A) Criando Índices

Cada índice é aplicado a uma tabela, especificando uma ou mais colunas dessa tabela.

Índices são criados por meio do comando CREATE INDEX, que cria um índice sobre colunas de um tabela.

Formato:

```
CREATE [UNIQUE] INDEX <nome do índice>
ON <nome da tabela> (<coluna(s)>);
```

A cláusula UNIQUE é opcional e define que para aquela coluna não existirão valores duplicados, ou seja, todos os dados armazenados na coluna serão únicos.

A junção do índice unique e da especificação NOT NULL para uma coluna define a chave primária da tabela quanto ao aspecto lógico, pois uma chave primária, como vimos neste livro, não pode ser NULA.

A criação dos índices depende muito do projeto do banco de dados e das necessidades de pesquisa formuladas pelos usuários do banco de dados. Os índices estão muito ligados às necessidades de velocidade na recuperação da informação, e na execução rápida de uma operação de JOIN.

Para cada SGBD existem cláusulas específicas operacionais que devem ser usadas, mas neste caso vamos apresentar a sintaxe padrão geral do SQL ANSI.

Exemplos:

```
CREATE INDEX nome_pro
ON produto (descrição);
```

- Cria a tabela de índices chamada nome_pro baseada no campo descrição da tabela produto;

```
CREATE INDEX ped_pro
      ON item_produto (num_pedido, cod_produto);
```

- Cria a tabela de índices ped_pro baseada na concatenação dos campos num_pedido e cod_produto da tabela item_pedid

  É importante considerar que praticamente todas as sintaxes em se tratando de SGBDs relacionais exige que se identifique o database proprietário da tabela, principalmente em Microsoft® SQL Server™.

```
CREATE UNIQUE INDEX clientex
ON [nome do database]cliente (cod_cliente);
```

- Cria o índice único para a tabela cliente baseada no código do cliente, não podendo haver duplicidade de informação armazenada.

C) Eliminando Índices

Da mesma forma que um índice é criado, ele pode ser eliminado, dependendo das necessidades do projeto do banco de dados.

- Formato:

```
DROP index <nome do índice>;
```

- Exemplos:

```
DROP index nome_pro ;
DROP index ped_pro ;
```

14.4.10 - Tópicos Avançados de SQL

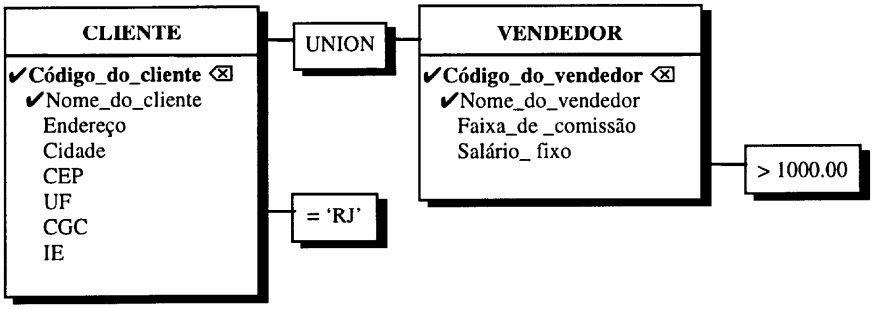
A) Combinando Resultados de Pesquisas (UNION)

Eventualmente, é necessário combinar os resultados de duas ou mais consultas feitas sobre tabelas. Para realizar esta operação, a linguagem SQL oferece o operador UNION e uma listagem contendo os resultados das consultas combinadas.

- Problema:

  - Listar os nomes e códigos dos vendedores que têm salário fixo maior que R$ 1.000,00 e clientes que residem no Rio de Janeiro.

  - Diagrama gráfico:



- Sintaxe:

```
SELECT codigo = cod_cliente, nome = nome_cliente
FROM cliente
WHERE UF = 'RJ'
UNION
SELECT cod_vendedor, nome_vendedor
FROM vendedor
WHERE salario_fixo > 1000.00
```

** Observa-se que as duas listas de colunas dos SELECT contêm o mesmo número de itens (duas colunas) e os tipos de dados são compatíveis.

C) Realizando um JOIN entre uma Tabela e ela mesma

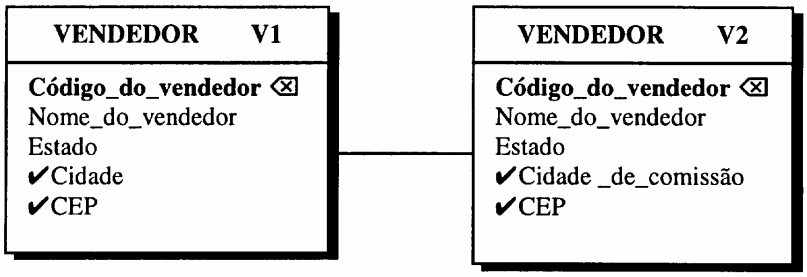
Às vezes, é necessário realizar pesquisas complexas dentro da mesma tabela. Porque esse join envolve uma tabela em junção com ela mesma, é necessário que se providenciem dois aliases para a tabela.

Este tipo de enfoque é melhor explicado com um exemplo:

♦ Problema:

- Determinar quais vendedores cujo estado é Califórnia, residem na mesma cidade Oakland e vivem no mesmo ZIP Code (Cep americano):

- Diagrama Gráfico:



- Sintaxe ANSI

```
SELECT nome_menor = V1.nome_vendedor,
salario_menor = V1.salario_fixo,
nome_maior = V2.nome_vendedor,
salario_maior = V2.salario_fixo,
FROM vendedor V1 CROSS JOIN vendedor V2
WHERE V1.salario_fixo < V2.salario_fixo ORDER BY 1;
```

- Sintaxe Microsoft® SQL Server ™ :

```
SELECT nome_menor = V1.nome_vendedor,
salario_menor = V1.salario_fixo,
nome_maior = V2.nome_vendedor,
salario_maior = V2.salario_fixo,
FROM vendedor V1  vendedor V2
WHERE V1.salario_fixo < V2.salario_fixo ORDER BY 1;
```

- Resultado:

NOME MENOR	SALÁRIO MENOR	NOME MAIOR	SALÁRIO MAIOR
Carlos	2490,00	João	2780,00
Carlos	2490,00	Antônio	9500,00
Carlos	2490,00	Felipe	4600,00
Carlos	2490,00	João	2650,00
Carlos	2490,00	Maurício	2930,00
Felipe	4600,00	Antônio	9500,00
João	2780,00	Antônio	9500,00
João	2780,00	Felipe	4600,00
João	2780,00	Maurício	2930,00
João	2650,00	João	2780,00
João	2650,00	Antônio	9500,00
João	2650,00	Felipe	4600,00
João	2650,00	Maurício	2930,00
⋮	⋮	⋮	⋮

A seguir, será apresentado o uso da Linguagem SQL para a construção das bases de dados dos estudos de casos apresentados no Capítulo 11. Serão utilizados como SGBD, o ORACLE Versão 7®e Microsoft® SQL Server 7.0 ™.

14.5 - Estudo de Caso 1

Para que o nosso leitor possa comparar as sintaxes dos comandos SQL, nesta revisão estamos apresentando os scripts gerados para os estudos de caso deste livro.

Utilizamos para este fim o software ERWin 3.5 para os servidores SQL Oracle e Microsoft SQL Server.

Desta forma, o leitor terá à sua disposição duas formas de criação para os bancos de dados físicos dos estudos de caso. Bom proveito, amigo!

Microsoft®-SQL Server.™ 7.0

```
CREATE TABLE ator (
        cdator                  int NOT NULL,
        noator                  char(30) NULL,
        idade                   int NULL,
        nacionalidade           char(20) NULL
)
go

CREATE TABLE genero (
        cd_genero               int NOT NULL,
        nome_genero             char(30) NULL
)
go

CREATE TABLE filme (
        cdfilme                 int NOT NULL,
        cddiretor               int NULL,
        titulo_original         char(30) NULL,
        titulo_brasil           char(30) NULL,
        impropriedade           int NOT NULL,
        duracao                 int NULL,
        origem                  char(15) NULL,
        cd_genero               int NULL,
        FOREIGN KEY (cddiretor)
                                 REFERENCES ator,
        FOREIGN KEY (cd_genero)
                                 REFERENCES genero
)
go

CREATE INDEX XIF19filme ON filme
(
        cd_genero
)
go

CREATE INDEX XIF24filme ON filme
(
        cddiretor
)
go

CREATE TABLE elenco (
        cdator                  int NOT NULL,
        cdfilme                 int NOT NULL,
        numero_personagem       int NOT NULL,
        tempo_de_atuacao        datetime NULL,
```

```
        nome_personagem       char(30) NULL,
        FOREIGN KEY (cdfilme)
                               REFERENCES filme,
        FOREIGN KEY (cdator)
                               REFERENCES ator
)
go

CREATE INDEX XIF22elenco ON elenco
(
        cdator
)
go

CREATE INDEX XIF23elenco ON elenco
(
        cdfilme
)
go
CREATE TABLE cinema (
        cdcinema              int NOT NULL,
        nome_fantasia         char(30) NULL,
        Logradouro            char(30) NULL,
        Bairro                char(30) NULL,
        Municipio             char(30) NULL,
        Estado                char(20) NULL,
        CEP                   char(9) NULL,
        Capacidade            int NULL
)
go


CREATE TABLE passa (
        cdcinema              int NOT NULL,
        cdfilme               int NOT NULL,
        data_inicio_exibicao datetime NULL,
        data_fim_exibicao    datetime NULL,
        FOREIGN KEY (cdfilme)
                               REFERENCES filme,
        FOREIGN KEY (cdcinema)
                               REFERENCES cinema
)
go

CREATE INDEX XIF17passa ON passa
(
        cdcinema
)
go

CREATE INDEX XIF18passa ON passa
(
        cdfilme
```

```
)
go

CREATE TABLE sessao (
        cdfilme                 int NULL,
        cdcinema                int NULL,
        data_sessao             datetime NULL,
        publico                 int NULL,
        horario_sessao          datetime NULL,
        FOREIGN KEY (cdcinema, cdfilme)
                                 REFERENCES passa
)
go

CREATE INDEX XIF20sessao ON sessao
(
        cdfilme,
        cdcinema
)
go
```

Oracle 8.xx

```
CREATE TABLE ator (
        cdator                  INTEGER NOT NULL,
        noator                  CHAR(30) NULL,
        idade                   INTEGER NULL,
        nacionalidade           CHAR(20) NULL,
        PRIMARY KEY (cdator)
);
CREATE TABLE genero (
        cd_genero               INTEGER NOT NULL,
        nome_genero             CHAR(30) NULL,
        PRIMARY KEY (cd_genero)
);

CREATE TABLE filme (
        cdfilme                 INTEGER NOT NULL,
        cddiretor               INTEGER NULL,
        titulo_original         CHAR(30) NULL,
        titulo_brasil           CHAR(30) NULL,
        impropriedade           INTEGER NOT NULL,
        duracao                 INTEGER NULL,
        origem                  CHAR(15) NULL,
        cd_genero               INTEGER NULL,
        PRIMARY KEY (cdfilme),
        FOREIGN KEY (cddiretor)
                                 REFERENCES ator,
        FOREIGN KEY (cd_genero)
                                 REFERENCES genero
```

```
);

CREATE INDEX XIF19filme ON filme
(
       cd_genero
);

CREATE INDEX XIF24filme ON filme
(
       cddiretor
);


CREATE TABLE elenco (
       cdator                INTEGER NOT NULL,
       cdfilme               INTEGER NOT NULL,
       numero_personagem     INTEGER NOT NULL,
       tempo_de_atuacao      DATE NULL,
       nome_personagem       CHAR(30) NULL,
       PRIMARY KEY (cdator, cdfilme, numero_personagem),
       FOREIGN KEY (cdfilme)
                              REFERENCES filme,
       FOREIGN KEY (cdator)
                              REFERENCES ator
);

CREATE INDEX XIF22elenco ON elenco
(
       cdator
);

CREATE INDEX XIF23elenco ON elenco
(
       cdfilme
);

CREATE TABLE cinema (
       cdcinema              INTEGER NOT NULL,
       nome_fantasia         CHAR(30) NULL,
       Logradouro            CHAR(30) NULL,
       Bairro                CHAR(30) NULL,
       Municipio             CHAR(30) NULL,
       Estado                CHAR(20) NULL,
       CEP                   CHAR(9) NULL,
       Capacidade            INTEGER NULL,
       PRIMARY KEY (cdcinema)
);


CREATE TABLE passa (
       cdcinema              INTEGER NOT NULL,
       cdfilme               INTEGER NOT NULL,
       data_inicio_exibicao DATE NULL,
       data_fim_exibicao     DATE NULL,
```

```
        PRIMARY KEY (cdcinema, cdfilme),
        FOREIGN KEY (cdfilme)
                                REFERENCES filme,
        FOREIGN KEY (cdcinema)
                                REFERENCES cinema
);

CREATE INDEX XIF17passa ON passa
(
        cdcinema
);

CREATE INDEX XIF18passa ON passa
(
        cdfilme
);


CREATE TABLE sessao (
        cdfilme                 INTEGER NULL,
        cdcinema                INTEGER NULL,
        data_sessao             DATE NULL,
        publico                 INTEGER NULL,
        horario_sessao          DATE NULL,
        FOREIGN KEY (cdcinema, cdfilme)
                                REFERENCES passa
);

CREATE INDEX XIF20sessao ON sessao
(
        cdfilme,
        cdcinema
);
```

14.6 - Estudo de Caso 2

Oracle 8.xx

```
CREATE TABLE Departamento (
        Codigo_do_Departamento INTEGER NOT NULL,
        Nome_Depto             CHAR(20) NULL,
        PRIMARY KEY (Codigo_do_Departamento)
);

CREATE UNIQUE INDEX XPKDepartamento ON Departamento
(
        Codigo_do_Departamento
);
```

```
CREATE TABLE Disciplina (
       Codigo_da_disciplina INTEGER NOT NULL,
       Codigo_do_Departamento INTEGER NOT NULL,
       Nome_Disciplina      CHAR(20) NULL,
       Descrição_Curricular CHAR(40) NULL,
       PRIMARY         KEY         (Codigo_da_disciplina,
Codigo_do_Departamento),
       FOREIGN KEY (Codigo_do_Departamento)
                              REFERENCES Departamento
);

CREATE UNIQUE INDEX XPKDisciplina ON Disciplina
(
       Codigo_da_disciplina,
       Codigo_do_Departamento
);

CREATE INDEX XIF26Disciplina ON Disciplina
(
       Codigo_do_Departamento
);


CREATE TABLE Pre_requisito (
       Codigo_do_Departamento INTEGER NOT NULL,
       Codigo_da_disciplina INTEGER NOT NULL,
       Codigo_Prerequisito  CHAR(18) NULL,
       Norma_CFE            CHAR(18) NULL,
       PRIMARY        KEY        (Codigo_do_Departamento,
Codigo_da_disciplina),
       FOREIGN         KEY         (Codigo_da_disciplina,
Codigo_do_Departamento)
                              REFERENCES Disciplina,
       FOREIGN         KEY         (Codigo_da_disciplina,
Codigo_do_Departamento)
                              REFERENCES Disciplina
);

CREATE UNIQUE INDEX XPKPre_requisito ON Pre_requisito
(
       Codigo_do_Departamento,
       Codigo_da_disciplina
);


CREATE TABLE Professor (
       Codigo_do_Departamento INTEGER NOT NULL,
       Codigo_do_professor  INTEGER NOT NULL,
       Nome_Professor       CHAR(30) NULL,
       Inscrição_CFE        INTEGER NULL,
```

```
        PRIMARY         KEY         (Codigo_do_Departamento,
Codigo_do_professor),
        FOREIGN KEY (Codigo_do_Departamento)
                                REFERENCES Departamento
);

CREATE UNIQUE INDEX XPKProfessor ON Professor
(
        Codigo_do_Departamento,
        Codigo_do_professor
);

CREATE INDEX XIF11Professor ON Professor
(
        Codigo_do_Departamento
);


CREATE TABLE Habilitacao (
        Codigo_do_Departamento INTEGER NOT NULL,
        Codigo_do_professor  INTEGER NOT NULL,
        Codigo_da_disciplina INTEGER NOT NULL,
        Data_Habilitacao     DATE NULL,
        PRIMARY         KEY         (Codigo_do_Departamento,
Codigo_do_professor,
                Codigo_da_disciplina),
        FOREIGN         KEY         (Codigo_do_Departamento,
Codigo_do_professor)
                                REFERENCES Professor,
        FOREIGN          KEY          (Codigo_da_disciplina,
Codigo_do_Departamento)
                                REFERENCES Disciplina
);

CREATE UNIQUE INDEX XPKHabilitacao ON Habilitacao
(
        Codigo_do_Departamento,
        Codigo_do_professor,
        Codigo_da_disciplina
);

CREATE INDEX XIF13Habilitacao ON Habilitacao
(
        Codigo_da_disciplina,
        Codigo_do_Departamento
);

CREATE INDEX XIF14Habilitacao ON Habilitacao
(
        Codigo_do_Departamento,
```

```
        Codigo_do_professor
);


CREATE TABLE Curso (
        Codigo_do_Departamento INTEGER NOT NULL,
        Codigo_Curso          INTEGER NOT NULL,
        Nome_Curso            CHAR(20) NULL,
        PRIMARY         KEY         (Codigo_do_Departamento,
Codigo_Curso),
        FOREIGN KEY (Codigo_do_Departamento)
                               REFERENCES Departamento
);
CREATE UNIQUE INDEX XPKCurso ON Curso
(
        Codigo_do_Departamento,
        Codigo_Curso
);
CREATE INDEX XIF18Curso ON Curso
(
        Codigo_do_Departamento
);


CREATE TABLE Possui (
        Codigo_do_Departamento INTEGER NOT NULL,
        Codigo_Curso          INTEGER NOT NULL,
        Codigo_da_disciplina INTEGER NOT NULL,
        Norma_CFE             CHAR(18) NULL,
        PRIMARY KEY (Codigo_do_Departamento, Codigo_Curso,
               Codigo_da_disciplina),
        FOREIGN          KEY          (Codigo_da_disciplina,
Codigo_do_Departamento)
                               REFERENCES Disciplina,
        FOREIGN KEY (Codigo_do_Departamento, Codigo_Curso)
                               REFERENCES Curso
);

CREATE UNIQUE INDEX XPKPossui ON Possui
(
        Codigo_do_Departamento,
        Codigo_Curso,
        Codigo_da_disciplina
);

CREATE INDEX XIF10Possui ON Possui
(
        Codigo_da_disciplina,
        Codigo_do_Departamento
);
```

```
CREATE INDEX XIF9Possui ON Possui
(
       Codigo_do_Departamento,
       Codigo_Curso
);


CREATE TABLE Aluno (
       Numero_matricula     INTEGER NOT NULL,
       Codigo_do_Departamento INTEGER NOT NULL,
       Codigo_Curso         INTEGER NOT NULL,
       Nome                 CHAR(20) NOT NULL,
       Endereco             CHAR(30) NULL,
       Bairro               CHAR(20) NULL,
       Cidade               CHAR(20) NULL,
       Estado               CHAR(20) NULL,
       CEP                  INTEGER NOT NULL,
       PRIMARY            KEY            (Numero_matricula,
Codigo_do_Departamento,
              Codigo_Curso),
       FOREIGN KEY (Codigo_do_Departamento, Codigo_Curso)
                              REFERENCES Curso
);

CREATE UNIQUE INDEX XPKAluno ON Aluno
(
       Numero_matricula,
       Codigo_do_Departamento,
       Codigo_Curso
);
CREATE INDEX XIF22Aluno ON Aluno
(
       Codigo_do_Departamento,
       Codigo_Curso
);

CREATE TABLE CURSA (
       Codigo_do_Departamento INTEGER NOT NULL,
       Codigo_da_disciplina INTEGER NOT NULL,
       Numero_matricula     INTEGER NOT NULL,
       Codigo_Curso         INTEGER NOT NULL,
       nota_final           DECIMAL(2) NULL,
       PRIMARY        KEY         (Codigo_do_Departamento,
Codigo_da_disciplina,
              Numero_matricula, Codigo_Curso),
       FOREIGN          KEY          (Codigo_da_disciplina,
Codigo_do_Departamento)
                              REFERENCES Disciplina,
       FOREIGN            KEY            (Numero_matricula,
Codigo_do_Departamento,
```

```
                Codigo_Curso)
                                REFERENCES Aluno
);
CREATE UNIQUE INDEX XPKCURSA ON CURSA
(
        Codigo_do_Departamento,
        Codigo_da_disciplina,
        Numero_matricula,
        Codigo_Curso
);
CREATE INDEX XIF3CURSA ON CURSA
(
        Numero_matricula,
        Codigo_do_Departamento,
        Codigo_Curso
);
CREATE INDEX XIF4CURSA ON CURSA
(
        Codigo_da_disciplina,
        Codigo_do_Departamento
);

CREATE TABLE REALIZOU (
        Codigo_do_Departamento INTEGER NOT NULL,
        Numero_matricula     INTEGER NOT NULL,
        Codigo_da_disciplina INTEGER NOT NULL,
        Codigo_Curso         INTEGER NOT NULL,
        media                DECIMAL(1) NOT NULL,
        periodo              INTEGER NOT NULL,
        PRIMARY        KEY        (Codigo_do_Departamento,
Numero_matricula,
                Codigo_da_disciplina, Codigo_Curso),
        FOREIGN         KEY         (Codigo_da_disciplina,
Codigo_do_Departamento)
                                REFERENCES Disciplina,
        FOREIGN           KEY           (Numero_matricula,
Codigo_do_Departamento,
                Codigo_Curso)
                                REFERENCES Aluno
);

CREATE UNIQUE INDEX XPKREALIZOU ON REALIZOU
(
        Codigo_do_Departamento,
        Numero_matricula,
        Codigo_da_disciplina,
        Codigo_Curso
);
```

```
CREATE INDEX XIF20REALIZOU ON REALIZOU
(
       Codigo_do_Departamento,
       Numero_matricula,
       Codigo_Curso
);

CREATE INDEX XIF21REALIZOU ON REALIZOU
(
       Codigo_do_Departamento,
       Codigo_da_disciplina
);


CREATE TABLE Regencia (
       Codigo_do_Departamento INTEGER NOT NULL,
       Codigo_do_professor  INTEGER NOT NULL,
       Codigo_da_disciplina INTEGER NOT NULL,
       nro_periodo          CHAR(18) NULL,
       txavaliacao          CHAR(10) NULL,
       PRIMARY        KEY        (Codigo_do_Departamento,
Codigo_do_professor,
              Codigo_da_disciplina),
       FOREIGN         KEY         (Codigo_da_disciplina,
Codigo_do_Departamento)
                              REFERENCES Disciplina,
       FOREIGN        KEY        (Codigo_do_Departamento,
Codigo_do_professor)
                              REFERENCES Professor
);

CREATE UNIQUE INDEX XPKRegencia ON Regencia
(
       Codigo_do_Departamento,
       Codigo_do_professor,
       Codigo_da_disciplina
);

CREATE INDEX XIF24Regencia ON Regencia
(
       Codigo_do_Departamento,
       Codigo_do_professor
);

CREATE INDEX XIF25Regencia ON Regencia
(
       Codigo_do_Departamento,
       Codigo_da_disciplina
);
```

Microsoft®-SQL Server.™ 7.0

```
CREATE TABLE Departamento (
       Codigo_do_Departamento int NOT NULL,
       Nome_Depto             char(20) NULL,
       PRIMARY KEY (Codigo_do_Departamento)
)
go


CREATE TABLE Disciplina (
       Codigo_da_disciplina int NOT NULL,
       Codigo_do_Departamento int NOT NULL,
       Nome_Disciplina      char(20) NULL,
       Descricao_Curricular char(40) NULL,
       PRIMARY          KEY          (Codigo_da_disciplina,
Codigo_do_Departamento),
       FOREIGN KEY (Codigo_do_Departamento)
                              REFERENCES Departamento
)
go


CREATE TABLE Pre_requisito (
       Codigo_do_Departamento int NOT NULL,
       Codigo_da_disciplina int NOT NULL,
       Codigo_Prerequisito  char(18) NULL,
       Norma_CFE            char(18) NULL,
       PRIMARY         KEY         (Codigo_do_Departamento,
Codigo_da_disciplina),
       FOREIGN          KEY          (Codigo_da_disciplina,
Codigo_do_Departamento)
                              REFERENCES Disciplina,
       FOREIGN          KEY          (Codigo_da_disciplina,
Codigo_do_Departamento)
                              REFERENCES Disciplina
)
go


CREATE TABLE Professor (
       Codigo_do_Departamento int NOT NULL,
       Codigo_do_professor  int NOT NULL,
       Nome_Professor       char(30) NULL,
       Inscricao_CFE        int NULL,
       PRIMARY         KEY         (Codigo_do_Departamento,
Codigo_do_professor),
       FOREIGN KEY (Codigo_do_Departamento)
                              REFERENCES Departamento
)
```

```
go

CREATE TABLE Habilitacao (
       Codigo_do_Departamento int NOT NULL,
       Codigo_do_professor  int NOT NULL,
       Codigo_da_disciplina int NOT NULL,
       Data_Habilitacao     datetime NULL,
       PRIMARY         KEY         (Codigo_do_Departamento,
Codigo_do_professor,
              Codigo_da_disciplina),
       FOREIGN         KEY         (Codigo_do_Departamento,
Codigo_do_professor)
                              REFERENCES Professor,
       FOREIGN          KEY          (Codigo_da_disciplina,
Codigo_do_Departamento)
                              REFERENCES Disciplina
)
go

CREATE TABLE Curso (
       Codigo_do_Departamento int NOT NULL,
       Codigo_Curso         int NOT NULL,
       Nome_Curso           char(20) NULL,
       PRIMARY KEY (Codigo_do_Departamento, Codigo_Curso),
       FOREIGN KEY (Codigo_do_Departamento)
                              REFERENCES Departamento
)
go

CREATE TABLE Possui (
       Codigo_do_Departamento int NOT NULL,
       Codigo_Curso         int NOT NULL,
       Codigo_da_disciplina int NOT NULL,
       Norma_CFE            char(18) NULL,
       PRIMARY KEY (Codigo_do_Departamento, Codigo_Curso,
              Codigo_da_disciplina),
       FOREIGN          KEY          (Codigo_da_disciplina,
Codigo_do_Departamento)
                              REFERENCES Disciplina,
       FOREIGN KEY (Codigo_do_Departamento, Codigo_Curso)
                              REFERENCES Curso
)
go
```

```
CREATE TABLE Aluno (
       Numero_matricula     int NOT NULL,
       Codigo_do_Departamento int NOT NULL,
       Codigo_Curso         int NOT NULL,
       Nome                 char(20) NOT NULL,
       Endereco             char(30) NULL,
       Bairro               char(20) NULL,
       Cidade               char(20) NULL,
       Estado               char(20) NULL,
       CEP                  int NOT NULL,
       PRIMARY            KEY            (Numero_matricula,
Codigo_do_Departamento,
              Codigo_Curso),
       FOREIGN KEY (Codigo_do_Departamento, Codigo_Curso)
                              REFERENCES Curso
)


go
CREATE TABLE CURSA (
       Codigo_do_Departamento int NOT NULL,
       Codigo_da_disciplina int NOT NULL,
       Numero_matricula     int NOT NULL,
       Codigo_Curso         int NOT NULL,
       nota_final           decimal(2) NULL,
       PRIMARY         KEY         (Codigo_do_Departamento,
Codigo_da_disciplina,
              Numero_matricula, Codigo_Curso),
       FOREIGN          KEY          (Codigo_da_disciplina,
Codigo_do_Departamento)
                              REFERENCES Disciplina,
       FOREIGN            KEY            (Numero_matricula,
Codigo_do_Departamento,
              Codigo_Curso)
                              REFERENCES Aluno
)
go


CREATE TABLE REALIZOU (
       Codigo_do_Departamento int NOT NULL,
       Numero_matricula     int NOT NULL,
       Codigo_da_disciplina int NOT NULL,
       Codigo_Curso         int NOT NULL,
       media                decimal(1) NOT NULL,
       periodo              int NOT NULL,
       PRIMARY         KEY         (Codigo_do_Departamento,
Numero_matricula,
              Codigo_da_disciplina, Codigo_Curso),
```

```
        FOREIGN          KEY          (Codigo_da_disciplina,
Codigo_do_Departamento)
                                 REFERENCES Disciplina,
        FOREIGN            KEY            (Numero_matricula,
Codigo_do_Departamento,
                Codigo_Curso)
                                 REFERENCES Aluno
)
go


CREATE TABLE Regencia (
        Codigo_do_Departamento int NOT NULL,
        Codigo_do_professor  int NOT NULL,
        Codigo_da_disciplina int NOT NULL,
        nro_periodo          char(18) NULL,
        txavaliacao          char(10) NULL,
        PRIMARY         KEY         (Codigo_do_Departamento,
Codigo_do_professor,
                Codigo_da_disciplina),
        FOREIGN          KEY          (Codigo_da_disciplina,
Codigo_do_Departamento)
                                 REFERENCES Disciplina,
        FOREIGN         KEY         (Codigo_do_Departamento,
Codigo_do_professor)
                                 REFERENCES Professor
)
go
```

14.7 - Estudo de Caso 3

Microsoft® SQL Server™ 7.0

```
CREATE TABLE VIRUS (
        COD_DO_VIRUS          char(18) NOT NULL,
        NOME_VIRUS            char(30) NULL,
        DESCRICAO             char(100) NULL,
        PRIMARY KEY (COD_DO_VIRUS)
)
go
CREATE TABLE CONTRATO (
        NUMERO_DO_CONTRATO    char(18) NOT NULL,
        VALOR                 money NULL,
        DATA_DE_INICIO        datetime NULL,
        DATA_DE_TERMINO       datetime NULL,
        PRIMARY KEY (NUMERO_DO_CONTRATO)
```

```
)
go
CREATE TABLE INSTITUTO (
       COD_INSTITUTO          int NOT NULL,
       NOME_INSTITUTO         char(30) NULL,
       ENDERECO               char(18) NULL,
       LOGRADOURO             char(20) NULL,
       BAIRRO                 char(20) NULL,
       MUNICIPIO              char(25) NULL,
       ESTADO                 char(25) NULL,
       CEP                    char(8) NULL,
       PRIMARY KEY (COD_INSTITUTO)
)
go
CREATE TABLE PUBLICACAO (
       COD_PUBLICACAO         char(18) NOT NULL,
       NUMERO_DO_CONTRATO     int NOT NULL,
       COD_INSTITUTO          int NULL,
       TITULO                 char(40) NULL,
       JORNAL                 char(20) NULL,
       NUMERO_VOLUME          char(3) NULL,
       NUMERO_DA_EDICAO       char(2) NULL,
       ANO                    char(4) NULL,
       TIPO                   char(1) NULL,
       PRIMARY KEY (COD_PUBLICACAO),
       FOREIGN KEY (NUMERO_DO_CONTRATO)
                              REFERENCES CONTRATO,
       FOREIGN KEY (COD_INSTITUTO)
                              REFERENCES INSTITUTO
)
go
CREATE INDEX XIF2PUBLICACAO ON PUBLICACAO
(
       COD_INSTITUTO
)
go
CREATE INDEX XIF20PUBLICACAO ON PUBLICACAO
(
       NUMERO_DO_CONTRATO
)
go
CREATE TABLE PUBLICACAO_VIRUS (
       COD_PUBLICACAO         char(18) NOT NULL,
       COD_DO_VIRUS           char(18) NOT NULL,
       PRIMARY KEY (COD_PUBLICACAO, COD_DO_VIRUS),
       FOREIGN KEY (COD_DO_VIRUS)
                              REFERENCES VIRUS,
       FOREIGN KEY (COD_PUBLICACAO)
                              REFERENCES PUBLICACAO
)
```

```
go
CREATE INDEX XIF25PUBLICACAO_VIRUS ON PUBLICACAO_VIRUS
(
        COD_PUBLICACAO
)
go
CREATE INDEX XIF26PUBLICACAO_VIRUS ON PUBLICACAO_VIRUS
(
        COD_DO_VIRUS
)
go
CREATE TABLE REFERENCIA (
        COD_PUBLICACAO        char(18) NOT NULL,
        COD_PUBLICACAO_REFERENCIADA int NOT NULL,
        PRIMARY KEY (COD_PUBLICACAO),
        FOREIGN KEY (COD_PUBLICACAO)
                                REFERENCES PUBLICACAO,
        FOREIGN KEY (COD_PUBLICACAO)
                                REFERENCES PUBLICACAO
)
go
CREATE TABLE AUTOR (
        COD_AUTOR             char(18) NOT NULL,
        NOME_AUTOR            char(25) NULL,
        NACIONALIDADE         char(15) NULL,
        DATA_DE_NASCIMENTO    datetime NULL,
        PRIMARY KEY (COD_AUTOR)
)
go
CREATE TABLE ESCREVE (
        COD_AUTOR             char(18) NOT NULL,
        COD_PUBLICACAO        char(18) NOT NULL,
        DATA_ENTREGA          datetime NULL,
        PRIMARY KEY (COD_AUTOR, COD_PUBLICACAO),
        FOREIGN KEY (COD_PUBLICACAO)
                                REFERENCES PUBLICACAO,
        FOREIGN KEY (COD_AUTOR)
                                REFERENCES AUTOR
)
go
CREATE INDEX XIF4ESCREVE ON ESCREVE
(
        COD_AUTOR
)
go
CREATE INDEX XIF5ESCREVE ON ESCREVE
(
        COD_PUBLICACAO
)
go
```

Oracle 8.xx

```
CREATE TABLE VIRUS (
       COD_DO_VIRUS          CHAR(18) NOT NULL,
       NOME_VIRUS            CHAR(30) NULL,
       DESCRICAO             CHAR(100) NULL,
       PRIMARY KEY (COD_DO_VIRUS)
);


CREATE TABLE CONTRATO (
       NUMERO_DO_CONTRATO    CHAR(18) NOT NULL,
       VALOR                 NUMBER NULL,
       DATA_DE_INICIO        DATE NULL,
       DATA_DE_TERMINO       DATE NULL,
       PRIMARY KEY (NUMERO_DO_CONTRATO)
);


CREATE TABLE INSTITUTO (
       COD_INSTITUTO         INTEGER NOT NULL,
       NOME_INSTITUTO        CHAR(30) NULL,
       ENDERECO              CHAR(18) NULL,
       LOGRADOURO            CHAR(20) NULL,
       BAIRRO                CHAR(20) NULL,
       MUNICIPIO             CHAR(25) NULL,
       ESTADO                CHAR(25) NULL,
       CEP                   CHAR(8) NULL,
       PRIMARY KEY (COD_INSTITUTO)
);


CREATE TABLE PUBLICACAO (
       COD_PUBLICACAO        CHAR(18) NOT NULL,
       NUMERO_DO_CONTRATO    INTEGER NOT NULL,
       COD_INSTITUTO         INTEGER NULL,
       TITULO                CHAR(40) NULL,
       JORNAL                CHAR(20) NULL,
       NUMERO_VOLUME         CHAR(3) NULL,
       NUMERO_DA_EDICAO      CHAR(2) NULL,
       ANO                   CHAR(4) NULL,
       TIPO                  CHAR(1) NULL,
       PRIMARY KEY (COD_PUBLICACAO),
       FOREIGN KEY (NUMERO_DO_CONTRATO)
                              REFERENCES CONTRATO,
       FOREIGN KEY (COD_INSTITUTO)
                              REFERENCES INSTITUTO
);

CREATE INDEX XIF2PUBLICACAO ON PUBLICACAO
```

```
(
        COD_INSTITUTO
);

CREATE INDEX XIF20PUBLICACAO ON PUBLICACAO
(
        NUMERO_DO_CONTRATO
);


CREATE TABLE PUBLICACAO_VIRUS (
        COD_PUBLICACAO        CHAR(18) NOT NULL,
        COD_DO_VIRUS          CHAR(18) NOT NULL,
        PRIMARY KEY (COD_PUBLICACAO, COD_DO_VIRUS),
        FOREIGN KEY (COD_DO_VIRUS)
                               REFERENCES VIRUS,
        FOREIGN KEY (COD_PUBLICACAO)
                               REFERENCES PUBLICACAO
);

CREATE INDEX XIF25PUBLICACAO_VIRUS ON PUBLICACAO_VIRUS
(
        COD_PUBLICACAO
);

CREATE INDEX XIF26PUBLICACAO_VIRUS ON PUBLICACAO_VIRUS
(
        COD_DO_VIRUS
);


CREATE TABLE REFERENCIA (
        COD_PUBLICACAO        CHAR(18) NOT NULL,
        COD_PUBLICACAO_REFERENCIADA INTEGER NOT NULL,
        PRIMARY KEY (COD_PUBLICACAO),
        FOREIGN KEY (COD_PUBLICACAO)
                               REFERENCES PUBLICACAO,
        FOREIGN KEY (COD_PUBLICACAO)
                               REFERENCES PUBLICACAO
);


CREATE TABLE AUTOR (
        COD_AUTOR             CHAR(18) NOT NULL,
        NOME_AUTOR            CHAR(25) NULL,
        NACIONALIDADE         CHAR(15) NULL,
        DATA_DE_NASCIMENTO    DATE NULL,
        PRIMARY KEY (COD_AUTOR)
);
```

```
CREATE TABLE ESCREVE (
       COD_AUTOR            CHAR(18) NOT NULL,
       COD_PUBLICACAO       CHAR(18) NOT NULL,
       DATA_ENTREGA         DATE NULL,
       PRIMARY KEY (COD_AUTOR, COD_PUBLICACAO),
       FOREIGN KEY (COD_PUBLICACAO)
                             REFERENCES PUBLICACAO,
       FOREIGN KEY (COD_AUTOR)
                             REFERENCES AUTOR
);

CREATE INDEX XIF4ESCREVE ON ESCREVE
(
       COD_AUTOR
);

CREATE INDEX XIF5ESCREVE ON ESCREVE
(
       COD_PUBLICACAO
);
```

Referências Bibliográficas

[1] BATINI, C.; CERI, S.; NAVATHE, S. B. **Conceptual Database Design:** *An Entity-Relationship Approach.* California: Benjamin/Cummings Publishing, 1992.

[2] BOEHM, B. W. **Software Engineering Economics**. New Jersey: Prentice-Hall, 1981.

[3] BOHM, C.; JACOPINI, G. **Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules**. EUA: Communications of the ACM, V. 9, N. 5, p. 366-371, 1966.

[4] BRADLEY, J. **Introduction to DataBase Management in Business**. 2. ed. New York: HRW IE, 1987.

[5] CHEN, P. P. **The Entity-Relationship Model:** *Toward a Unified View of Data.* EUA: ACM Transactions on DataBase Systems, V. 1, N. 1, p. 9-36, 1976.

[6] COOD, E. F. **A relational Model of Data for Large Shared Data Banks**. EUA: Communications ACM, V. 13, N. 6, p. 377-387, 1970.

[7] CODD, E. F. **Is Your DBMS Really Relational?**. EUA: Computer-World, outubro, 1985.

[8] DEMARCO, T. **Structured Analisys and System Specification**. New Jersey: Prentice-Hall, 1978.

[9] DIJKSTRA, E. W. **Go To Statements Considered Harmful**. EUA: Communications of the ACM, V. 11, N. 3, p. 147-148, maio, 1966.

[10] FERNANDES, A. C.; LAENDER, A. H. F. **MER+:** *uma extensão do modelo de entidades e relacionamentos para o projeto conceitual de Banco de Dados.* Rio de Janeiro: Revista Brasileira de Computação, V. 5, N. 1, 08/09, SBC/NCE, 1989.

[11] FLAVIN, M. **Fundamental Concepts of Information Modeling.** New York: Yourdon Press, 1981.

[12] GANE, C; SARSON, T. **Structured Systems Analysis:** *Tools and Techniques.* New Jersey: Prentice-Hall, 1979.

[13] KENT, W. **A Simple Guide to Five Normal Forms in Relational DataBase Theory**. EUA: Communications of the ACM, V. 26, N. 2, p. 120--125, 1983.

[14] KIPPER, E. F. et al. **Engenharia de Informações:** *Conceitos, Técnicas e Métodos.* Porto Alegre: SAGRA-DC LUZZATO, 1993.

[15] KOWAL, J. A. **Analyzing Systems**. New Jersey: Prentice-Hall, 1988.

[16] MACIASZEK, L. A. **Database Design and Implementation.** Austrália: Prentice-Hall, 1990.

[17] MARTIN, J.; FINKELSTEIN, C. **Information Engineering**. V. I / II., Inglaterra, Savant Institute, 1981.

[18] SETZER, V. W. **Bancos de Dados:** *Conceito, Modelos, Gerenciadores, Projeto Lógico e Projeto Físico.* São Paulo: Edgard Blücher, 1986.

[19] SHLAER, S.; MELLOR, J. S. **Análise de Sistemas Orientada para Objetos**. São Paulo: McGraw-Hill/Newstec, 1990.

[20] STAA, A. V. **Engenharia de Programas**. Rio de Janeiro: LTC, 1986.

[21] STEVENS, W.; et al. **Structured Design**. EUA: IBM Systems Journal, V. 13, N. 2, p. 115-139, 1979.

[22] TASKER, D. **Fourth Generation Data:** *A Guide to Data Analysis for New and Old Systems.* Austrália: Prentice Hall, 1989.

[23] TRIMBLE (Jr.), J. H.; CHAPPELL, D. **A Visual Introdution to SQL.** New York: John Wiley & Sons/Metaphor Computer System INC., 1989.

[24] YOURDON, E. **Análise Estruturada Moderna**. Rio de Janeiro: Campus, 1990.

[25] YOURDON, E.; COAD, P. **Análise Baseada em Objetos**. Rio de Janeiro: Campus, 1992.

Marcas Registradas

Todos os nomes citados, marcas mencionadas, ou direitos de uso citados pertencem aos seus respectivos proprietários.

PROJETO DE BANCO DE DADOS

Uma Visão Prática

Neste livro, os consultores Felipe Machado e Maurício Abreu unificam a visão didática e a prática, referentes aos aspectos fundamentais do Projeto de Banco de Dados. Os dois colocam à disposição do leitor sua larga experiência no desenvolvimento de aplicações de banco de dados, adquirida em vários projetos envolvendo áreas, tais como: Administração, Recursos Humanos, Produção Industrial, Finanças, Materiais, Oceanografia, Petróleo e outras mais. Os profissionais de desenvolvimento de aplicações em banco de dados (Analistas de Sistemas, Programadores, Analistas de Negócio e Administradores de Dados), assim como os estudantes universitários que iniciam nessas áreas, encontrarão assuntos fundamentais para o desenvolvimento desses projetos, tais como: modelagem conceitual de dados, identificação de elementos de informação presentes em uma realidade, abordagem relacional, passagem do modelo de dados para o modelo relacional, construção de consultas utilizando linguagem SQL com sintaxe ANSI e Microsoft® SQL Server™ 7.0, construção de bancos de dados físicos em exemplos utilizando Oracle® 8.xx e Microsoft® SQL Server™ 7.0. Neste trabalho, é enfatizado de forma profunda o uso do modelo Entidade Relacionamento (E-R) como ferramenta básica para o projeto de banco de dados. São apresentados vários exemplos práticos pertencentes ao mundo real. Cada exemplo é profundamente detalhado, sendo apresentado passo a passo o processo de desenvolvimento lógico de forma simples e objetiva.

Os autores preocuparam-se em colocar neste texto um capítulo que indica ao profissional por onde começar um projeto de banco de dados, detalhando de forma didática os passos que devem ser seguidos para alcançar o sucesso. Talvez pela primeira vez em uma literatura técnica desta área é colocado no texto o chamado "Pulo do Gato".

Invista em você.
Visite uma livraria.

www.editoraerica.com.br

ISBN: 85-7194-312-5
9 788571 943124