

Banco de Dados

Projeto e Implementação

Felipe Nery Rodrigues Machado

Banco de Dados

Projeto e Implementação

3^a Edição



www.editoraerica.com.br

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

Machado, Felipe Nery Rodrigues

Projeto e implementação de banco de dados / Felipe Nery Rodrigues Machado. -- 3. ed. --
São Paulo : Érica, 2014.

Bibliografia.

ISBN 978-85-365-0984-6

1. Banco de dados I. Título.

14-05221

CDD-005.75

Índice para catálogo sistemático:

1. Bancos de Dados: Sistemas: Projeto e implementação: Processamento de dados 005.75

Copyright © 2004 da Editora Érica Ltda.

Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida por qualquer meio ou forma sem prévia autorização da Editora Érica. A violação dos direitos autorais é crime estabelecido na Lei nº 9.610/98 e punido pelo Artigo 184 do Código Penal.

Coordenação Editorial: Rosana Arruda da Silva
Capa: Samir M. de Machado
Maurício S. de França
Editoração e Finalização: Dalete R. de Oliveira
Lívia Vilela de Lima
Rosana Ap. A. dos Santos
Marlene Teresa S. Alves

O Autor 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 erros estarão disponíveis para download no site da Editora Érica.

Conteúdo adaptado ao Novo Acordo Ortográfico da Língua Portuguesa, em execução desde 1º de janeiro de 2009.

A Ilustração de capa e algumas imagens de miolo foram retiradas de <www.shutterstock.com>, empresa com a qual se mantém contrato ativo na data de publicação do livro. Outras foram obtidas da Coleção MasterClips/MasterPhotos® da IMSI, 100 Rowland Way, 3rd floor Novato, CA 94945, USA, e do CorelDRAW X5 e X6, Corel Gallery e Corel Corporation Samples. Copyright® 2013 Editora Érica, Corel Corporation e seus licenciadores. Todos os direitos reservados.

Todos os esforços foram feitos para creditar devidamente os detentores dos direitos das imagens utilizadas neste livro. Eventuais omissões de crédito e copyright não são intencionais e serão devidamente solucionadas nas próximas edições, bastando que seus proprietários contatem os editores.

Seu cadastro é muito importante para nós

Ao preencher e remeter a ficha de cadastro constante no site da Editora Érica, você passará a receber informações sobre nossos lançamentos em sua área de preferência.

Conhecendo melhor os leitores e suas preferências, vamos produzir títulos que atendam suas necessidades.

Contato com o editorial: editorial@editoraerica.com.br

Editora Érica Ltda. | Uma Empresa do Grupo Saraiva

Rua São Gil, 159 - Tatuapé

CEP: 03401-030 - São Paulo - SP

Fone: (11) 2295-3066 - Fax: (11) 2097-4060

www.editoraerica.com.br

Dedicatória

Aos meus filhos Samir e Thamis pelo amor e carinho que me dedicam e por serem a razão do meu viver. *In memoriam* ao meu pai, Dr. Nery de Almeida Machado, que me ensinou determinação e caráter, e acima de tudo me ensinou que honestidade é o bem maior de uma vida. À minha esposa Maria Cristina Vaz Machado e a meus enteados João Pedro, Luiz Felipe e Lucas pelo carinho e amor que me dedicam e pela família que constituímos.

Agradecimentos

A Deus, por estar sempre ao meu lado, com seus anjos e com sua energia irradiante, que me mantém sempre com apoio e segurança, mesmo nos momentos mais difíceis em que nEle tenho o repouso e o empurrão para a frente.

Produzirão as montanhas frutos de paz ao vosso povo; e as colinas, frutos de justiça.

SI 71, 3

Sobre o Autor

Felipe Nery Rodrigues Machado

Consultor com mais de vinte anos de vivência na área de informática, na qualidade de projetista de sistemas. Com formação em Engenharia Mecânica, possui larga experiência no projeto de banco de dados com profundos conhecimentos de metodologias e modelagem de dados.

Especialista em projetos de bancos de dados para aplicações transacionais e gerenciais, possui amplo conhecimento no desenvolvimento de projetos de bancos de dados para as mais diversas áreas de negócio, tais como indústria metalúrgica, indústria de alimentos, varejo e atacado, jornais e televisão, distribuição de produtos, concessionárias de automóveis, órgãos públicos diversos, hospitais e companhias aéreas. Sua experiência abrange o ciclo completo de negócios de uma organização, tendo já desenvolvido igualmente aplicações com Arquitetura de Data Warehouse, processos de ETL, com grande ênfase em modelagem multidimensional e arquitetura de processos OLAP.

Professor universitário de disciplinas de Bancos de Dados e Metodologias de Desenvolvimento, no Rio de Janeiro, dedica-se à pesquisa e divulgação das técnicas e metodologias do estado da arte em desenvolvimento de aplicações.

Autor dos livros *Tecnologia e Projeto de Data Warehouse*, *Banco de Dados: Projeto e Implementação* e *Análise Relacional de Sistemas*, todos publicados pela Editora Érica e adotados nas principais universidades de informática do País.

E-mail: fnr.machado@gmail.com

Sumário

Capítulo 1 - O Que é Projeto de Banco de Dados	13
Modelagem de Dados	14
Que É Abstração?	16
Minimundo	18
Banco de Dados	18
Modelo Conceitual	18
Modelo Lógico	19
Modelo Físico	20
O Projeto de Banco de Dados	21
O Modelo Entidade-Relacionamento (ER)	23
A Implementação de Banco de Dados	24
Conclusão	26
Capítulo 2 - Abstração em Modelagem de Dados	27
Classificação de Abstração	28
Agregação de Abstração	29
Generalização de Abstração	29
A Cozinha	31
Começar por Dados	33
Agregações Binárias	33
Cardinalidade Mínima	34
Cardinalidade Máxima	36
Exercícios de Abstração	37
Capítulo 3 - Bancos de Dados Relacionais	39
Introdução	39
Teoria Relacional	40
Características Principais de uma Relação (Tabela Relacional)	43
Domínio	44
Chave Primária	46
Valores Nulos e Integridade de Identidade	51
Regra de Integridade de Identidade	53
Chave Primária	53
Chave Estrangeira	54
Integridade Referencial	57
Restrições para Garantir a Integridade Referencial	59
Resumo de Restrições de Integridade Relacional	61
Exercícios	62

Capítulo 4 - Modelo Entidade-Relacionamento.....	65
Elementos de um Modelo ER.....	67
Entidades	68
Relacionamentos.....	70
Atributos	72
Grau de Relacionamento	75
Conectividade de um Relacionamento.....	76
Conectividade Um-para-Um	77
Conectividade Um-para-Muitos	78
Conectividade Muitos-para-Muitos	79
Atributos em um Relacionamento	80
Opcionalidade de Relacionamento	82
Condicionalidade de um Relacionamento.....	82
Relacionamentos Reflexivos	83
Resolução de Relacionamentos Muitos-para-Muitos.....	85
Entidade Fraca ou Dependente	92
Como Reconhecer Entidades na Prática	92
Como Reconhecer Relacionamentos.....	93
Uma Distribuidora de Filmes	94
Exercícios	99
Capítulo 5 - Extensões do Modelo Entidade-Relacionamento	102
Generalização: Supertipos e Subtipos	102
Relacionamentos Ternários	106
Modelagem de Atributos (Modelagem Lógica).....	108
O Problema	110
Quando os Fatos Podem Confundir.....	115
Capítulo 6 - Agregação: Uma Extensão Especial	122
O Hotel	129
Explicação Adicional sobre as Chaves	136
Clínicas Médicas	136
A Fábrica	144
Regras para Identificar e Utilizar Agregação.....	146
Agregação Reflexiva	152
Produto Composto e Componente.....	155
Uma Distribuidora de Filmes - O Retorno	157
Conclusão	158
Capítulo 7 - Tratamento de Interpretações de Dados.....	159
Pontos de Vista Diferentes.....	159
Relacionamentos entre Interpretações	162

Tratamento de Subinterpretações.....	165
Mais Interpretação	167
Diagrama Hierárquico de Interpretações	174
Conclusão	175
Cuidado, Abra o Olho!.....	176
Capítulo 8 - Normalização.....	177
Conceituação	177
Primeira Forma Normal (1FN).....	179
Segunda Forma Normal (2FN).....	184
Terceira Forma Normal (3FN).....	187
Forma Normal de Boyce/Codd (FNBC).....	190
Entidade Cliente	192
Entidade Agência	192
Entidade Empréstimos	193
Quarta Forma Normal (4FN).....	195
Quinta Forma Normal (5FN)	200
Roteiro de Aplicação da Normalização	200
Aplicação da 1FN	200
Aplicação da 2FN	200
Aplicação da 3FN	201
Aplicação da FNBC	201
Aplicação da 4FN	201
Aplicação da 5FN	201
Considerações Finais sobre Normalização.....	202
Desnormalização dos Dados	203
Alguns Motivos para a Desnormalização	203
Capítulo 9 - Um Estudo de Caso	204
O Problema - Administração de Cirurgias.....	204
Modelagem	205
Capítulo 10 - Hierarquias	215
Tratamento de Hierarquias de Dados	215
Capítulo 11 - Modelo Físico	218
O Modelo Físico em Si.....	218
Propriedades de uma Coluna	219
A Opção de Nulo	220
Uma Regra de Validação	220
Valor Padrão	221
Visões de Dados	221

Índices do Banco de Dados	223
Chaves Substitutas.....	223
As Generalizações	224
Tabelas do Exemplo.....	225
Relação entre Modelo Lógico e Modelo Físico	227
Capítulo 12 - Mapeamento de Objetos ER	231
Mapeamento de Objetos para Tabelas (ER).....	232
Regra 1.....	233
Classes com Coleções de Objetos	234
Regra 2.....	234
Regra 3 - Transposição de Associações Um-para-Um.....	235
Regra 4 - Transposição de Associações Um-para-Muitos	236
Regra 5 - Transposição de Associações Um-para-Muitos com Classe de Associação.....	236
Regra 6 - Transposição de Associações Muitos-para-Muitos.....	237
Regra 7 - Transposição de Associações Muitos-para-Muitos com Classe de Associação.....	237
Regra 8 - Transposição de Generalizações	238
Regra 9 - Transposição de Agregações.....	240
Considerações Finais	241
Capítulo 13 - Álgebra Relacional	242
Álgebra Relacional e Operações Relacionais	242
Os Operadores de Álgebra Relacional.....	243
Operação de Projeção	244
Operação de Seleção.....	247
Produto Cartesiano	251
Operação Renomear	259
Operação de União (Binária).....	266
Operação de Diferença (Binária).....	269
Operação de Intersecção de Tabelas	273
Operação de Junção	275
Operação de Divisão.....	281
Exercícios Resolvidos e Operadores Adicionais.....	284
Funções em Álgebra Relacional	290
Projeção Generalizada.....	292
Operação de Semijunção	298
Junção Externa (Outer-Join)	300
Exercícios Propostos.....	302
Capítulo 14 - SQL.....	306
A Importância da Linguagem SQL	306
A Linguagem SQL	307

Vantagens e Desvantagens da Linguagem SQL	309
O Exemplo.....	310
Criação e Distribuição de Tabelas	313
Criação de Tabelas	314
Criação de Chaves Primárias Compostas	317
Eliminação de uma Tabela	318
Alteração da Estrutura das Tabelas	318
Coluna Calculada	320
Criação de Ações em Cascata.....	321
A Cláusula ON DELETE CASCADE e ON UPDATE CASCADE.....	321
Regras de Validação	321
Extração de Dados de uma Tabela: SELECT	322
Seleção de Colunas Específicas da Tabela.....	322
Seleção de Todas as Colunas da Tabela	324
Alteração do <i>Heading</i> (Cabeçalho) da Coluna	325
Manipulação de Dados Numéricos: Operadores Aritméticos.....	326
Seleção de Somente Algumas Linhas da Tabela	326
Comparações na Cláusula WHERE.....	327
Ordenação dos Dados Selecionados.....	334
Realização de Cálculos com Informação Selecionada	337
Utilização de Funções de Agregação sobre Conjuntos	338
Busca de Máximos e Mínimos (MAX, MIN)	338
Totalização dos Valores de Colunas (SUM)	338
Cálculo de Médias (AVG).....	339
Contagem dos Registros (COUNT).....	339
Utilização da Cláusula DISTINCT	339
Agrupamento de Informações Selecionadas (GROUP BY e HAVING)	340
Utilização com HAVING	342
Recuperação de Dados de Várias Tabelas (JOINS)	342
O Conceito de Qualificadores de Nome	342
<i>Inner Joins</i>	344
<i>Cross Join</i> ou Produto Cartesiano	345
<i>Outer Joins</i>	346
Uso de Aliases	350
Junção de Mais de Duas Tabelas	351
Utilização de Consultas Encadeadas (Subqueries)	354
Inserir, Modificar e Apagar Registros	358
Adição de Registro à Tabela	358
Adição de Registros com um SELECT	359
Atualização de um Registro - UPDATE.....	360
Alteração de Registros com Dados de Outra Tabela.....	362
Apagar Registros da Tabela.....	363

Apagar Registros da Tabela com Base em Dados de Outra Tabela	364
Utilização de Views	364
Criação de uma <i>View</i> por meio de um <i>Join</i>	366
Utilização de uma View	367
Listagem	367
Inserção de Linhas numa <i>View</i>	367
Modificação de uma Linha da <i>View</i>	368
Apagar	368
Eliminação de uma <i>View</i>	368
Garantia dos Privilégios de Acesso - GRANT e REVOKE	369
Comando GRANT (Garantir)	369
Lista de Opções de Privilégios	370
Comando REVOKE (Revogação)	371
Trabalho com Índices	372
Um Checklist para Criação de Índices	372
Quando Não Criar Índices	373
Criação de Índices.....	373
Eliminação de Índices.....	374
Tópicos Avançados de SQL	375
Combinação de Resultados de Pesquisas (UNION).....	375
Realização de um <i>Join</i> entre uma Tabela e Ela Mesma	376
NVL	377
A utilização da expressão condicional DECODE	378
A utilização da expressão condicional CASE.....	378
Trabalho com tempo (campos date).....	379
Estudo de Caso - Banco de Dados	379
Capítulo 15 - Modelagem de Dados e Métodos Ágeis.....	386
Métodos Ágeis	386
A refatoração de banco de dados	389
O futuro da modelagem de dados	390
Bibliografia	391
Índice Remissivo	393

Prefácio

A Modelagem de Dados e o Projeto de Banco de Dados têm sofrido significantes evoluções nos anos recentes com o avanço de aplicações empresariais com uso do modelo de dados relacional e aplicações em sistemas de bancos de dados.

Este livro continua a concentrar-se nas técnicas de modelagem e projeto de bancos de dados para o ambiente relacional com a abordagem entidade-relacionamento, buscando a criação de modelos conceituais, lógicos e físicos de dados, porém com um olhar sobre as metodologias do estado da arte no desenvolvimento de aplicações como as tecnologias orientadas a objetos.

Entendemos que o desenvolvimento de aplicações continua enfrentando os mesmos problemas do surgimento dos *papers* do Dr. Peter Chen, pela inexistência de farta informação sobre as técnicas de modelagem de dados com exemplos do mundo real e também, em parte, pelo pouco estímulo dado aos trabalhos e tempos gastos com o processo de projetar bancos de dados. Outrossim, o vínculo entre os conceitos de Álgebra Relacional na interpretação e construção de consultas em SQL para bancos de dados relacionais é explorado com intuito de fornecer ao desenvolvedor uma visão mais ampla dos caminhos de navegação por um modelo de dados.

As múltiplas extensões do modelo original de Peter Chen também foram preocupação neste livro, que explora e apresenta extensões tais como generalização, agregação, assim como o tratamento a ser dado, durante a análise, às interpretações diferentes de objetos do mundo real que originam as hierarquias e visões de dados, com exemplos aderentes a situações simples do dia a dia.

A importância das chaves primárias e estrangeiras também é bastante discutida e estudada, objetivando criar no desenvolvedor o costume de sempre utilizar esse padrão, que nada mais é que uma exigência técnica para integridade e acesso a um banco de dados, seja ele de que tipo for ou para que aplicação se destine.

Atualmente existe uma incrível sede de conhecimento para a execução de projetos de software no País, e despretensiosamente buscamos contribuir em formato educacional e sem academicismos com uma técnica já consagrada de projeto de software, mas cujas opções de solução ainda são pouco exploradas, e importante por sua capacidade de aderência ao mundo real e sua expressividade nos processos de negócios.

Desde a antiguidade o homem procura transmitir e documentar seu conhecimento, objetos e situações da vida real. Nas cavernas pré-históricas foram encontrados desenhos de animais, caçadas e cenas do cotidiano. Com símbolos que representavam objetos e animais, os habitantes das cavernas eternizavam a sua realidade. O homem evoluiu.

As metodologias que chegam ao País, do exterior, centralizam tudo quanto aos projetos de desenvolvimento de aplicativos. Estes são, por consequência, proprietários dos dados. A modelagem de dados fica então reduzida, simplesmente, a uma atividade paralela, quase desconhecida e muitas vezes desprezada. Quando utilizada, seu objetivo é meramente documental.

Ao reconhecer os dados como um dos recursos mais importantes das corporações, ultrapassando as próprias fronteiras dos sistemas aplicativos, a modelagem de dados se torna, com justa razão, a mais importante técnica utilizada na produção dos resultados das fases de planejamento e análise dos projetos de sistemas de informação na era dos bancos de dados relacionais.

Os princípios básicos da modelagem de dados já são, nos dias de hoje, conhecidos, ensinados e divulgados pelos mais diversos meios. Na vida real, entretanto, muitas vezes nos defrontamos com situações inusitadas, desconhecidas e que nunca foram cobertas ou sequer mencionadas em cursos e publicações. Os projetistas de sistemas de informações precisam então acumular uma vasta experiência na área, para fazer frente a esses eventos, ou procurar auxílio externo. Neste sentido, o autor deste livro disponibiliza sua vasta experiência, de uma maneira clara, simples e didática, desde os primeiros passos da arte de modelar dados até as mais avançadas soluções.

A questão que sempre se coloca é: quantos modelos de dados você realiza por ano?

Sabemos que durante a permanência em uma organização nos envolvemos sempre com um ou dois sistemas de aplicação, e na maior parte realizamos muitas das vezes somente manutenção de aplicações. Isso leva a um ócio de modelagem de dados e limita a exploração de soluções e o próprio domínio amplo da técnica, independentemente da sua importância. Estamos cada dia mais ligados a códigos do que ao projeto de uma aplicação em si.

É preciso procurar 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 de toda a aplicação. Neste contexto, a modelagem de dados continua a ter uma importância maior ainda.

O desafio de criar um livro de leitura e compreensão fáceis, que seja instrumento de referência de projeto de bancos de dados, foi o fator que motivou a escrevê-lo.

Nesta terceira edição, apresentamos um novo capítulo importante para apresentar a relação entre o projeto e a implementação de bancos de dados e a utilização de métodos ágeis, como o Scrum, assim como as implicações decorrentes desta associação.

Destacamos os fatores importantes a serem considerados na gestão da implantação de banco de dados em conjunto com essas metodologias, buscando colocar o conteúdo deste livro aderente ao estado da arte em desenvolvimento de software.

O autor



O QUE É PROJETO DE BANCO DE DADOS

Durante muito tempo criou-se a ideia de que projetar bancos de dados era uma disciplina com identidade própria, uma atividade específica e até certo ponto isolada no ciclo de vida de um sistema, que tinha existência própria e podia ser realizada a partir de primitivas e conceitos exclusivos da técnica de modelagem de dados.

Com a evolução das tecnologias de desenvolvimento de aplicações, novas linguagens e principalmente com o advento da orientação a objetos não mais restrita ao ambiente de programação, mas extensiva às metodologias de análise de sistemas, o trabalho de projetar as bases de dados que serão utilizadas por um sistema em desenvolvimento assume, nos dias de hoje, características que objetivam mixar um projeto orientado a objetos com as necessidades de esse mesmo sistema interagir com um banco de dados relacional, constituído por um conjunto de tabelas, que equivale à denominada camada de persistência de dados.

Essa necessidade de mixagem é real pela absoluta ausência de projetos comerciais que utilizem bancos de dados orientados a objetos que sejam confiáveis a grandes massas de dados, à não popularização desses produtos e aos grandes investimentos já realizados em softwares de Sistemas Gerenciadores de Bancos de Dados Relacionais existentes no mercado nacional e mundial.

É indiscutível a vantagem obtida em um projeto orientado a objetos; logo, surge a necessidade de uma nova técnica de projetar banco de dados, que não é a formatação pura de classes de dados, mas uma interação alta entre o ambiente de análise orientada a objetos e a nossa sempre presente modelagem de dados, que é estritamente necessária à administração de dados da organização de TI. A utilização de ferramentas para a camada de persistência, tais como o Hibernate, acaba levando o desenvolvedor a deixar de lado a preocupação com as estruturas do banco de dados, assim como a preocupação com a qualidade das queries realizadas em uma aplicação.

Não existem técnicas nem ferramentas que possibilitem tanto ao administrador de dados (DA) quanto ao administrador de bancos de dados (DBA) realizar suas funções sobre classes de dados, pois esses mesmos bancos de dados relacionais atuam e têm todas as suas funcionalidades sobre tabelas relacionais de dados, as quais são hoje de domínio maior dos usuários de aplicações.

Há muito tempo escrevemos sobre modelagem de dados e o processo continua existindo como sempre existiu, porém o que desejamos neste livro é apresentar essas técnicas integradas com a análise orientada a objetos, de forma a permitir que quem trabalha tanto com OO como com análise estruturada tenha domínio de projetos de bancos de dados eficientes a uma aplicação, seja qual for o ambiente de desenvolvimento em que esteja. Busca-se ainda permitir que um projeto totalmente desenvolvido em OO seja facilmente inserido em um ambiente de banco de dados relacional, com o mínimo de traumas e mantida a coerência com os objetos de negócios de uma organização.

Modelagem de Dados

Modelagem de dados é o estudo das informações existentes em um contexto sob observação para a construção de um modelo de representação e entendimento de tal contexto. A modelagem de dados minera as informações que representam um contexto, estruturando-as em um conjunto que denominamos modelo lógico de dados.

Uma das principais características da abordagem de modelagem de banco de dados é que ela fornece níveis de abstração de dados que omitem do usuário final detalhes sobre o armazenamento dos dados. Não existe a preocupação com um banco de dados tecnologicamente falando.

O modelo de dados é um conjunto de conceitos que podem ser utilizados para descrever as estruturas lógicas e físicas de um banco de dados.

Já um modelo de classes de objetos não se detém somente às informações e dados, sendo mais amplo no momento em que integra as informações e os seus métodos de acesso, recuperação, processamento e outros em um único objeto.

Logo, temos de encontrar e separar os dados dos seus processos em um determinado momento, para que possamos efetivamente obter domínio do negócio para o qual a aplicação está sendo desenvolvida. Os objetivos reais da aplicação continuam sendo mais facilmente compreendidos por meio de um modelo de dados, e não de um modelo de objetos.

Normalmente as aplicações têm sido desenvolvidas com AOO (Análise Orientada a Objetos) mesclada com a tradicional modelagem de dados, com a utilização de *patterns*, ou melhor, padrões de modelos de dados, que não são obtidos tão simplesmente quando da pura criação de diagramas de classes. Desenvolvemos com isso um processo que pode ser chamado *two fase commit* entre diagrama de classes e modelo de dados Entidade-Relacionamento.

Para que isso possa ser realizado, é preciso entender e dominar as técnicas de modelagem de dados e depois, finalmente, aprender a lidar com a sua equivalência em um modelo de classes de objetos.

O objetivo deste livro é explorar e ensinar da melhor forma possível como modelar dados para ambientes de bancos de dados relacionais, complementando com a implementação desses bancos seja em um ambiente tradicional, seja em um ambiente orientado a objetos, tal como uma aplicação desenvolvida em Java, com base em experiências por nós realizadas.

Estrutura é a forma como esses dados estão agrupados, os relacionamentos entre eles e as restrições que podem recair sobre eles, o que não é totalmente explícito em um diagrama de classes da análise orientada a objetos.

Historicamente, os primeiros modelos de bancos de dados datam da década de 1960. Desde então, a pesquisa científica na área procura evoluir no sentido de definir, encontrar modelos que representem da melhor maneira possível os dados de uma realidade, ou seja, que organizem os dados de uma forma mais próxima da maneira como são vistos e manipulados pelas pessoas no mundo real.

A ideia é definir abstrações que facilitem a compreensão da organização dos dados pelo usuário, tornando cada vez mais transparente e independente o conhecimento da organização física, independente de precisar utilizar o conhecimento técnico de um conjunto de técnicas orientadas a objetos para ter esse entendimento.

O objetivo de um modelo de dados é ter certeza de que todos os objetos de dados existentes em determinado contexto e requeridos pela aplicação e pelo banco de dados estão completamente representados e com precisão. Pelo fato de os dados modelados usarem notações facilmente compreendidas e em um idioma natural, eles podem ser revisados e verificados pelos usuários finais.

O modelo de dados também deve ser detalhado o bastante para ser usado pelo implementador (DBA) do banco de dados como uma espécie de fotocópia para construir o banco de dados físico.

Será utilizada toda a informação que está no modelo de dados lógico para definir as tabelas de um banco de dados relacional, chaves primárias e estrangeiras, procedimentos armazenados (*stored procedures*) e gatilhos (*triggers*).

Um banco de dados mal projetado requer mais tempo e retrabalho a longo prazo.

Sem planejamento e análise cuidadosa você pode criar um banco de dados que omita alguns dados exigidos, ou inconsistente em relação ao contexto de informações que ele deve refletir. Os resultados incorretos ou incompatíveis em uma aplicação sistêmica impossibilitam inclusive acomodar as mudanças de exigências dos usuários ao longo do tempo, ou implicam que o banco de dados tenha constante manutenção em suas estruturas e a aplicação fique extremamente dependente do próprio banco de dados.

Para ser efetivo, um modelo de dados deve ser simples o bastante para comunicar ao usuário final a estrutura de dados requerida pelo banco de dados e bastante detalhado para o administrador de banco de dados usar para criação da estrutura física correspondente em um SGBD.

O modelo de Entidade-Relacionamento (ER) é o método mais comum para construção de modelos de dados para bancos de dados de relacionais.

O mais comum em um modelo de dados é uma pequena coleção de mecanismos de abstração ditos primitivos, que são classificação, agregação e generalização.

Abstrações ajudam o analista a entender, classificar e modelar uma realidade, alguma coisa do mundo real que está em observação.

Que É Abstração?

Vamos levar o leitor a resolver exercícios que possibilitem um maior desenvolvimento dessa sua capacidade, antes de desenvolvermos os conhecimentos e técnicas específicos de modelagem de dados.

Uma abstração ou a capacidade de abstração é um processo, com certeza mental, que usamos quando selecionamos várias características e propriedades de um conjunto de objetos ou fatos, e excluímos outras que não são relevantes em um contexto. Em outras palavras, aplicamos abstração quando nos concentramos nas propriedades de um conjunto de objetos ou coisas que consideramos essenciais, e desprezamos outras que não consideramos importantes, sempre dentro de um determinado contexto sob observação ou análise.

O analista, durante a modelagem conceitual dos dados, deve se concentrar na observação dos objetos relevantes que existem em uma realidade ou contexto, com a finalidade de construir um modelo de compreensão e conceitos existentes nessa realidade. Esse primeiro modelo denominamos de minimundo, sem pensar no momento em automatizar ou processar a informação dessa realidade.

Abstração, em síntese, nada mais é do que a visão, sem conceituações técnicas, que obtemos na mente de uma realidade qualquer do mundo real.

Por exemplo, quando olhamos a figura ao lado, temos o resultado de um processo de abstração em que excluímos detalhes da estrutura de uma bicicleta, como os pedais, os freios, os mecanismos de tração e inclusive as possíveis diferenças entre várias bicicletas, e definimos o objeto como o que conceituamos e denominamos de "bicicleta". Normalmente associamos um nome a cada abstração. No caso da figura, o conceito bicicleta é uma representação dessa prática de abstração. Definimos um conceito para um objeto existente em contexto sob observação. Denominamos essa capacidade humana de modelo conceitual.



Em realidade o que criamos é um modelo de conceitos de objetos em um primeiro instante que denominamos de modelo conceitual.

Outra representação poderia ser a descrição textual da mesma figura apresentada. Vamos detalhar mais os conceitos de abstração e as primitivas em capítulo à parte neste livro.

Quando modelamos, como veremos no livro, devemos buscar a realidade do ambiente, o contexto em análise com total abstração em primeiro lugar, para somente depois iniciarmos um processo de modelagem lógico que detalharemos adiante.

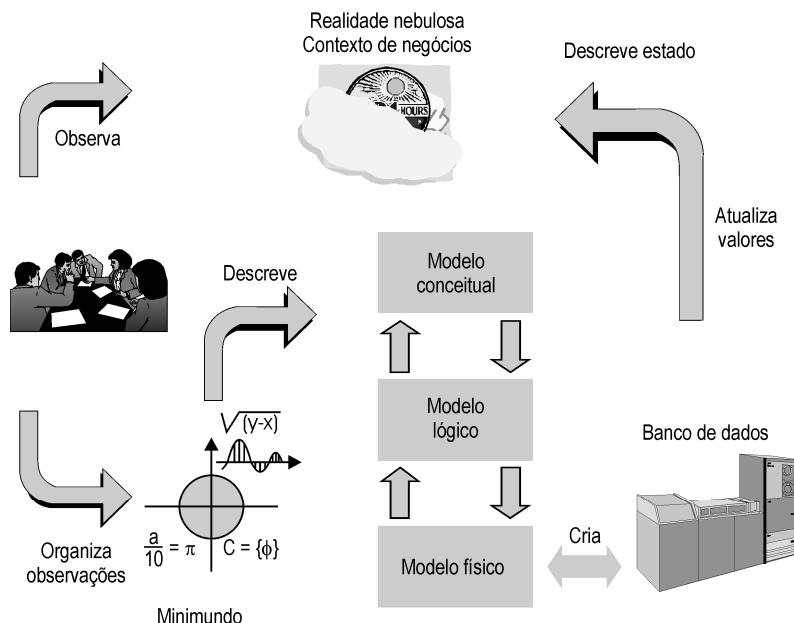
Ao coletar e selecionar os objetos relevantes de um contexto, o analista deve identificar os elementos geradores de cada objeto, as leis que os regem no contexto, bem como as operações que incidem sobre os objetos básicos (dados).

Nesse momento, os documentos que registram esses objetos ou elementos de negócio 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 de 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 um contexto) precisam para alcançar os objetivos em seu ambiente de negócios.

Para registrarmos as necessidades de informação de uma realidade e sempre dentro do contexto desta realidade, precisamos fazer uso de um modelo de dados, ou seja, algo que nos mostre as informações existentes e como elas estão relacionadas (regras de negócio).

Com base nesse modelo criado, os analistas podem interagir com os usuários validando seus objetivos e metas, permitindo o detalhamento do que denominamos base de dados para construção de um sistema de informações aderente às necessidades dos usuários finais.



Em seguida apresentamos uma descrição dos principais macroelementos construídos com a aplicação de técnicas de abstração.

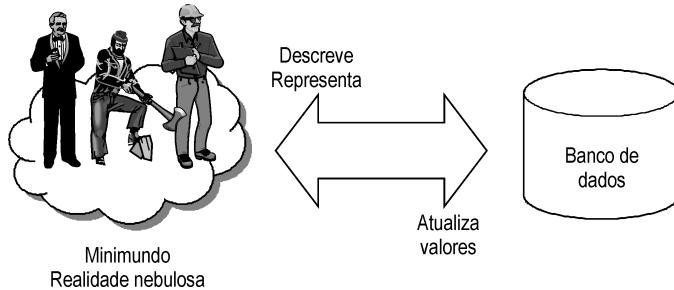
Minimundo

Porção da realidade, captada pelo analista, que a gestão de negócios de uma organização tem interesse em observar, controlar e administrar. A complexidade existente no momento de analisar um minimundo pode levar o analista a subdividi-lo em partes menores, às quais damos o nome de visão de processo de negócio.

Banco de Dados

Um banco de dados pode ser definido como um conjunto de dados devidamente relacionados. Podemos compreender como dados os objetos conhecidos que podem ser armazenados e que possuem um significado implícito, porém o significado do termo banco de dados é mais restrito que simplesmente a definição dada anteriormente. Um banco de dados possui as seguintes propriedades:

- ▶ É uma coleção lógica coerente de dados com um significado inerente; uma disposição desordenada dos dados não pode ser referenciada como banco de dados.
- ▶ Ele é projetado, construído e preenchido com valores de dados para um propósito específico; um banco de dados possui um conjunto predefinido de usuários e de aplicações.
- ▶ Ele representa algum aspecto do mundo real, o qual é chamado de minimundo; qualquer alteração efetuada no minimundo é automaticamente refletida no banco de dados.



Modelo Conceitual

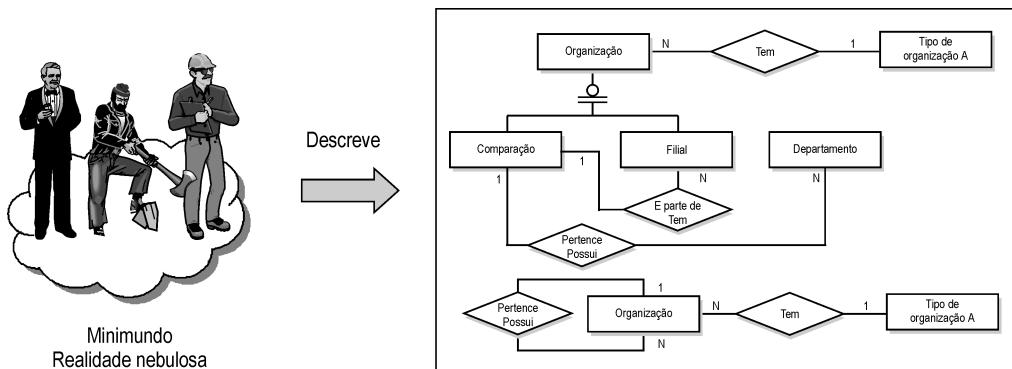
Representa, descreve a realidade do ambiente do problema, constituindo-se em uma visão global dos principais dados e seus relacionamentos (estruturas de informação), completamente independente dos aspectos de sua implementação tecnológica.

Quando falamos em modelo conceitual, estamos nos referindo àquela que deve ser sempre a primeira etapa de um projeto de banco de dados.

O objetivo do modelo conceitual é descrever de forma simples e facilmente compreendida pelo usuário final as informações de um contexto de negócios, as quais devem ser armazenadas em um banco de dados.

É uma descrição de alto nível (macrodefinição), mas que tem a preocupação de captar e retratar toda a realidade de uma organização, processo de negócio, setor, repartição, departamento etc.

É importante destacar que o modelo conceitual não retrata nem é vinculado aos aspectos ligados à abordagem do banco de dados que será utilizado, tampouco se preocupa com as formas de acesso ou estruturas físicas implementadas por um Sistema Gerenciador de Banco de Dados específico. **Sem modelo conceitual não temos uma visão clara das regras do negócio e acabamos por criar aplicações sem entender para que elas foram criadas.**



O resultado de um modelo conceitual é um esquema gráfico que representa a realidade das informações existentes em determinado contexto de negócios, assim como as estruturas de dados em que estão organizadas essas informações.

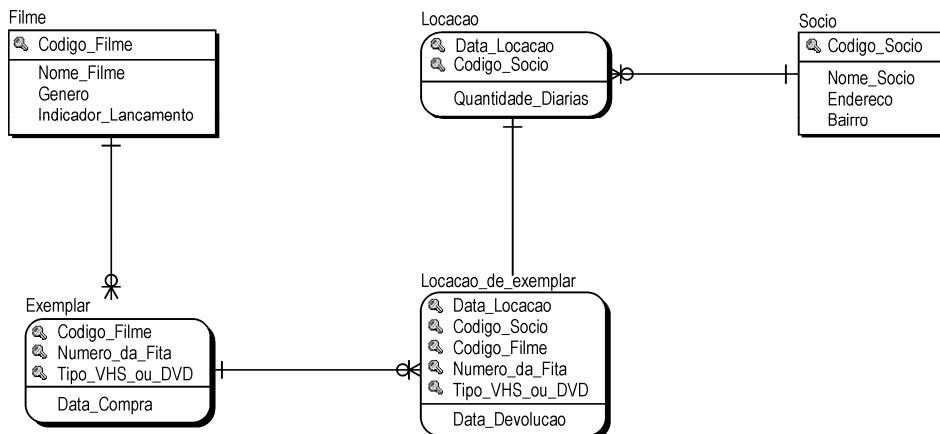
O modelo conceitual nunca deve ser construído com considerações sobre processos de negócio, com preocupações de acesso aos dados, não devendo existir nenhuma preocupação com o modo como serão realizadas as operações de consulta e manutenção dos dados nele apresentados. O foco deve ser dirigido sempre ao entendimento e à representação de uma realidade, de um contexto.

Modelo Lógico

Ele somente tem início após a criação do modelo conceitual, pois agora vamos considerar uma das abordagens possíveis da tecnologia Sistemas Gerenciadores de Banco de Dados (relacional, hierárquica, rede ou orientada a objetos) para estruturação e estabelecimento da lógica dos relacionamentos existentes entre os dados definidos no modelo conceitual.

A elaboração direta e um modelo lógico de dados, independentemente de já sabermos a abordagem para banco de dados, para a qual estamos realizando um projeto, levam à vinculação tecnológica de nosso raciocínio, perturbando a interpretação pura e simples de um contexto. Sempre que analisamos um contexto sob a óptica tecnológica, temos a tendência de sermos

técnicos demais, distorcer a realidade, conduzindo-a às restrições da tecnologia empregada, o que sempre, e já estatisticamente comprovado, leva a erros de interpretação da realidade, criando assim modelos de dados que não possuem aderência ao minimundo descrito.



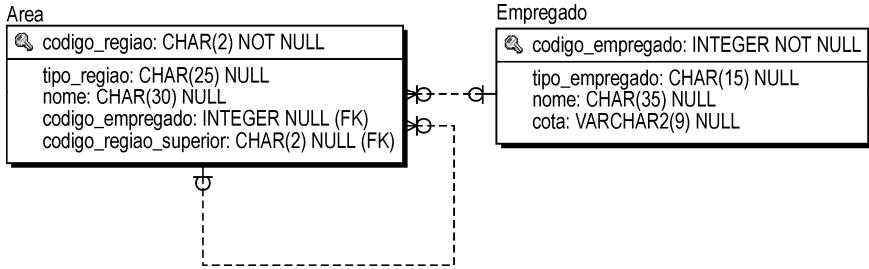
O modelo lógico descreve em formato as estruturas que estarão no banco de dados de acordo com as possibilidades permitidas pela sua abordagem, mas sem considerar, ainda, nenhuma característica específica de um Sistema Gerenciador de Banco de Dados (SGBD). Isso resulta um esquema lógico de dados sob a óptica de uma das abordagens citadas, pelo emprego de uma técnica de modelagem de dados orientada às restrições de cada abordagem.

Modelo Físico

O modelo físico será construído a partir do modelo lógico e descreve as estruturas físicas de armazenamento de dados, tais como:

- ▶ Tipo e tamanho de campos;
- ▶ Índices;
- ▶ Domínio de preenchimento desses campos;
- ▶ Nomenclaturas;
- ▶ Exigência de conteúdo;
- ▶ Gatilhos etc.

Elas são projetadas de acordo com os requisitos de processamento e uso mais econômico dos recursos computacionais. Esse modelo detalha o estudo dos métodos de acesso do SGBD para criação dos índices necessários para 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 sua montagem no dicionário de dados. Em ambiente de banco de dados relacional denominamos script de criação do banco de dados o conjunto de comandos em SQL (DDL) que será executado no Sistema Gerenciador de Banco de Dados para a criação de banco de dados correspondente ao modelo físico.

A seguir apresentamos um exemplo de script (conjunto de comando DDL) escrito em SQL para criação de um pequeno banco de dados:

```

CREATE TABLE Empregado (
    codigo_empregado      INTEGER NOT NULL,
    tipo_empregado         CHAR(15) NULL,
    nome                   CHAR(35) NULL,
    cota                  VARCHAR2(9) NULL,
    PRIMARY KEY (codigo_empregado);

CREATE UNIQUE INDEX XPKEmpregado ON Empregado
(
    codigo_empregado          ASC);

CREATE TABLE Area (
    codigo_regiao           CHAR(2) NOT NULL,
    tipo_regiao              CHAR(25) NULL,
    nome                     CHAR(30) NULL,
    codigo_empregado         INTEGER NULL,
    codigo_regiao_superior  CHAR(2) NULL,
    PRIMARY KEY (codigo_regiao),
    FOREIGN KEY (codigo_regiao_superior)
        REFERENCES Area,
    FOREIGN KEY (codigo_empregado)
        REFERENCES Empregado);

CREATE UNIQUE INDEX XPKArea ON Area
(codigo_regiao ASC);

CREATE INDEX XIFIArea ON Area
(codigo_empregado ASC);

CREATE INDEX XIF2Area ON Area
(codigo_regiao_superior ASC);

```

O Projeto de Banco de Dados

Todo projeto de sistema de banco de dados necessita de um foco, um centro nervoso. A modelagem de um banco de dados por meio da abordagem Entidade-Relacionamento representa esse 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 representação e entendimento dos dados que compõem a essência de um sistema.

O projeto de banco de dados para sistemas de aplicação, hoje em dia, não é mais uma tarefa a ser realizada somente por profissionais especializados em bancos de dados da área de desenvolvimento de sistemas, mas também por analistas de sistemas independentemente de experiência anterior em um determinado produto de banco de dados, pela utilização de técnicas estruturadas como a modelagem conceitual de dados Entidade-Relacionamento.

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

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

Desafortunadamente as metodologias de projeto de banco de dados, para sistemas de aplicação, apesar de já serem muito populares entre a comunidade técnica, não são utilizadas corretamente.

Em várias organizações os profissionais utilizam-se de pequenas técnicas pessoais, ou ainda pior, de uma inexistência completa de metodologia para esses projetos e com distorções na utilização das técnicas, sendo esta uma das maiores causas de falhas nos sistemas de informação desenvolvidos, mesmo com a existência de modelos de dados.

A utilização de abordagem correta de metodologia orientada a modelagem de 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 o momento para adequação do modelo construído à ponderação desses fatores.

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

Devido a essa não similaridade completa 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 ter uma metodologia estruturada que permita construir vários modelos, resultando o chamado metamodelo, ou seja, uma técnica específica de ampla abrangência e flexibilidade.

O metamodelo a ser utilizado deve ter a característica de poder modelar qualquer realidade, ter uma forma de trabalho bastante simples e características gráficas que sejam bem simples de construir e entender.

O nosso metamodelo é o modelo Entidade-Relacionamento (ER).

Como já destacamos, o modelo de dados é um plano para construir o banco de dados. Para ser efetivo, deve ser simples o bastante para comunicar ao usuário final a estrutura de dados requerida pelo banco de dados e detalhada o suficiente para criar a estrutura física de banco de dados.

O modelo Entidade-Relacionamento (ER) é o método mais comum para a construção de modelos de dados para bancos de dados relacionais.

O Modelo Entidade-Relacionamento (ER)

Ele foi proposto originalmente por Peter, em 1976 (CHEN, 1976), como um modo de unificar as visões de um banco de dados relacional, e teve como base a teoria relacional criada por E. F. Codd (1970).

Simplesmente declarado como modelo de ER, ele é conceitual e vê o mundo real como entidades e relacionamentos. Um componente básico do modelo é o diagrama de Entidade-Relacionamento, que visualmente representa objetos de dados.

Ao longo dos anos, vários estudiosos (Theorey, Fry, James Martin, entre outros) evoluíram e expandiram esse metamodelo.

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

Grande parte das extensões ao metamodelo baseia-se em alguns mecanismos de abstração: classificação, generalização e agregação.

O conceito de abstração, como já destacamos, 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 ER destinam-se prioritariamente ao projeto de banco de dados, entretanto eles podem ser utilizados para o entendimento de um determinado negócio, por exemplo, na modelagem de processos de negócios (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 ER é ainda a base para o desenvolvimento de sistemas orientados a objetos.

Somente o domínio das técnicas de modelagem não é suficiente para produzir bons modelos. O essencial na modelagem de dados é o total entendimento dos conceitos pertencentes à realidade em questão.

Ao modelar dados, ou qualquer outra coisa, o **objeto observado, o ambiente sob análise** será sempre o ponto de partida. Ele serve como referência para a construção de todo o

modelo. Por este motivo a fase de levantamento de dados, investigação e análise dos dados propriamente dita é tão importante.

O processo de modelagem consiste em cinco aspectos importantes:

- ▶ **Observação:** entrevistas, reuniões, questionários, análise de documentos aliados ao conhecimento e experiência prévios da área de negócios ou seu perfeito entendimento e compreensão.
- ▶ **Entendimento dos conceitos:** núcleo do processo de modelagem. Essa fase destina-se a identificar, conceituar, entender e assimilar o objeto observado.
- ▶ **Representação dos objetos:** aplicação de técnicas de modelagem de dados Entidade-Relacionamento.
- ▶ **Verificação de fidelidade e carências:** detectar falhas e anomalias, identificando respectivas causas que podem residir em conceitos malformados, pontos de vista equivocados, falha na concepção ou aplicação errada da técnica de representação.



- ▶ **Validações:** nessa fase busca-se a aprovação formal do modelo. Para que esse objetivo seja alcançado, é necessária a participação ativa do usuário final, bem como a visão de outros profissionais da área técnica (administrador de dados, analista de sistemas, administrador de banco de dados etc., de acordo com o caso). Esse processo deve ser rigoroso e crítico, tanto quanto possível.

As técnicas de modelagem de dados devem ser disseminadas à área usuária para que a validação não se transforme em um monólogo de analistas de sistemas e deixe de cumprir o seu verdadeiro objetivo (detectar falhas e anomalias), porém nunca tente fazer do usuário final um *expert* no assunto modelagem de dados.

A Implementação de Banco de Dados

Em síntese é a criação efetiva em um Sistema Gerenciador de Banco de Dados de uma estrutura de dados correspondente ao modelo físico derivado do modelo construído por meio das técnicas de modelagem Entidade-Relacionamento.

Muitas vezes ouvimos afirmações como: projeta-se o modelo Entidade-Relacionamento, mas o que se implementa é um conjunto de tabelas que muitas vezes não têm nada a ver com o modelo projetado.

Em suma isso até ocorre, **mas constitui um ERRO da administração de dados em conflito com a administração de banco de dados**, pois o modelo de tabelas, por assim dizer, implementado deve obrigatoriamente corresponder ao modelo lógico derivado do modelo conceitual projetado.

Em empresas são muitas as situações cujo modelo de dados é desprezado em decorrência de os projetistas não possuírem abrangência de aspectos físicos e operacionais de um Sistema Gerenciador de Banco de Dados no tocante à forma de recuperação dos dados nele inseridos, quanto à utilização de índices ou à disponibilidade de área de armazenamento e visão da população que vai ter um banco de dados. Isso faz com que o projetista de BD seja meramente conceitual, saltando-se desta forma o modelo lógico ou o conceitual, tratando o processo de modelagem em somente dois níveis:

- ▶ Modelo conceitual e físico, ou
- ▶ Modelo lógico e físico.

Isso provoca buracos de análise bem sensíveis que conduzem a uma implementação equivocada e de alto índice de manutenção das aplicações e do banco de dados.

Sejam quais forem os processos utilizados, completos ou parciais, devemos validar esse modelo antes de implementá-lo efetivamente, pela aplicação de conceitos e comandos de álgebra relacional na busca de recuperação das informações que nele estarão.

É fundamental para o projetista de banco de dados que ele possua capacitação na navegação do banco de dados proposto e implementado e, principalmente, compreensão do modelo, conhecendo seus caminhos de navegação. Apresentamos a álgebra relacional em capítulo específico adiante.

Quando falamos de álgebra relacional, destacamos o fator de importância que ela assume em relação ao projeto de banco de dados, destacando a seguinte afirmativa:

De que adianta projetar banco de dados se não sei como recuperar suas informações?

De que adianta saber comandos SQL se não sei como é a estrutura interna de um comando para recuperar dados de um conjunto de tabelas?

Foi comprovado, ao longo do tempo, que o domínio e o conhecimento de álgebra relacional dotam o analista de sistemas da capacidade de resolver qualquer tipo de consulta necessária à navegação e recuperação de situações, da mais simples à mais complexa exigida em uma aplicação de banco de dados.

Conclusão

São indiscutíveis as vantagens de trabalharmos hoje em dia com orientação a objetos, pelo alto índice de reutilização de código, pela simplicidade de criação de aplicações e pela alta velocidade de desenvolvimento, mas, outrossim, é também indiscutível a contínua necessidade de executar a modelagem de dados para que as duas técnicas se complementem na execução de sistemas eficazes que utilizem tecnologia Java, por exemplo, com um banco de dados relacional.

Como o ambiente tradicional em que são implementados os bancos de dados projetados é o de bancos de dados relacionais, vamos detalhar também a abordagem relacional de bancos de dados que, por mais escrita, lida e estudada, mantém a importância da aderência de seus conceitos para o processo de projeto de bancos de dados.

ABSTRAÇÃO EM MODELAGEM DE DADOS

Como já citamos no capítulo anterior, é preciso usar de abstração para construção de um modelo de conceitos sobre uma realidade em observação.

Mas como usar a abstração?

Este é um ponto em que vamos levar o leitor a executar exercícios que possibilitem um maior desenvolvimento dessa sua capacidade.

Repetindo a nossa afirmação, abstração ou a capacidade de abstração é um processo, digamos, mental, que usamos quando selecionamos várias características e propriedades de um conjunto de objetos ou fatos, e excluímos outras características que não são relevantes. Em outras palavras, aplicamos abstração quando nos concentramos nas propriedades de um conjunto de objetos ou coisas que consideramos essenciais, e desprezamos outras que não consideramos importantes.

No exemplo do capítulo anterior vimos como resultado de um processo de abstração uma bicicleta, da qual excluímos detalhes da estrutura, como os pedais, os freios, os mecanismos de tração e, inclusive, as possíveis diferenças entre várias bicicletas e associamos um nome à abstração. A figura da bicicleta era a representação dessa abstração.



Abstração em síntese nada mais é do que a visão, sem conceituações técnicas, que obtemos na mente de qualquer realidade que observamos no mundo real.

O que queremos buscar com a conceituação é a utilização mental desse processo de abstração, que é fundamental para um processo de análise de sistemas.

Não existe neste caso a desculpa de que não podemos gastar tempo no desenvolvimento de um projeto com técnicas de abstração, pois em realidade sempre o fazemos, porém com baixa intensidade e sem preocupação e destaque para isso. Podemos sim, e ainda devemos ter essa característica como relevante no processo de desenvolvimento, pois ela vai permitir que se obtenham todas as variantes que vão influenciar na criação de um modelo de dados, assim como a qualidade e a estética desses modelos também são fundamentais.

Existem três tipos de abstração que vamos utilizar em modelo conceitual de dados, ou melhor, para a construção de modelos conceituais de dados: classificação, agregação e generalização, como nos referimos no capítulo anterior. Lembre-se de que são tipos de abstração e nada têm ainda a ver com os conceitos de entidades e relacionamentos da técnica Entidade-Relacionamento de Peter Chen.

Classificação de Abstração

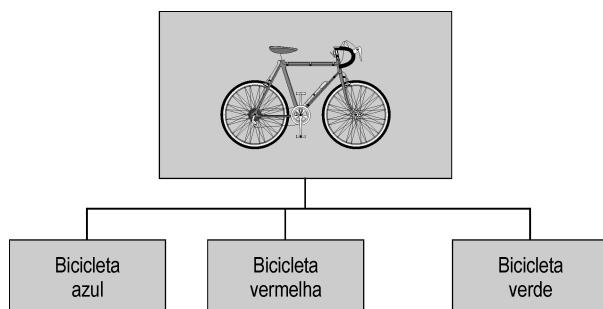
O processo chamado de classificação de abstração é o utilizado por nós para definir o conceito de uma classe de objetos ou coisas do mundo real caracterizadas por propriedades comuns.

Por exemplo, o conceito de bicicleta é uma classe de objetos cujos membros são todas as bicicletas semelhantes (bicicleta azul, vermelha, a minha bicicleta etc.). Similarmente o conceito de mês é uma classe cujos membros são janeiro, fevereiro, maio etc.

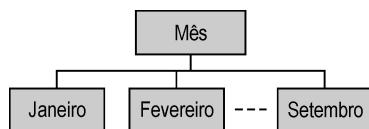
Quando pensamos em um mês, nos abstraímos nas características específicas de cada mês, por exemplo, o número de dias daquele mês, e ainda enfatizamos na mente aspectos comuns a todos os meses, como os dias predefinidos como o primeiro e o último dia do mês, que os meses são aproximadamente iguais em tamanho, 28 ou 31 dias.

Podemos representar a classificação graficamente com um nível maior com o nome da classe e um segundo nível com os elementos que pertencem a essa classe.

Nos exemplos referidos teremos para bicicleta:



Para mês:



Observe que este é um processo incondicional que realizamos mentalmente. Não está ligado ou relacionado a nenhuma tecnologia, porém é espontâneo do nosso processo de raciocínio e entendimento do universo como tal.

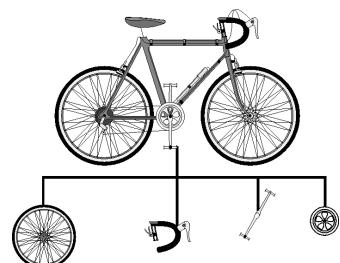
Agregação de Abstração

Agregar significa compor parte de alguma coisa.

O conceito de agregação de abstrações significa que podemos entender classes de coisas como elementos de composição de outra classe maior.

Algo como entender que um conjunto de várias classes compõe outra classe superior.

Vamos supor que as classes guidão, pedais, coroas e rodas sejam o nosso ponto inicial de análise, pois as encontramos ao analisarmos o universo de componentes de uma bicicleta.



Essas classes existem, pois temos vários tipos de coroa, de pedal etc., e essas classes vão compor o que denominamos classe bicicleta.

Devemos entender este conceito como um objeto que é composto de partes.

O processo de entendimento tanto pode ser realizado por decomposição de partes como pelo entendimento e compreensão inicial das partes até chegarmos ao todo.

Generalização de Abstração

A generalização de abstração ocorre quando definimos um subconjunto de relacionamentos entre elementos de duas ou mais classes. Por exemplo, a classe veículo é a generalização da classe bicicleta, pois toda bicicleta é um veículo.

Estamos generalizando um conceito que temos de uma realidade.

Logo, veículo é uma generalização da classe de objetos bicicleta.

Como visualizar isso em coisas mais simples?

Vamos olhar a cozinha em casa. Neste livro vamos falar muito do armário de cozinha com alimentos e produtos de limpeza.

Imagine você entrar na sua cozinha e olhar para o armário da despensa.

Quais os conceitos do que você observa? Alimentos, materiais de limpeza, panelas?

Epa! Temos já duas classes no mínimo: classe alimentos e classe materiais de limpeza.

Veja que seu primeiro nível de abstração é de generalização.

Você identifica objetos como alimentos e materiais de limpeza.

Mentalmente distinguimos e relacionamos componentes e participantes dessas duas classes. Temos então um processo de classificação e generalização de coisas abstratas até o momento: alimentos e materiais de limpeza.

Alimento é uma classe de objetos e produto de limpeza é outra classe de objetos.

Por que então um processo de generalização neste caso?

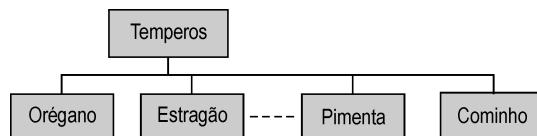
Analizando os objetos da classe alimentos, vamos perceber que há classes menores dentro dela, por exemplo, líquidos e temperos.

Ora, amigos, na classe líquido temos óleos, azeites, vinagres etc.

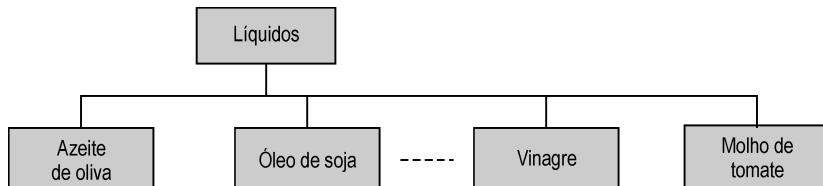
Em temperos podemos ter pimenta, caldo de carne, orégano etc.

Graficamente existe a demonstração inicialmente da classificação:

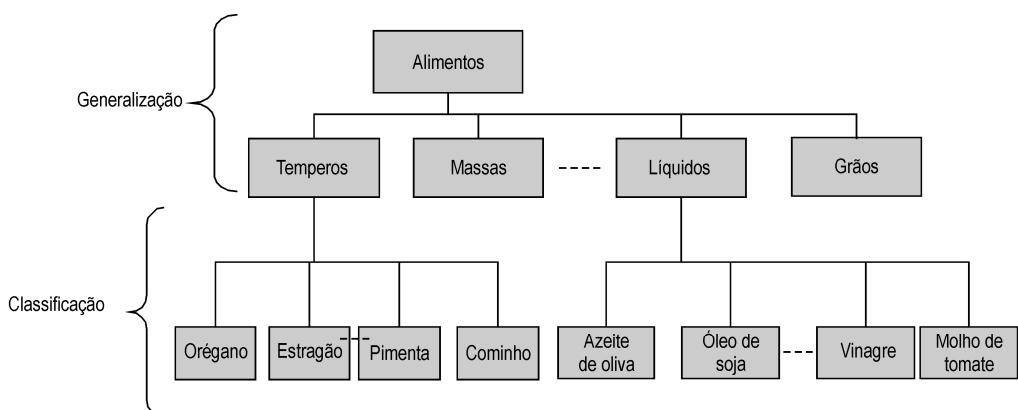
Para a classe temperos:



Para a classe líquidos:



Agora vamos entender graficamente a generalização.



Observe que no momento em que realizamos uma segunda ordenação das classes que possuem várias coisas, estamos generalizando ou classificando em um nível de hierarquia. Isto é generalizar.

Dizer que temperos, líquidos, massas e grãos são alimentos é generalizar as classes temperos e companhia em alimentos.

A Cozinha

Como exercício você deve realizar a classificação completa dos produtos de limpeza, pois certamente encontrará os dois processos.

Para obter um processo de agregação, experimente abstrair em relação a uma feijoada, que pertence à classe de alimentos preparados. Imagine aquela feijoada enlatada ou a que você come aos sábados.

Provavelmente, para os que leem e concluem até agora isto é óbvio.

Mas, em realidade, na hora em que fazemos análise de sistemas não utilizamos metodicamente esse processo de abstração, principalmente na análise de dados. Se utilizássemos, obteríamos certamente modelos mais consistentes com a realidade a que se destinam ou a qual devem refletir.

O processo comum nos seres humanos ao analisarem um ambiente, uma pequena porção do mundo real, é de interpretá-lo por meio de uma agregação.

Inconscientemente realizamos sempre um processo de entendimento em um mais alto nível, o de conceitos agragados.

Vejamos ainda o caso da cozinha.

Perguntando a um aluno quais objetos, conceitos ele obtinha ao observar uma cozinha, a resposta imediata foi eletrodoméstico.

Eletrodoméstico é uma classe de objetos, um conceito de algo em uma realidade, entretanto é uma generalização ou uma classificação?

Se considerarmos que a cozinha já tem um conceito de utensílios de cozinha, existem mais conceitos em uma cozinha a serem explorados. Eletrodoméstico faz parte de utensílios de cozinha, é uma parte de utensílios de cozinha, logo agregamos essa abstração em relação ao conceito de utensílios de cozinha.

Mas eletrodoméstico pode ser um liquidificador, um forno micro-ondas, uma geladeira, um fogão.

Podemos ainda lembrar que panelas e talheres são utensílios de cozinha.

Como organizar esse conjunto imenso de conceitos abstratos?

Bem, já sabemos que eletrodoméstico é uma classe de conceitos.

Eletrodomésticos podem ser de vários tipos: liquidificador, forno micro-ondas, batedeira, torradeira etc.



Mas existe uma classe acima na hierarquia de interpretação, que é a classe utensílios de cozinha.

Se observarmos, veremos que mais coisas existem numa cozinha. Talheres, alimentos, materiais de limpeza e panelas, por exemplo.

A nossa análise inicial somente se deteve nos alimentos, classificando-os em temperos e líquidos.

Mas devemos realizar uma observação e um esforço no sentido de refinar e classificar o melhor possível o conceito de alimento, pois cada elemento do armário, por exemplo, pode ter uma classificação diferenciada dentro da classe alimento.



A figura serve para lembrá-lo de que temos frutas, verduras, legumes, açúcar, pães etc. Logo, devemos eliminar a tendência generalista e simplesmente parar a nossa análise após a obtenção de um conceito simples, pois o perfeito entendimento de uma realidade é composto da definição clara de todos os objetos que o realizam.

Para utilizar a modelagem conceitual de dados com a técnica de entidades e relacionamentos, necessitamos utilizar ao máximo os conceitos de abstração. Somente abstraindo o nosso entendimento de uma realidade é que 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 procedimentos ou fluxo de dados de processos existentes.

No livro Projeto de Banco de Dados: Uma Visão Prática, publicado pela Editora Érica, afirmamos que modelagem de dados é como a arte fotográfica. Prepara-se a câmera e tira-se a foto, sem se importar com os movimentos, mas cabe destacar que essa foto é interpretada pela nossa capacidade de abstração, e nela aplicamos os conceitos apresentados.

Entretanto, o esquema produzido pelo trabalho de modelagem de dados possibilita a visualização dos fatos que ocorrem, fornece uma noção clara de movimento, dinâmica de

negócios e permite que realizemos a abstração das atividades e procedimentos que podem ser exercidos sobre eles.

Começar por Dados

A teoria de que a análise de um sistema deve começar sempre por dados se justifica pela estabilidade que os dados possuem para representar qualquer realidade.

Em segundo lugar é fato evidente que não conseguimos entender nenhum procedimento sem os dados.

Vamos voltar à cozinha para demonstrar.

Como preparar um prato, cozinhar se você não tiver a receita do prato? Você precisa de dados para executar um processo, para entender um procedimento.

Para executar, preparar um prato, você precisa dos ingredientes obrigatoriamente. A execução de uma receita culinária nada mais é do que a sequência em que você mistura e acrescenta os ingredientes, com um dado importante sempre que é o tempo de execução.

Como você pode entender o que é cozinhar um prato se não souber os seus dados principais, que são os ingredientes? Simplesmente impossível!

Agregações Binárias

Agregações binárias, segundo Batini, Navathe e Ceri, no livro *Conceptual Database Design*, são o mapeamento estabelecido entre duas classes de coisas, ou objetos que visualizamos através de abstração.

Por exemplo, motorista representa a agregação binária de uma classe pessoas e de uma classe veículos.

Elas existem quando se estabelece uma correspondência entre os elementos de duas classes. Obviamente, interpretamos esse mapeamento como um fato, o fato de uma pessoa dirigir um veículo. Novamente é preciso destacar que também utilizamos abstração para fatos e acontecimentos.

Esse tipo de abstração é que nos permite entender as relações entre os objetos e fatos do mundo real.

Vamos expandir um pouco mais nossos exemplos.



Considerando a realidade comum de moradia, um fato, um negócio ou um acontecimento, como querer neste primeiro momento, qual conjunto de coisas nós podemos abstrair e classificar nesta realidade?

Existem pessoas que moram em imóveis, assim como existem pessoas que são os proprietários de imóveis em que vamos morar ou que vamos alugar.

Abstraindo então temos um conjunto de coisas agrupadas que formam a classe imóvel, e temos outro conjunto de coisas que denominamos pessoa, uma outra classe de objetos com vários elementos cada um.

Ao realizarmos uma abstração binária, ou melhor, uma análise binária, obteremos os fatos que retratam a correspondência ou relação existente entre elementos dos objetos em análise.

- ▶ Pessoa é proprietária de imóvel.
- ▶ Pessoa é morador ou locador de imóvel.

Temos duas classes mais neste caso: proprietários e moradores, e isso caracteriza uma abstração binária, pois essas classes somente existem como resultado da associação, da correspondência entre elementos das outras duas classes. Somente existem em função de fatos que ocorrem no mundo real.



Genericamente podemos afirmar que, em realidade, em um imóvel podem morar diversas pessoas no decorrer do tempo, porém somente uma pessoa é proprietária do imóvel.

Esta observação relata o que se chama de cardinalidade do mapeamento entre as classes. Essa cardinalidade descreve a forma como as coisas, os elementos de um grupo ou classe se correspondem ou se relacionam com os elementos da outra classe ou grupo de coisas.

Cardinalidade Mínima

Considerando a relação entre duas classes de objetos denominadas C1 e C2, cardinalidade mínima de C1 nesta relação é o menor número de vezes que cada elemento de C1 pode participar da relação.

Vejamos no nosso exemplo de pessoas e imóveis.

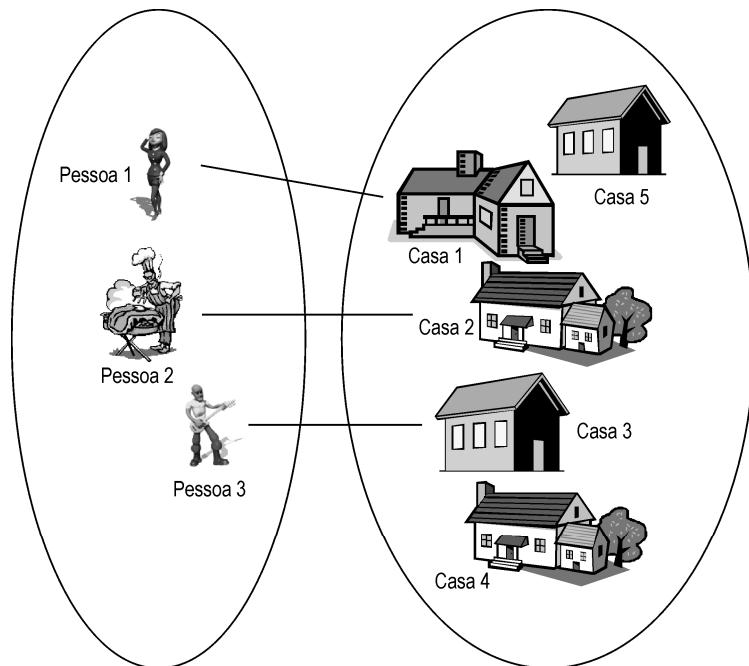
Consideremos as agregações binárias de proprietário e morador entre pessoas e imóveis.

Se nós considerarmos que cada pessoa mora em somente um imóvel, a cardinalidade mínima é igual a 1 (um), considerando-se a leitura dessa relação no sentido da classe pessoas e a agregação binária aluga e a classe imóveis.

Ou seja, uma pessoa aluga somente um imóvel.

Se nós considerarmos que muitos imóveis podem não estar sendo habitados, ou seja, não mora ninguém neles, então a cardinalidade mínima é 0 (zero), considerando-se a classe imóvel e a agregação binária aluga.

Na realidade significa que um imóvel pode estar alugado ou estar vazio, desabitado.

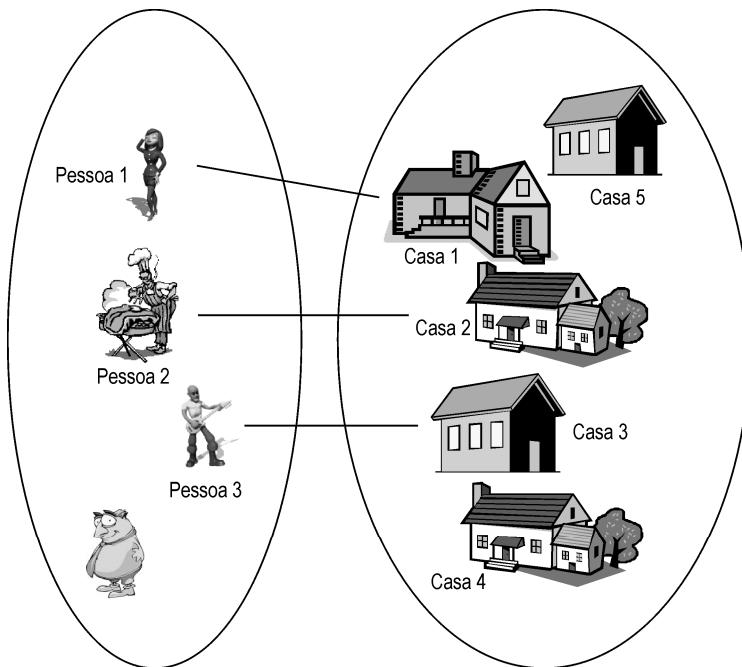


Se analisarmos agora a relação entre a classe pessoas e a agregação binária proprietário, ou seja, o mapeamento de elementos de pessoas e imóveis na agregação binária proprietário, temos a figura em seguida.

Se analisarmos a relação olhando por pessoas, podemos afirmar que muitos elementos de pessoa não são proprietários de nenhum imóvel, logo a cardinalidade mínima é 0 (zero).

Se analisarmos a relação olhando imóveis em relação à pessoa, a agregação proprietário vai nos apresentar uma cardinalidade mínima de 1, pois todo imóvel tem no mínimo um proprietário.

O importante é neste momento de abstração analisarmos a correspondência, a relação entre as duas classes de objetos em dois sentidos para que seja possível obter o entendimento correto da realidade que estamos observando e modelando.



Cardinalidade Máxima

Assim como analisamos e registramos a cardinalidade mínima, devemos analisar a cardinalidade máxima em cada uma dessas relações.

Se considerarmos que cada pessoa pode morar em vários imóveis, a cardinalidade máxima em realidade é infinita, ou não tem limites.

Se assumirmos que cada imóvel pode ter vários moradores, a cardinalidade máxima também é infinita.

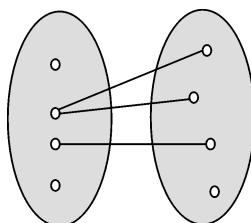
No sentido de imóvel podemos afirmar que cada imóvel pode ter muitos proprietários, logo a cardinalidade máxima é também infinita, ou seja, representada por $= n$.

Porém, se considerarmos que cada imóvel tem no mínimo obrigatoriamente 1 (um) proprietário e somente pode pertencer a uma pessoa, então a cardinalidade máxima também é 1 (um).

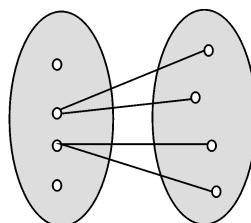
Esta exemplificação mostra que os valores importantes de cardinalidade máxima são 1 e n respectivamente; n representa qualquer número e indica que cada elemento de uma classe possui um grande número de relações com outra classe de objetos.

Se a cardinalidade mínima é igual a 1 e a cardinalidade máxima também é igual a 1, dizemos que essa agregação binária é de **um-para-um**. Se a agregação possui uma cardinalidade mínima = 1 e uma cardinalidade máxima igual a n , dizemos que é uma agregação de **um-para-muitos**. Finalmente, se a agregação possui ambas as cardinalidades iguais a n , dizemos que é de **muitos-para-muitos**.

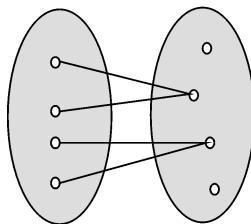
Caro leitor, analise qualquer situação em que você se abstraia e classifique classes de coisas. Logo após estude as classes de relação que existem entre essas classes. A partir do momento em que estabeleceremos uma relação entre os dois conjuntos de objetos ou coisas, podemos então determinar por meio de uma análise criteriosa a cardinalidade existente nessa relação.



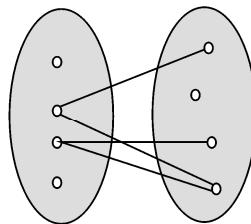
Um-para-um



Um-para-muitos



Muitos-para-um



Muitos-para-muitos

Exercícios de Abstração

1. Observe agora o seu ambiente de trabalho, seu escritório. Utilizando a abstração, execute a identificação dos conceitos que existem no ambiente, dos objetos e suas classificações e agregações.
2. Identifique o contexto de uma videolocadora XXX, uma empresa prestadora de serviços com sede no Rio de Janeiro, cuja principal finalidade é a locação de fitas de vídeo, DVDs e games. Com o intuito de melhor atender a seus clientes oferece alguns outros tipos de serviço, tais como:
 - a. Reserva de fitas/DVDs/games com antecedência de um dia.
 - b. Entrega de fitas/DVDs/games locados cujo número exceda três unidades.
 - c. Venda de fitas de vídeo/DVDs/games.
 - d. Encomenda de fitas/DVDs/games para venda.
 - e. Venda de produtos de consumo diferenciados, tais como balas, chocolates, pipoca, refrigerantes, artigos e acessórios para vídeo, som e imagem.

3. O setor de compras de uma empresa possui uma série de atividades, mas vamos exercitar a capacidade de abstração na área de controle de solicitações de compra a fornecedores. A seguir há um descritivo das principais atividades. Capte delas os objetos, classificações e agregações possíveis.
- ▶ A área recebe as solicitações de reposição ou compra de produtos.
 - ▶ Identifica se é um produto regular de estoques ou se é uma compra de produto específico, e se existirem solicitações do mesmo produto, unifica a necessidade totalizando as quantidades solicitadas.
 - ▶ Verifica em planilhas os fornecedores que disponibilizam cada produto e realiza cotação de preços para a solicitação. Se não existir ocorrência anterior de fornecimento, busca no mercado os fornecedores que possuem o produto solicitado e realiza uma cotação de preços.
 - ▶ Elabora uma planilha de cotações por fornecedor, produto e preços, efetua os cálculos de custos por produto/fornecedor e escolhe o fornecedor.
 - ▶ Ao fornecedor escolhido é dada uma ordem de compra de produtos especificando produto, quantidades, preço, prazos de pagamento e condições gerais de entrega.
 - ▶ As compras são acompanhadas até o momento da conferência de recebimento dos produtos adquiridos, comparando os dados da nota fiscal do fornecedor com os dados e condições da ordem de compra.
 - ▶ Após o recebimento são atualizadas as planilhas de compras realizadas, mantendo um histórico de desempenho de fornecedores quanto a prazos de entrega e condições de preço e pagamentos, além de informações sobre o controle de qualidade dos produtos adquiridos.
 - ▶ Observe que não são muitos os objetos existentes neste contexto, e sim mais processos que permitirão ao leitor identificar as relações entre as classes encontradas e suas cardinalidade.

BANCOS DE DADOS RELACIONAIS

Introdução

Este livro define os conceitos básicos de modelo de dados Entidade-Relacionamento, aplicados à análise de sistemas de informação, e apresenta os conceitos originais realizados por Peter Chen e estendidos por outros autores.



A utilização de modelagem de dados nos trabalhos de desenvolvimento de sistemas de informação cresceu muito nos últimos anos, entretanto esses trabalhos continuam dirigidos a aspectos físicos dos Sistemas Gerenciadores de Bancos de Dados, tais como índices e performance, provocando uma aceleração no processo de desenvolvimento que leva o analista de sistemas a criar estruturas de dados que não refletem nenhuma realidade de forma clara, criando bancos de dados inchados, e principalmente instáveis, contrariando alguns dos preceitos lançados por Codd, criador da Teoria Relacional.

O livro destina-se a projetar aplicações de Tecnologia da Informação orientadas a bancos de dados relacionais. Logo, para que isso se concretize, é preciso que haja um entendimento correto de como são realizadas as operações lógicas sobre os dados de um banco de dados relacional, assim como estão organizados esses dados nessas estruturas lógicas.

Criado por Edgar F. Codd, nos anos de 1970, 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 relações matemáticas e que estão representadas de maneira uniforme com o uso de tabelas bidimensionais.

Este princípio coloca os dados direcionados a estruturas mais simples de armazenamento de dados, que são as tabelas, e nas quais a visão do usuário é privilegiada.

É comum observarmos analistas desenhandos modelos de dados sem preocupação ou conhecimento lógico dos processos de recuperação e navegação em um banco de dados relacional, assim como não dedicando atenção suficiente à lógica por trás dos conceitos de chaves primárias e estrangeiras.

Observe que é uma concepção lógica. Claro que é inerente a uma tecnologia específica, a dos bancos de dados relacionais, porém são as concepções lógicas lançadas por Edgard Codd que norteiam a Teoria Relacional, em um formato lógico, que posteriormente foi implementado nas tecnologias dos Sistemas Gerenciadores de Bancos de Dados Relacionais.

Teoria Relacional

Agora vamos estudar os conceitos lógicos de banco de dados relacional, que ajudarão a entender e navegar em dados existentes em bancos dados relacionais.

Em 1970, Edgar F. Codd ofereceu-nos uma colaboração revolucionária ao formular o modelo de dados relacional. Em 1979, Codd e também Chris Date refinaram o modelo relacional, chegando ao que se denominou modelo relacional estendido.

A abordagem relacional representa uma forma de descrever o banco de dados através de conceitos matemáticos simples: a teoria dos conjuntos.

Voltada, principalmente, a melhorar a visão dos dados pelos usuários, a abordagem relacional faz com que os usuários vejam o banco de dados como um conjunto de tabelas bidimensionais, originadas em **linhas e colunas**.

Definição Clássica

São conjuntos de dados vistos segundo um conjunto de TABELAS, e as operações que as utilizam são feitas por linguagens que o manipulam, não sendo procedurais, ou seja, manipulando conjuntos de uma só vez.

O conceito principal vem da teoria dos conjuntos atrelado à concepção de que não é relevante ao usuário saber **onde os dados estão nem como os dados estão** (transparência).

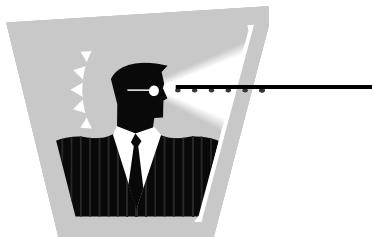


Tabela de Clientes

CodCliente	NomCliente	RuaCliente	CidadeCliente
1	Luis Sampaio	Rua A	Rio de Janeiro
2	Carlos Pereira	Rua B	Niterói
3	José Alves	Rua C	Rio de Janeiro
4	Luis Paulo Souza	Rua D	Niterói

Os usuários manipulam estes objetos dispostos em linhas e colunas das tabelas que possuem informações sobre o mesmo assunto de forma estruturada e organizada.

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDeptº
101	Luis Sampaio	10/08/2003	M	C3	D5
104	Carlos Pereira	02/03/2004	M	C4	D6
134	José Alves	23/05/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
195	Marta Silveira	05/01/2002	F	C1	D5
139	Ana Luiza Magalhães	12/01/2003	F	C4	D6
123	Pedro Sergio Doto	29/06/2003	M	CC7	D3
148	Larissa Silva	01/06/2002	F	C4	D6
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/02/2000	M	C2	D4

CdCargo	NumCargo	VlrSalario
C1	Aux Vendas	350,00
C3	Vendedor	800,00
C7	Diretor	2500,00
C2	Vigia	400,00
C5	Gerente	1000,00
C4	Aux Cobrança	250,00

Para lidar com estes objetos, o usuário conta com um conjunto de operadores e funções de alto nível, constantes na álgebra relacional.

A Teoria Relacional possui premissas que definem uma tabela de dados:

- ▶ Cada uma das tabelas é chamada de relação.
- ▶ O conjunto de uma linha e suas colunas é chamado de tupla.
- ▶ Cada coluna dessa tabela tem um nome e representa um domínio da tabela.
- ▶ **A ordem das linhas é irrelevante.**
- ▶ Não há duas linhas iguais.
- ▶ Usamos nomes para fazer referência às colunas.
- ▶ **A ordem das colunas também é irrelevante.**
- ▶ Cada tabela tem um nome próprio, distinto de qualquer outra tabela no banco de dados.

Vamos analisar algumas tabelas de dados para entendermos como estão organizados os dados em um banco de dados relacional.

É importante que exercitemos a mente para enxergar dados dentro de uma tabela relacional sempre que nos referirmos ao nome dela, e não somente a estrutura de seus dados. Por exemplo, uma tabela de CDs para o nosso microcomputador:

Tabela de CDs

Número do CD	Data da Gravação	Título do Conteúdo	Responsável	Local onde está guardado
1	24/1/2001	Clipart	Samir	Estojo Verde
3	13/2/2000	IRRF 2000	Felipe	Caixa Docs
2	14/12/2000	Backup textos	Felipe	Estojo Azul
4	25/1/2000	Fotos Gramado	Samir	Cx. Álbum 3

Este livro utiliza como nomenclatura o que existe mais tradicionalmente, que é chamarmos de tabelas e não de relações, assim como utiliza linhas em vez de tuplas.

Mas observe nesta simples tabela de CDs que os conceitos listados anteriormente em relação às tabelas relacionais estão destacados:

- ▶ A ordem das linhas é irrelevante, pois o CD número 2 vem após o CD de número 3. Se observar mais detalhadamente, você verá que o CD número 3 tem uma data inclusive menor que o CD número 2.
- ▶ Nenhuma linha se repete nesta tabela.
- ▶ A ordem das colunas também não tem nenhum destaque, ou importância, pois não estão em nenhuma ordem lógica.
- ▶ Todas as colunas têm nome, que identifica o seu conteúdo, ou melhor, o significado do valor de seu conteúdo.

Vamos observar outra tabela para fixar este entendimento:

Tabela de Funcionários, a clássica de exemplos

Nome	Sexo	Matrícula	Departamento	Cargo	Salário
João Carlos	Masc.	373	TI - Operações	Operador	3.000,00
Carlos Brito	Masc.	872	TI - Programação	Programador I	3.500,00
Silvia Moraes	Fem.	963	TI - Análise	Analista Sist II	5.500,00
Cláudia Tereza	Fem.	161	TI - Gerência	Secretária	1.500,00
Pedro Julio	Masc.	292	R. H.	Diretor	6.000,00
Pedro Julio	Masc	574	TI - Análise	Analista Sist I	4.500,00

Vamos analisar estes dados, já que parece que todos têm mais facilidade de entender esta tabela, talvez por já terem visto em tantos e tantos livros.

A coluna Matrícula não indica nenhuma ordem para as linhas, confirmando o conceito de que a ordem das linhas é irrelevante.

Todas as colunas têm um nome referente ao assunto, o que é bem evidente.

A disposição das colunas não tem nenhuma finalidade de classificação, tampouco indica ordem de leitura dos dados.

Observe o exemplo seguinte em que modificamos a ordem das colunas, sem prejuízo nenhum do assunto ou da representatividade da tabela.

Tabela Funcionários

Salário	Nome	Sexo	Matrícula	Departamento	Cargo
3.000,00	João Carlos	Masc.	373	TI -Operações	Operador
3.500,00	Carlos Brito	Masc.	872	TI - Programação	Programador I
5.500,00	Silvia Moraes	Fem.	963	TI - Análise	Analista Sist II
1.500,00	Cláudia Tereza	Fem.	161	TI - Gerência	Secretária
6.000,00	Pedro Julio	Masc.	292	R. H.	Diretor
4.500,00	Pedro Julio	Masc	574	TI - Análise	Analista Sist I

O importante é que toda vez que nos referirmos à tabela funcionários visualizemos os dados que ela contém, e simulando valores possíveis, pois assim vamos mentalizando e nos abstraindo em algum objeto que a tabela representa.

Características Principais de uma Relação (Tabela Relacional)

Vamos definir a concepção técnica de uma relação (tabela).

NumReg	NomeFunc	DtAdmissão			Sexo	Telefone	CdDept
		Dia	Mês	Ano			
101	Luis Sampaio	10	08	2003	M	2565-8974	D5
104	Carlos Pereira	02	03	2004	M	3131-4649	D6
134	José Alves	23	05	2002	M	2386-8897	D1
121	Luis Paulo Souza	10	12	2001	M	2241-5896	D5
195	Marta Silveira	05	01	2002	F	5693-521 / 5694-8523 / 5694-852	D5
139	Ana Luiza Magalhães	12	01	2003	F	4545-8899	D6
123	Pedro Sergio Doto	29	06	2003	M	4296-8853	D3
148	Larissa Silva	01	06	2002	F	4289-9675	D6
115	Roberto Fernandes	15	10	2003	M	2685-8132	D5
22	Sergio Nogueira	10	20	2000	M	2594-3122	D4

↑
↑
↑

Atributo composto
Atributo multivalorado

Todos os atributos (colunas) de uma relação devem ser atômicos, isto é, indivisíveis em termos de valores e componentes.

Isso quer dizer que não existem colunas do tipo subgrupo, todas são itens elementares não subdivididos em nenhuma hipótese E também não é permitida a existência da múltipla ocorrência de valores (multivaloração) em nenhum de seus atributos (colunas). Observe o exemplo anterior.

É importante a compreensão de que cada linha de uma tabela representa um objeto, um assunto que é descrito pelos valores de cada uma dessas colunas.

O esquema de uma relação, ou seja, a sua definição, pode ser interpretada como uma declaração, ou um tipo de afirmação.

O exemplo de uma tabela funcionário apresenta:

- ▶ Número de Registro (NUMREG), Nome do Funcionário (NOMEFUNC), Data de Admissão (DTADMISSÃO), Sexo (SEXO), Telefone (TELEFONE) e Departamento (CDDEPTO).
- ▶ Cada tupla (linha) da relação (tabela) deve ser interpretada como um fato ou uma ocorrência particular dessa afirmação.

Fato	NumReg	NomeFunc	DtAdmissão	Sexo	Telefone	CdDept
Existe um funcionário de nome Luis Sampaio de NumReg="101, com data de admissão em '10/08/2003', do sexo 'masculino', com telefone '2565-8974 e CdDept = "D5".	→ 101	Luis Sampaio	10/08/2003	M	2565-8974	D5
	104	Carlos Pereira	02/03/2004	M	3131-4649	D6
	134	José Alves	23/05/2002	M	2386-8897	D1
	121	Luis Paulo Souza	10/12/2001	M	2241-5896	D5
	195	Marta Silveira	05/01/2002	F	5693-521	D5
	139	Ana Luiza Magalhães	12/01/2003	F	4545-8899	D6
	123	Pedro Sergio Doto	29/06/2003	M	4296-8853	D3
	148	Larissa Silva	01/06/2002	F	4289-9675	D6
	115	Roberto Fernandes	15/10/2003	M	2685-8132	D5
	22	Sergio Nogueira	10/02/2000	M	2594-3122	D4

Domínio

Representa o conjunto de valores atômicos admissíveis de um componente (coluna) de uma relação (tabela).

Exemplo:

- ▶ Telefone: conjunto de 10 dígitos.
- ▶ CPF: conjunto de 7 dígitos.
- ▶ Idade_Empregado: $16 \leq \text{idade} \leq 70$.
- ▶ Departamentos: conjunto de departamentos de uma empresa.

A cada domínio está associado um tipo de dados ou formato.

Exemplo:

- ▶ Telefone (ddd) dddd-dddd em que $d = \{0,1,2,\dots,9\}$.
- ▶ IdadeEmpregado: número inteiro entre 16 e 70.

Um esquema de uma tabela R, definida por $R(A_1, A_2, \dots, A_n)$, é um conjunto de atributos do tipo $R = \{A_1, A_2, \dots, A_n\}$.

Cada atributo A_i é o nome de um papel realizado por algum domínio D na tabela R, ou seja, o nome de uma coluna na tabela R.

Restrições de domínios são estabelecidas determinando-se domínios de valores para cada coluna de uma tabela. Normalmente são estabelecidos e definidos os valores que uma coluna de uma tabela pode ter, isto é, o domínio da coluna.

Permitem, por exemplo:

- ▶ Verificar os valores inseridos em um banco de dados.
- ▶ Testar consultas para garantir que as comparações tenham sentido.
- ▶ Em geral o domínio é especificado por tipos primitivos de dados, tais como **integer**, **float**, **char**, **date**, **time**, **money** etc.

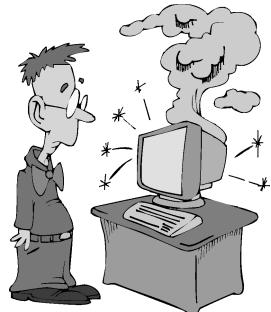
Também podem ser descritos pela definição de subconjuntos de tipos primitivos ou de listas enumeradas, ou seja, lista de valores possíveis de existir na coluna.

Observando os conceitos até agora descritos, destacamos mais um detalhe:

Nenhuma linha se repete também nesta tabela. Concorda?

Poderia o leitor questionar: posso ter duas pessoas com o mesmo nome?

Claro que sim. É mais comum ocorrer do que você possa imaginar, mas como eles têm matrículas na empresa diferentes, as linhas da tabela são obviamente diferentes.



Se o leitor for atuar como analista em órgãos públicos em nosso país, vai encontrar situações deste tipo: o mesmo servidor com duas matrículas. Isso parece uma anormalidade, porém é fato corriqueiro em administração pública.

Se cada linha da tabela representa um fato, não faz sentido descrevermos o mesmo fato mais de uma vez. Motivo conceitual básico para definir que não existem duas linhas iguais em uma mesma tabela.

Que outras características é importante entender e utilizar em uma tabela relacional?

Por exemplo, como garantir que as linhas não se repitam em uma tabela relacional?

Essa garantia é conseguida pela propriedade e conceito de chave primária.

Mas, efetivamente, o que é uma chave primária?

Chave Primária

Em toda e qualquer tabela existente em um banco de dados relacional haverá sempre uma coluna ou um conjunto de colunas concatenadas, cujos valores são únicos na tabela, isto é, nunca se repete aquele valor em nenhuma outra linha da tabela.

Essa coluna ou conjunto de colunas concatenadas identifica uma única linha da tabela. Então dizemos que essa coluna ou conjunto de colunas forma a **chave primária** da tabela.

Observe que falamos em uma coluna ou conjunto de colunas concatenadas.

O que analisamos é o valor de cada coluna ou desse conjunto para que não se repitam, ou não possam se repetir.



Ao definirmos uma chave primária, se não observarmos corretamente, podemos definir erradamente a chave primária de uma tabela.

Na tabela anterior que apresentamos de funcionários ainda não havíamos definido a sua chave primária, o que vamos efetivar neste ponto:

Olhe a tabela:

Nome	Sexo	Matrícula	Departamento	Cargo	Salário
João Carlos	Masc.	373	TI -Operações	Operador	3.000,00
Carlos Brito	Masc.	872	TI -Programação	Programador I	3.500,00
Silvia Moraes	Fem.	963	TI - Análise	Analista Sist II	5.500,00
Cláudia Tereza	Fem.	161	TI - Gerencia	Secretária	1.500,00
Pedro Julio	Masc.	292	R. H.	Diretor	6.000,00
Pedro Julio	Masc	574	TI - Análise	Analista Sist I	4.500,00

Qual coluna ou conjunto de colunas que, concatenadas, formam um identificador único para cada linha desta tabela?

Nome? Com certeza não, pois se repete. Sexo? Idem, se repete.

Departamento também não, e cargo e salário sozinhos também não fazem nada.

Sobra neste caso a única coluna que não tem valores repetidos, que é matrícula.

Matrícula
373
872
963
161
292
574

Existe somente um valor de matrícula para cada linha, que não se repete, logo podemos determinar que matrícula é a chave primária da tabela funcionários.

Vamos tentar observar uma tabela mais complicadinho, em que tenhamos de realizar um estudo maior para determinar a chave primária.

Vamos imaginar que temos uma tabela que apresenta registros de consumo de bebidas em um hotel.

Tabela de Consumo de Bebidas

Bebida	Qtde	Valor Unitário	Local Consumo	Quarto	Data Consumo	Hora Consumo	Valor Total
Cerveja Long Neck	2	3	Restaurante	101	22/01/2001	14:30	6.00
Chopp	3	2	Bar	203	19/01/2001	11:00	8.00
Cerveja Long Neck	2	3	Restaurante	101	23/01/2001	14:30	6.00
Refrig. Lata	3	2	Restaurante	203	20/01/2001	05:45	6.00
Mate	1	1,5	Frigobar	407	21/01/2001	16:30	1.50
Café	1	0,8	Restaurante	203	18/01/2001	08:00	0.80
Suco	1	3	Service Room	505	22/01/2001	21:30	3.00
Mate	1	1,5	Restaurante	203	21/01/2001	16:30	1.50
Chopp	4	2	Bar	101	19/01/2001	17:10	8.00
Água Min.	1	1	Frigobar	203	18/01/2001	08:30	1.00
Café	1	0,8	Restaurante		18/01/2001	18:00	0.80

Observe com atenção esta tabela.

Que coluna ou conjunto de colunas poderíamos utilizar para definir como identificador único de cada linha?

Poderíamos utilizar, por exemplo, a concatenação das colunas Bebida e Quarto do hotel?

Se observarmos com atenção, veremos que quando existe um segundo consumo da mesma bebida no mesmo bar, perde-se a característica de identificador único, pois o valor, por exemplo, Cerveja Long Neck_101, se repete.

Bebida	Qtde	Valor Unitário	Local Consumo	Quarto	Data Consumo	Hora Consumo	Valor Total
Cerveja Long Neck	2	3	Restaurante	101	22/1/2001	14:30	6.00
Cerveja Long Neck	2	3	Restaurante	101	23/1/2001	14:30	6.00

Se acrescentarmos o local de consumo à concatenação das colunas referidas, também não obteremos a condição de identificador único, pois um outro consumo da mesma bebida, pelo mesmo quarto em datas diferentes, propiciaria a existência de duas ocorrências de linhas iguais.

Bebida	Qtde	Valor Unitário	Local Consumo	Quarto	Data Consumo	Hora Consumo	Valor Total
Cerveja Long Neck	2	3	Restaurante	101	22/1/2001	14:30	6.00
Cerveja Long Neck	2	3	Restaurante	101	23/1/2001	14:30	6.00

Valores resultantes da concatenação de colunas:

Bebida+Local Consumo+Quarto

Cerveja Long NeckRestaurante101
ChoppBar203
Cerveja Long NeckRestaurante101
Refrig. LataRestaurante203
MateFrigobar407
CaféRestaurante203
SucoService Room505
MateRestaurante407
ChoppBar203
Água Min.Frigobar101
CaféRestaurante203

Chaves encontradas repetidas:

Cerveja Long NeckRestaurante101
Cerveja Long NeckRestaurante101

Vamos então tentar acrescentar uma coluna à concatenação que estamos tentando: a data de consumo.

Como teoricamente os consumos foram realizados em datas diferentes, podemos inserir a data na composição da chave primária.

Bebida+Local Consumo+Quarto+Data Consumo

Cerveja Long NeckRestaurante10122/1/2001
ChoppBar20319/1/2001
Cerveja Long NeckRestaurante10123/1/2001
Refrig. LataRestaurante20320/1/2001
MateFrigobar40721/1/2001
CaféRestaurante20318/1/2001
SucoService Room50522/1/2001
MateRestaurante40721/1/2001
ChoppBar20319/1/2001
Água Min.Frigobar10118/1/2001
CaféRestaurante20318/1/2001

Desta vez não aparecem valores repetidos? Será que conseguimos efetivamente uma chave primária?

Voltando a analisar toda a tabela e seus dados, existe algum caso em que estes valores estejam repetidos?

Classificando as concatenações obtidas, podemos observar se existem repetições:

Bebida+Local Consumo+Quarto+Data Consumo

Água Min.Frigobar10118/1/2001
CaféRestaurante20318/1/2001
CaféRestaurante20318/1/2001
Cerveja Long NeckRestaurante10122/1/2001
Cerveja Long NeckRestaurante10123/1/2001
ChoppBar20319/1/2001
ChoppBar20319/1/2001
MateFrigobar40721/1/2001
MateRestaurante40721/1/2001
Refrig. LataRestaurante20320/1/2001
SucoService Room50522/1/2001

Observe o consumo do Mate pelo quarto 407, o consumo de Chopp do quarto 203 e o consumo de Café do mesmo 203.

Para estes casos houve dois consumos na mesma data.

Bebida	Qtde	Valor Unitário	Local Consumo	Quarto	Data Consumo	Hora Consumo	Valor Total
Chopp	3	2	Bar	203	19/1/2001	11:00	8.00
Chopp	4	2	Bar	203	19/1/2001	17:10	8.00
Café	1	0,8	Restaurante	203	18/1/2001	08:00	0.8
Café	1	0,8	Restaurante	203	18/1/2001	18:00	0.8
Mate	1	1,5	Frigobar	407	21/1/2001	16:30	1.50
Mate	1	1,5	Restaurante	407	21/1/2001	16:30	1.50

Porém, os dois consumos de Mate foram realizados em locais diferentes, logo o valor de nossa chave primária é diferente nas duas linhas em questão. O consumo de Chopp do quarto 203 e o consumo de Café do quarto 203 também foram ambos realizados no mesmo local e na mesma data. Logo, não podemos utilizar esta concatenação como a chave primária desta tabela. Então vamos continuar a analisar o conteúdo da tabela para concluirmos melhor.

Analizando os valores da tabela, percebemos que os dois consumos foram realizados no mesmo local, mesma data e pelo mesmo quarto, mas em horários diferentes. Salvação!

Temos então mais uma alternativa para finalmente obtermos a chave primária. Vamos concatenar mais uma coluna, a coluna horário do consumo, e ver como fica o resultado desta concatenação.

Colunas Concatenadas

Bebida+Local Consumo+Quarto+Data Consumo+Hora Consumo

Água Min.Frigobar10118/1/200114
CaféRestaurante20318/1/200111:00
CaféRestaurante20318/1/200114:30
Cerveja Long NeckRestaurante10122/1/200108:45
Cerveja Long NeckRestaurante10123/1/200116:30
ChoppBar20319/1/200108:00
ChoppBar20319/1/200121:30
MateFrigobar40721/1/200116:30
MateRestaurante40721/1/200117:10
Refrig. LataRestaurante20320/1/200108:30
SucoService Room50522/1/200118:00

Os casos que temos de concatenação com repetição de valores foram eliminados, não existindo agora nenhuma linha repetida nesse resultado, ou seja, não existem dois valores iguais nesta tabela para o resultado da concatenação de colunas.

Logo, podemos definir como chave primária da tabela o conjunto de colunas concatenações pelo qual obtivemos este resultado:

**Chave Primária (Primary Key) =
Bebida + Local Consumo + Quarto + Data Consumo + Hora Consumo**

Veja a importância de darmos atenção ao conteúdo possível em uma tabela de banco de dados antes de determinarmos a sua chave primária.

Vamos continuar e estudar outra característica importante da chave primária de uma tabela relacional.

Valores Nulos e Integridade de Identidade

Para a apresentação deste tópico é necessário o perfeito entendimento do significado de valor nulo. Vamos ilustrar abstraindo-nos em alguma realidade do dia a dia.

O que existe no mundo que contém de tudo o que se imagina? Resposta: as bolsas femininas.

Este mundo incrivelmente inventivo faz com que esses fantásticos seres coloquem os objetos mais incríveis e até às vezes absurdos dentro das pequenas e médias bolsas que sempre carregam orgulhosas.



A pergunta que colocamos neste momento é: qual o valor do conteúdo de uma dessas bolsas olhando-a simplesmente, sem abri-la e sem ter visto o que foi colocado nela?

O leitor pode dizer:

Deve terhummm..???? hummmm...???????

Deve ter, pois você não sabe o que há lá dentro.

Para você o conteúdo da bolsa no instante é desconhecido. É uma informação perdida, ou não disponível, logo é nula a sua resposta.

O valor do conteúdo da bolsa logo é nulo.

Vamos a outro exemplo. Ao observarmos a saída de um *shopping*, vemos muitas pessoas carregar sacolas de compras. Elas nunca saem com uma sacola somente.

Pergunta-se:

Qual o conteúdo daquelas sacolas para quem as observa, sem abri-las?

(O marido apavorado!)



Você deve concordar comigo que o conteúdo é desconhecido. Você tem certeza de que existe alguma coisa dentro, porém não sabe o que nem quanto custou.

O valor do conteúdo de cada bolsa é então um valor nulo para nossa mente!

Concluindo, então, valor nulo é um valor ausente, desconhecido para nós.

Por definição, todas as linhas de uma tabela têm de ser distinguíveisumas das outras, devem possuir um identificador único.

Um identificador de chave primária nulo significa dizer que existe uma ocorrência (linha) na tabela sem identificação única, ou não distinguível.

Se uma tabela relacional tem uma chave primária, que é um valor único para cada linha da tabela, logo esse valor não pode em hipótese nenhuma estar nulo, ou seja, ser desconhecido.

Estendendo este conceito, destacamos que nenhuma das colunas que participam da composição da chave primária pode ter valor nulo, pois o resultado da concatenação seria nulo, em uma operação de concatenação de valores.

Imagine o leitor então nossa tabela utilizada como exemplo na qual existisse uma linha cuja data e hora de consumo fossem não informadas ou não registradas, desconhecidas, isto é, fossem colunas com valor nulo como apresentado em seguida:

Bebida	Qtde	Valor Unitário	Local Consumo	Quarto	Data Consumo	Hora Consumo	Valor Total
Cerveja Long Neck	2	3	Restaurante	101	nula	nula	6.00
Cerveja Long Neck	2	3	Restaurante	101	23/1/2001	14:30	6.00

O resultado da concatenação de colunas neste caso seria:

Bebida+Local Consumo+Quarto+Data Consumo+Hora Consumo

NULO

Cerveja Long NeckRestaurante10123/1/200116:30

Qualquer valor concatenado com **nada** é igual a **nada**.

No caso de consumos do hotel, isto quer dizer, por exemplo, que não poderíamos ter uma linha de consumo sem a existência de um valor para a data de consumo e hora de consumo. Você não poderia localizar o consumo nem confirmá-lo.

O leitor poderia questionar, como já aconteceu de um aluno perguntar:

Considerando que só existe uma linha cuja concatenação resultou em valor nulo, e esse valor nulo logicamente não se repete na tabela, por que não é válido como identificador único (afinal ele é único na tabela)? Por que não pode ser chave primária?

A resposta é simples; um valor nulo não identifica absolutamente nada. É uma questão mais conceitual do que matemática.

Como recuperar a linha cuja identificação é nula?

Como identificar uma pessoa cujo registro de nascimento é nulo, ou seja, desconhecido?

Qual o valor de um consumo que não podemos identificar quando ocorreu? Você aceitaria ser cobrado por um consumo em um hotel sem que ele fosse identificado?

Imagine esta situação:

O atendente do *check-out* do hotel informa-lhe:

O senhor consumiu duas cervejas no restaurante, mas não sabemos informar o dia nem a hora, porém o valor é de R\$ 10,00.

Você pagaria?

Se você pagar, vai permitir que isso se repita! Logo, permitirá chave primária duplicada.



Se permitirmos valores nulos como chave primária, como garantir que não se repitam?

Regra de Integridade de Identidade

Desta forma concluímos que a identificação de uma linha de uma tabela não pode ser feita por um valor desconhecido, motivo pelo qual a chave primária de uma tabela não pode possuir nenhum elemento de sua composição com valor nulo.

Chave Primária

- ▶ Coluna ou concatenação de colunas.
- ▶ Valor único na tabela.
- ▶ Cada linha tem um valor diferente na chave primária.
- ▶ Não existem valores nulos na chave primária.

Atributos cujos valores no mundo real podem ser duplicados não devem ser definidos como chaves de uma tabela (Nome, como já destacamos).

Em geral, uma tabela pode ter mais de uma chave que possua a capacidade de identificação única das linhas da tabela. Neste caso, cada uma dessas chaves da tabela é chamada de CHAVE CANDIDATA.

Uma das chaves é definida como primária e as outras ficam como chaves alternativas à chave primária.

Em geral, entre todas as chaves candidatas, escolhe-se para chave primária aquela com o menor número de atributos ou mais significativa conceitualmente na identificação de cada fato ou ocorrência (linha) de uma tabela.

Esquema de uma Tabela

A definição de uma tabela relacional é realizada por um formato denominado esquema da tabela. Esse formato destaca a chave primária por um sublinhado no nome da coluna:

Para a tabela de funcionários que apresentamos o esquema seria:

- ▶ Funcionário {NUMREG, NOMEFUNC, DTADMISSAO, SEXO, TELEFONE, CDDEPTO}

Chave Estrangeira

Uma tabela relacional, como já vimos, é uma estruturação dos dados por assunto, organizada em tabelas com linhas e colunas, e cada linha é a representação de uma ocorrência de um objeto, um assunto, descrita por valores em cada coluna. Dessas colunas já sabemos que uma ou um conjunto delas forma o que definimos como o identificador único de cada linha que é a chave primária da tabela.

Vamos agora apresentar outras propriedades que as colunas de uma tabela relacional podem ter e as regras dessas propriedades.

Continuando o estudo, vamos analisar mais características das colunas das tabelas em um banco de dados relacional, e para isso utilizaremos um conjunto de tabelas em um banco de dados como apresentado em seguida.

Existem três tabelas nesse pequeno banco de dados.

São as tabelas referentes aos produtos de nossa cozinha, neste caso somente os alimentos. Uma chamaremos de tabela de Estoque de alimentos, uma de Fornecedores e uma de Unidade de armazenamento.

Uma característica importante nas tabelas relacionais é que elas têm muitas vezes colunas comuns.

Estoque de Alimentos

Alimento	Quantidade	Data Validade	Fabricante	Unidade
Feijão	2	20/08/2004	2	1
Leite	3	12/07/2004	4	2
Açúcar	5	12/08/2004	1	1
Arroz	3	10/10/2004	6	1
Azeite	2	12/03/2004	5	6
Café	1	12/12/2004	3	1

Fornecedores

Fabricante	NomeFab
2	Coral
4	CCPL
1	União
6	Tio João
5	Galo
3	Pilão

Unidades de Armazenamento

Unidade	Descricao
1	Kg
2	Litro
3	Peca
4	Envelope
5	Pote 500 g
6	Vidro 500 g

Esquema do Banco de Dados

- ▶ Estoque de Alimentos {ALIMENTO, QUANTIDADE, DATA VALIDADE, FABRICANTE, UNIDADE}
- ▶ Fornecedores {FABRICANTE, NOME FABRICANTE}
- ▶ Unidades de armazenamento {UNIDADE, DESCRIÇÃO}

Observe que a tabela de Estoque de alimentos tem as colunas Número do Fabricante e Código da Unidade, que também existem, respectivamente, nas tabelas Fornecedores e Unidade de armazenamento.

Esta é uma característica que em primeiro lugar tem como objetivo evitar que sejam inseridos na tabela de alimentos, por exemplo, valores relativos a um mesmo fornecedor de duas maneiras: Tio João e T. João, fazendo com que apresentássemos o mesmo produto como se fossem de dois fornecedores diferentes.

Isso ajuda a eliminar ou diminuir erros de entrada de dados nos sistemas, e manter a consistência do banco de dados, pois utilizamos o número do fornecedor em lugar de, talvez, digitar o seu nome. Isso será objeto do capítulo de normalização neste livro.

Mas bem, o que significa uma tabela ter coluna ou colunas que existem em outra tabela do mesmo banco de dados?

Analisando a figura anterior e o esquema do banco de dados, observamos que cada tabela tem uma chave primária.

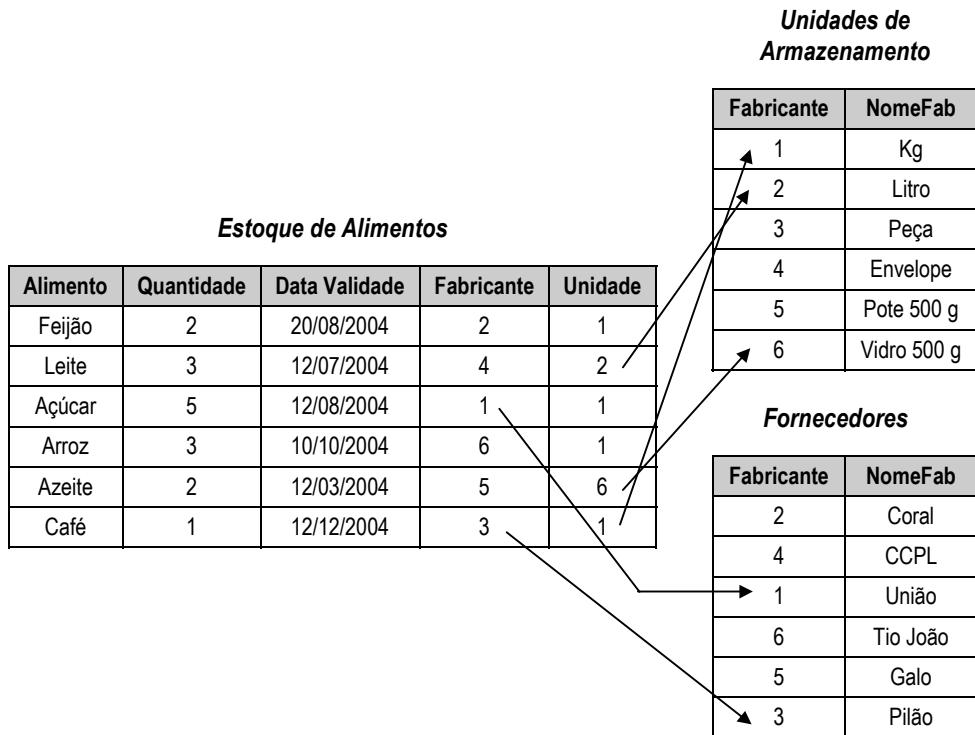
A tabela Estoque de alimentos tem como chave primária a coluna Alimento.

A tabela Fornecedores tem como chave primária a coluna Fabricante, e a tabela Unidades de armazenamento tem como chave primária a coluna Unidade.

O que significa quando temos um campo que é chave primária de uma tabela que faz parte dos campos de outra tabela?

Isso é o que denominamos de **chave estrangeira**.

É uma referência de um elemento de uma tabela a um elemento de outra tabela, uma relação entre as tabelas, uma ligação lógica entre elas.



Então, em nosso exemplo a coluna Fabricante na tabela Estoque de alimentos é uma chave estrangeira, assim como Unidade também é outra chave estrangeira nesta tabela.

Uma tabela pode ter tantas chaves estrangeiras quantas forem as suas associações a outras tabelas.

Uma tabela pode ter um conjunto de atributos que contêm valores com o mesmo domínio de um conjunto de atributos que formam a chave primária de uma outra tabela.

Esse conjunto se chama **chave estrangeira**.

Vejamos outro exemplo clássico e simples.

Funcionário

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/8/2003	M	C3	D5
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
123	Pedro Sergio Doto	29/6/2003	M	C7	D3
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/2/2000	M	C2	D4

Departamento

CdDept	NumDept	RamalTel
D1	Assist.Técnica	2246
D2	Estoque	2589
D3	Administração	2772
D4	Segurança	1810
D5	Vendas	2599
D6	Cobrança	2688

Cargo

CdCargo	NumCargo	VlrSalario
C1	Aux Vendas	350,00
C3	Vendedor	800,00
C7	Diretor	2500,00
C2	Vigia	400,00
C5	Gerente	1000,00
C4	Aux Cobrança	250,00

A coluna CdDept na tabela Funcionários é uma chave estrangeira, pois ela é chave primária na tabela Departamentos.

A coluna CdCargo em Funcionário também é uma chave estrangeira, pois ela é chave primária em Cargos

Para lembrar e fixar, vamos deixar neste ponto regras que norteiam o conceito de chave estrangeira em uma tabela.

Sempre que uma coluna de uma determinada tabela A (Funcionário) for uma chave primária em uma tabela B (Cargo), essa coluna (CodCargo) na tabela A é uma chave estrangeira em relação à mesma coluna (CodCargo) na tabela B.

Este conceito estabelece uma regra em bancos de dados relacionais denominada de integridade referencial.

Vamos então entender um pouco mais isso.

Integridade Referencial

Quando colocamos uma coluna como chave estrangeira em uma tabela, assumimos responsabilidade com o banco de dados por assim dizer.

As colunas pertencentes à chave estrangeira da tabela A devem ter o mesmo domínio das colunas pertencentes à chave primária da tabela B.

O valor de uma chave estrangeira em uma tabela A deve ser de chave primária da tabela B, ou então ser nulo.

Sintetizando, uma tabela contém uma chave estrangeira, então o valor dessa chave só pode ser:

- ▶ Nulo. Neste caso pode, pois representa a inexistência de referência para uma linha da tabela.
- ▶ Igual ao valor de alguma chave primária na tabela referenciada.

Você pode perguntar como ficaria uma tabela com chave estrangeira nula. Vejamos:

Funcionário

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/8/2003	M	C3	D5
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
123	Pedro Sergio Doto	29/6/2003	M	Nulo	D3
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/2/2000	M	C2	D4

Na linha de Pedro Sergio Doto o valor para CdDept está nulo, o que pode significar que ainda não está alocado a nenhum departamento, ou foi deslocado de algum departamento.

O que importa é que ele não tem um departamento assinalado, o que é uma situação válida.

O que não pode haver é um valor de chave estrangeira que não exista como chave primária de nenhuma linha da tabela referenciada, no caso a tabela Departamento.

Na definição de uma chave estrangeira somente podemos nos referenciar a uma chave primária de uma outra tabela?

Nem sempre isso é verdade.

Na criação de uma chave estrangeira, além de podemos nos referenciar a um campo chave primária de outra tabela, também podemos nos referenciar a uma coluna que tenha sido definida como única, uma chave candidata.

Qual a razão da integridade referencial? O que ela implica?

Restrições para Garantir a Integridade Referencial

Existe um conjunto de regras de operação para um banco de dados relacional que coloca restrições, regras nas operações de atualização das tabelas do banco de dados, de forma a garantir e manter a integridade referencial.

► PARENT DELETE RESTRICT: (Deleção Restrita)

Ao excluir (deletar) a tabela pai (parent), se ela possuir filhos relacionados (ou seja, se o departamento tiver funcionários), a exclusão é impedida (RESTRICT).

Isso evita que o banco de dados fique inconsistente, ou seja, linhas de Funcionário com valor de chave estrangeira inexistente como chave primária na tabela associada.

Outras opções para garantir a integridade de referências do banco de dados seriam excluir todos os filhos em cascata (CASCADE), fazendo com que todos os funcionários referenciem um departamento padrão, CdDept=3 (Administração), por exemplo, ou fazer com que todos os funcionários fiquem sem departamento, CdDept = NULL

► CHILD INSERT RESTRICT: (Inclusão e Linha Restrita)

Ao inserir um funcionário, caso seja obrigatório que já possua departamento associado, verifica se ele está relacionado a um departamento existente na tabela Departamento, senão impede a operação (RESTRICT).

► CHILD UPDATE RESTRICT:

Ao atualizar (UPDATE) a chave estrangeira de uma tabela (CHILD), verifica se existe uma linha da tabela associada que possua como chave primária o novo valor da chave estrangeira, senão impede essa operação (RESTRICT).

A opção Cascade (Cascata) é sempre perigosa de ser utilizada em banco de dados, pois existe o risco de perder todos os dados existentes em uma determinada tabela se optar por apagar as suas linhas que estão associadas a uma determinada linha que será deletada da tabela que possui a chave primária referenciada.

Vamos observar esta hipótese com uma simulação de tabelas.

Suponha que a tabela tenha somente os funcionários do departamento de vendas (D5).

Funcionário

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/8/2003	M	C3	D5
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
123	Pedro Sergio Doto	29/6/2003	M	Nulo	D3
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/2/2000	M	C2	D4

Departamento

CdDept	NumDept	RamalTel
D1	Assist.Técnica	2246
D2	Estoque	2589
D3	Administração	2772
D4	Segurança	1810
D5	Vendas	2599
D6	Cobrança	2688

Se estabelecermos para a tabela Departamento a regra de Cascade, e apagarmos (deletar) a linha cuja chave primária é ="D5", o resultado será a tabela funcionário como apresentada em seguida:

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept

Vazia, completamente sem linhas. O Cascade provoca que todas as linhas de tabelas associadas a essa chave primária sejam deletadas, apagadas para evitar que existam no banco de dados referências às linhas inexistentes em uma tabela.

Não é somente o caso de deleção completa da tabela que preocupa, pois trabalhamos com nossa tabela original de funcionários:

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/8/2003	M	C3	D5
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
123	Pedro Sergio Doto	29/6/2003	M	Nulo	D3
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/2/2000	M	C2	D4

Se executarmos a mesma operação de deleção da linha relativa a Departamento de Vendas (D5), essa tabela perderá todas as linhas que estavam associadas pelo valor de chave estrangeira = "D5", perdendo os dados de alguns funcionários:

Funcionário

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
123	Pedro Sergio Doto	29/6/2003	M	Nulo	D3
22	Sergio Nogueira	10/2/2000	M	C2	D4

Importante: As chaves estrangeiras baseiam-se em valores (dados) e são puramente lógicas, não existem apontadores físicos.

Resumo de Restrições de Integridade Relacional

Dada uma linha α de uma tabela A:

Restrição de Inclusão

- ▶ se ocorre uma inclusão da linha α ,
- ▶ se algum atributo da chave primária da linha α for nulo,
- ▶ ou existe outra linha na tabela A com o mesmo valor da chave primária da linha α ,
- ▶ então impede-se a inclusão de linha,
- ▶ senão, realiza-se a inclusão de linha.

Restrição de Deleção

- ▶ se ocorre uma exclusão da linha α ,
- ▶ se algum atributo de uma outra tabela faz referência à chave primária da linha α (existe uma chave estrangeira com o valor da chave primária de α),
- ▶ então impede-se a inclusão de linha,
- ▶ ou realiza-se a deleção em cascata das linhas da outra tabela que referenciam o valor da chave primária de α ,
- ▶ ou modifica-se (altera-se) para nulo o valor da chave estrangeira da outra tabela.

Exercícios

1. Dadas as tabelas de um banco de dados e as operações a serem realizadas, complete as tabelas resultantes:

Tabela R

A	B	C
3	5	2
3	4	1
5	5	4
6	7	8

Tabela S

B	C	D
4	2	9
5	1	3
7	8	3

Tabela R{A,B,C}

Tabela S{B,C,D}

- a. Inclusão da linha {2,4,2} na tabela R.

Tabela R

A	B	C

Tabela S

B	C	D

- b. Deleção da linha {4,2,9} da tabela S.

Tabela R

A	B	C

Tabela S

B	C	D

2. Dadas as tabelas de um banco de dados bancário, defina as chaves candidatas, as chaves primárias e as chaves estrangeiras existentes nas tabelas.

Cliente

CodCliente	NomCliente	RuaCliente	CidadeCliente
1	Luis Sampaio	Rua A	Rio de Janeiro
2	Carlos Pereira	Rua B	Niterói
3	Jose Alves	Rua C	Rio de Janeiro
4	Luis Paulo Souza	Rua B	Niterói

Conta-Corrente

CodAgencia	NumConta	CodCliente	Saldo
1	256589	1	1200,00
3	328941	1	845,00
4	749621	3	512,00
2	856200	2	2650,00
3	454501	4	800,00
2	468952	3	6920,00
4	278156	1	10000,00

Agência

CodAgencia	NomAgencia	CidadeAgencia
1	Rio Branco	Rio de Janeiro
2	Icarai	Niterói
3	Leblon	Rio de Janeiro
4	Ipanema	Rio de Janeiro
5	Copacabana	Rio de Janeiro

Empréstimo

CodAgencia	CodCliente	NumEmprest	Valor
1	1	902230	500,00
3	1	902231	1500,00
4	2	902240	1200,00
2	3	902289	3000,00
3	1	902255	850,00
1	3	902299	700,00
4	3	902212	400,00

3. Responda às questões:

- a. O que acontece se deletamos a linha...

3	1	902231	1500,00
---	---	--------	---------

... da tabela Empréstimo?

- b. O que acontece se deletamos a linha...

3	Leblon	Rio de Janeiro
---	--------	----------------

... da tabela Agência?

- c. Como fica a tabela ContaCorrente se deletarmos a linha seguinte da tabela Cliente, considerando a manutenção da integridade referencial?

2	Carlos Pereira	Rua B	Niterói
---	----------------	-------	---------

4. Explique a diferença entre manter a integridade referencial por deleção em Cascata e manter por chaves estrangeiras nulas.

5. Uma tabela B pode ter duas chaves estrangeiras para uma tabela A?

6. O que significa uma chave estrangeira com valor nulo?

7. Uma chave alternativa pode ter valores repetidos na mesma tabela?

8. Quais dos esquemas de BD apresentados em seguida estão errados?

- a. Cliente{NOME, IDADE, SEXO, TELEFONE, TELEFONE, RUA}
- b. Cliente{CODCLIENTE, NOME, IDADE, SEXO, TELEFONE, ENDERECO, CIDADE}
- c. Cliente{CODCLIENTE, NOME, IDADE, SEXO, TELEFONE, ENDERECO, CIDADE, CODSEXO}
Sexo {NUMSEXO,NOMESEXO}
- d.)Cliente{CodCLIENTE, NOME, IDADE, SEXO, TELEFONE, , ENDERECO, CIDADE, CODSEXO}
Sexo {CodSEXO,NOMESEXO}
- e. Todos
- f. Nenhum

MODELO ENTIDADE-RELACIONAMENTO

A utilização da abordagem correta de uma 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 de um banco de dados:

- ▶ Um modelo conceitual;
- ▶ Um modelo lógico;
- ▶ Um modelo 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 de um determinado ambiente.

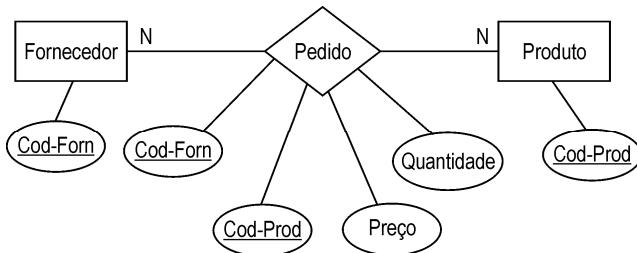
Devido a essa não similaridade entre ambientes de mesma natureza, será sempre necessária a criação de um modelo específico para cada nova realidade observada. Fica bem clara, então, a necessidade de termos um modelo que permita construir vários outros modelos, o qual é chamado de metamodelo.

O metamodelo modelo a ser utilizado deve ter a característica de poder modelar qualquer realidade, uma forma de trabalho bastante simples e características gráficas fáceis de construir e entender. O metamodelo que apresentamos neste livro é o Entidade-Relacionamento (ER).

O modelo Entidade-Relacionamento (ER) foi definido por Peter Pin-Shan Chen, em 1976, e baseia-se na percepção do mundo real como constituído por um conjunto de objetos básicos chamados entidades e relacionamentos e define uma técnica de diagramação para modelos de dados, o diagrama de entidades e relacionamentos.

A **estrutura** lógica global de um banco de dados pode ser expressa graficamente por um diagrama de entidades (representadas por retângulos), por relacionamentos (representados por losangos) e pelos atributos de cada entidade ou relacionamento através de elipses (notação Peter Chen).

Apesar de ser uma forma gráfica de representar um banco de dados relacional, o modelo ER possui um elevado grau de semântica, que o torna mais simples, além de permitir uma comunicação mais otimizada entre os usuários e os profissionais de informática, desenvolvedores de sistemas.



Dentro do que utilizamos como método de desenvolvimento podemos tanto iniciar a análise por modelagem de eventos ou casos de uso quanto por modelagem de dados para depois, com as entidades definidas, analisarmos os eventos que existem relativos a cada um desses objetos.

A própria UML prega a utilização da criação dos diagramas de casos de uso como primeira etapa de um processo metodológico de desenvolvimento de aplicações. Porém, é um cacoete derivado dos conceitos de análise estruturada e análise essencial, em que destaque e importância maior sempre foram dados aos processos e não aos dados.

Voltamos a lembrar que dados são sempre mais estáveis que processos.

Observe que podemos mudar completamente um processo de faturamento, mas os dados relativos ao faturamento continuaram estáveis. Na pior das hipóteses esses dados seriam reduzidos, mas não haveria uma mudança completa dos dados utilizados para esse processo.

Os diagramas de caso de uso fornecem um modo de descrever a visão externa do sistema e suas interações com o mundo exterior, representando uma visão de alto nível de funcionalidade intencional mediante o recebimento de um tipo de requisição de usuário. Entretanto, no caso de iniciarmos um desenvolvimento pela especificação de casos de uso, podem conduzir-nos à criação de classes de dados redundantes ou não aderentes ao mundo real, o que obtemos sempre quando trabalhamos com modelagem de dados pura.

Já a modelagem de dados pode permitir a criação mais concisa de modelo de classes, já que um modelo de dados retrata com maior fidelidade uma realidade, ou seja, o mundo real.

A estabilidade do modelo de dados é obtida quando se investe quantidade de tempo nos aspectos conceituais de modelagem de dados, revertendo esse tempo em benefícios consideráveis durante a implementação da base de dados.

A técnica de modelagem Entidade-Relacionamento proposta por Peter Chen está definida como uma notação orientada para o desenho de um modelo conceitual, pois permite a descrição desse esquema conceitual sem preocupação com problemas de implementação física ou de performance do banco de dados.

O diagrama Entidade-Relacionamento descreve a estrutura conceitual e lógica geral de um banco de dados.

Observa-se então que o objetivo da modelagem de dados é definir o contexto dos dados em que os sistemas funcionam. Por isso, o produto da modelagem de dados deve ser o mais fiel possível ao mundo real e, além disso, possuir uma característica adicional muito importante: as suas especificações não devem implicar e tampouco estarem limitadas a nenhuma implementação física em particular. Tudo aquilo que for proposto pela modelagem de dados deve ocorrer num nível conceitual e lógico, de maneira a não viciar a análise pela restrição de pontos de vista técnicos da equipe de participantes do processo de análise. Os detalhes físicos vêm depois.

A abordagem de sistemas para bancos de dados relacionais tem como principal característica o alto nível em que ocorrem as definições necessárias à implantação de uma base de dados.

Assim, a qualidade de um projeto como um todo é muito dependente da qualidade da modelagem de dados que o antecede.

É muito importante então que exista uma metodologia simples, precisa e eficiente para a representação dos objetos modelados pelo analista e que também seja de fácil transposição para os diversos SGBD disponíveis.

É isso que se obtém com a utilização do modelo ER proposto por Peter Chen.

Elementos de um Modelo ER

Um modelo de dados ER é composto de três classes de objetos: entidades, relacionamentos e atributos.

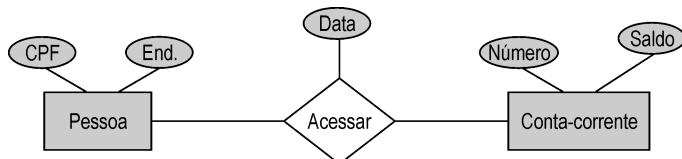
As literaturas existentes nunca deixam claro como podemos entender entidades e relacionamentos. Uma vez que a maioria dos profissionais de análise de sistemas tem sua cultura baseada em sistemas procedurais, em que os dados são o resultado e não o meio, existe a necessidade de que se coloque mais enfoque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

As técnicas orientadas a objetos, hoje utilizadas no País e que alcançaram divulgação profissional prática, baseiam-se na análise dos procedimentos, e com enfoque principalmente direcionado aos casos de uso (*use case*) e em diagramas de classes excessivamente preocupados em retratar não só dados e sim processo em conjunto, ou nos casos de análise essencial em diagramas de fluxo de dados, o que provoca uma derivação de foco e possibilita uma larga margem de erro de conceituação do mundo real.

Essas técnicas estruturadas colocam as informações derivadas dos procedimentos em classes de dados (OO) ou depósitos de dados (A Essenc.) as quais sintetizando-se finalmente acabam sendo convertidas em tabelas de um sistema, pura e simplesmente.

Para que se efetue então a migração dessa base cultural, é 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. Nossa objetivo primordial é entendermos o negócio, para o qual projetaremos um sistema, através de seus dados.



Quando escrevemos **objetos conceituais**, não pretendemos inserir na orientação a objetos. Apesar de a modelagem conceitual de dados ser a base para o entendimento dessa nova abordagem tecnológica, o objetivo é na realidade ir até as raízes da conceituação de modelo Entidade-Relacionamento.

Quando Peter Chen formulou a proposta do modelo Entidade-Relacionamento, ele se baseou não na visão de um sistema de aplicação como princípio e sim na compreensão da realidade em que se situava o problema. Como vamos 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 esse negócio, independentemente de preocupar-se com formas de tratamento das informações, procedimentos, programas etc.

Entidades

Correspondem a quaisquer coisas do mundo real sobre as quais se deseja armazenar informações.

São exemplos típicos de entidades: pessoas (físicas ou jurídicas, tais como funcionário, empresa, fornecedor e cliente); objetos materiais ou abstratos, como produto, veículo, disciplina e projeto e eventos ou fatos como pedido, viagem, empréstimo e venda principalmente.

No modelo ER são representados por meio de um retângulo com o nome representativo da entidade (um substantivo no singular) ao centro.

Uma entidade normalmente tem várias manifestações dela mesma. Por exemplo, a entidade funcionário representa todos os funcionários da empresa, e não apenas um deles; a entidade produto representa todos os produtos com os quais a empresa trabalha etc.

Se fizermos uma comparação com a UML, funcionário é o um objeto, o conjunto dos objetos funcionários forma a classe de objetos funcionário, que é a nossa entidade. Então uma entidade corresponde a uma classe de objetos e as ocorrências desta entidade são os objetos funcionários.

Dizemos então que uma entidade possui ocorrências ou instâncias, e cada um dos funcionários descritos pela entidade funcionário é uma de suas ocorrências, ou instâncias.

Entidade é a principal classe de objetos sobre a qual são coletadas informações. Ela normalmente denota pessoa, lugar, coisa ou fato de interesse de informações.

É todo objeto concreto ou abstrato que tem existência própria, quando considerado o âmbito de um negócio. São coisas sobre as quais desejamos arquivar informações.

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.

A figura seguinte apresenta o ambiente de uma clínica médica correspondente à descrição de um minimundo.

Uma clínica médica necessita controlar as consultas médicas realizadas e marcadas pelos médicos a ela vinculados, acompanhar os pacientes atendidos e manter o seu acompanhamento clínico. Para cada médico a clínica mantém uma ficha com o número de CRM, seu nome, endereço, especialidade etc. Os pacientes têm 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.

Quais são os objetos candidatos a entidades nesse ambiente em observação?

Observe com a sua capacidade de abstração as coisas que existem no ambiente: médicos, pacientes, exames, consulta.

Essas coisas que fazem parte do ambiente são entidades, pois podemos manter informações sobre elas, e são participativas na existência do ambiente. Cada uma delas tem significado próprio.

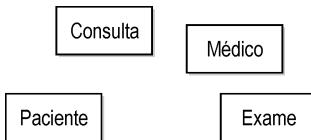


Se alguma coisa existente no ambiente negócio nos proporciona algum interesse em mantermos dados sobre essa coisa (informações armazenadas sobre ele), isso a caracteriza como uma entidade do negócio.

Cada entidade será então um elemento em nosso modelo conceitual de dados.

Uma entidade é a representação de uma classe de dados de um ambiente específico, um conjunto de informações de mesmas características. Cada ocorrência da coisa que representamos como uma entidade é chamada de instância (ocorrência) e representa um conjunto particular desses dados.

Quando representamos classe de dados, atuamos em um nível de abstração de dados interpretado de acordo com o entendimento do meio, do ambiente em que nos localizamos e seus interesses e objetivos.



Uma instância de uma entidade não tem representação no diagrama ER, porém devemos sempre visualizar em uma entidade a ocorrência de linhas de uma tabela relacional, uma vez que estamos realizando análise para o contexto lógico de banco de dados relacional.

Data_da_Consulta	CRM_do_Medico	Identificacao_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

Endereco	Sexo	Idade
R. Silva Sá, 23/11	Masc	33
R. Dias Melhores 334/122	Fem	18
Av. Arapanés 4487/1915	Fem	44
R. Botica do Ouvidor 44/fundos	Masc	29
Trav. das Camélias 661	Masc	38

Quando analisamos um ambiente, tiramos uma fotografia dele que nos apresenta as entidades que participam, ou estão naquele ambiente.

A representação gráfica de uma entidade em um diagrama ER é realizada por meio de um retângulo com o nome da classe de dados inserido em seu interior conforme apresentamos na fotografia da página anterior.

Numa visão simplista, entidade é similar a um arquivo de um sistema convencional, e é natural fazer esta correspondência: um conjunto de ocorrências de uma entidade corresponde a um arquivo, e uma única **ocorrência**, que podemos denominar de linha, corresponde a um registro.

Relacionamentos

Relacionamento é a representação das associações existentes entre entidades no mundo real. Muitos relacionamentos não têm existência física ou conceitual, outros dependem da associação de outras entidades.



No mundo real uma entidade muito raramente se apresenta isolada, tendo existência completamente independente de quaisquer outras.

Normalmente ocorre o contrário: é detectada a existência de uma associação entre as ocorrências de duas entidades distintas.

Essa conexão lógica entre duas ou mais entidades definimos como relacionamento, que é representado em um diagrama Entidade/Relacionamento por meio de uma linha unindo as entidades associadas, contendo ainda um losango com o nome do relacionamento (um verbo flexionado) ao centro.

Uma ocorrência em particular de um relacionamento é chamada de instância de relacionamento, ou ocorrência. Um relacionamento é descrito em termos de grau, cardinalidade e existência.

O mais comum dos termos associados a um relacionamento é a indicação da cardinalidade entre as ocorrências das entidades relacionadas: um-para-um, um-para-muitos e muitos-para-muitos.

Como cardinalidade é um termo que não explicita muita coisa, vamos utilizar a palavra conectividade porque acreditamos que entender o conceito de conexão é mais simples. Alguma coisa se conecta a outra.

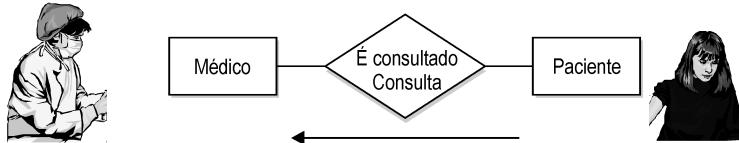
O nome do relacionamento normalmente é um verbo, pois é resultante de um fato que associa as entidades. Podemos dar dois nomes, digamos assim, a um relacionamento.

Um verbo para explicar o fato no sentido da entidade A para a entidade B, e outro verbo no sentido da entidade B para a entidade A, uma vez que a leitura de um relacionamento não possui um lado específico do modelo para ser realizada.



Entender relacionamentos e a capacidade de enxergar esses objetos, assim como a sua participação no mundo real, é fator primordial para o resultado de uma modelagem de dados com entendimento e retrato fiel do ambiente em análise.

Não devemos ter antecipadamente medo da provável complexidade de uma técnica, e sim lembrarmo-nos de que nada mais é do que uma forma estruturada de representar as coisas que existem e ocorrem no mundo real, como sempre fizemos com papel e lápis desde a infância.



Dentro deste enfoque relacionamento é um **fato**, 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.

Vamos continuar a conceituação dos elementos de um modelo de dados ER, e depois voltamos ao entendimento e identificação mais clara de relacionamento no mundo relacional.

Atributos

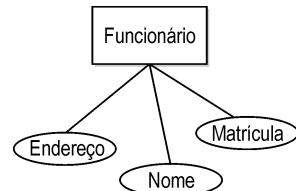
São as características de uma entidade que a descrevem detalhadamente.

Uma ocorrência específica de um atributo em uma entidade ou relacionamento denuncia-se valor do atributo.

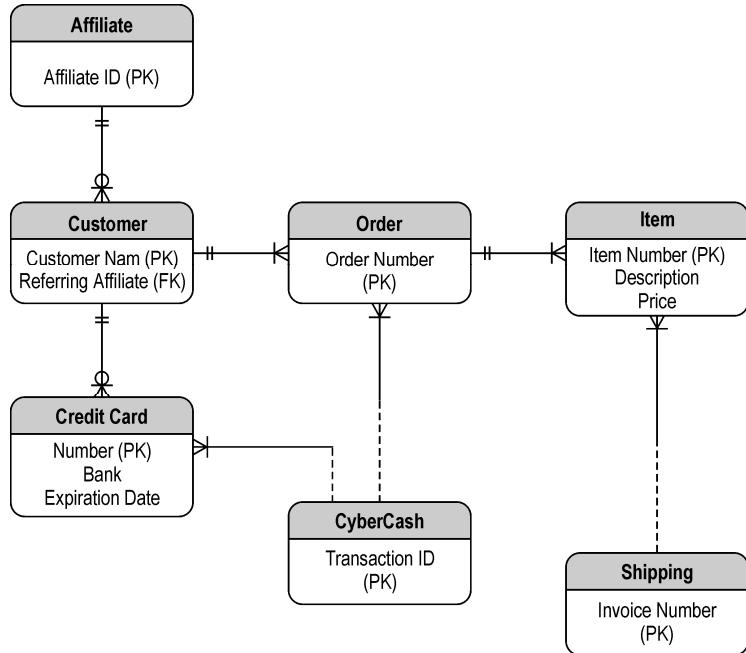
Os atributos representam propriedades elementares de uma entidade ou relacionamento. Cada atributo está associado a um domínio particular, que é um conjunto de valores válidos para o atributo.

Os atributos da entidade empregado, por exemplo, são normalmente a identificação do empregado, o nome, endereço, sexo, telefone, cargo, salário.

A notação de Peter Chen original apresenta o atributo como uma elipse com o nome do atributo em seu interior e conectada à entidade a que pertence.



A simbologia utilizada por Rumbaugh em seu livro coloca uma lista de atributos dentro do retângulo da entidade, deixando sobre ele o nome da entidade. Esta é uma notação gráfica também interessante, entretanto os produtos de software para modelagem de dados utilizam mais a notação IDEF1X e a notação da engenharia de informação que veremos também neste livro.



Existem dois tipos de atributo: identificadores e descritores.

Um identificador (ou chave) é usado unicamente para determinar a identificação de uma ocorrência de uma entidade. Em nosso ambiente relacional equivale à chave primária. Observe que quando estamos executando o processo de abstração para identificar as entidades, não existem chaves primárias do tipo código.

Os atributos descritores (não chaves) são utilizados para descrever características não únicas de uma ocorrência particular da entidade.

Por exemplo, no diagrama da entidade funcionário, o atributo chave, identificador de uma ocorrência de funcionário é matrícula; endereço e nome são seus descritores.

Todo objeto para ser uma entidade tem propriedades que são descritas por atributos e valores. Esses atributos e seus valores, juntos, descrevem as instâncias de uma entidade.

Entidade: Funcionário

Matricula	NomeFunc	DtAdmissao	Sexo
104	Carlos Pereira	2/3/2004	M
134	Jose Alves	23/5/2002	M
123	Pedro Sergio Doto	29/6/2003	M
22	Sergio Nogueira	10/2/2000	M

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 desse funcionário, sua data de admissão e sexo, como é representado na tabela anterior. Poderíamos ainda descrevê-lo com mais dados, tais como data de nascimento, valor de seu salário etc. Esses 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 nesses atributos. É o conjunto desses valores representados que devemos visualizar como uma linha de uma tabela de dados da entidade.

Os valores de determinados atributos ou de um determinado atributo, nas ocorrências dessa 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.

Esse(s) atributo(s) cujos valores nunca se repetem, sempre tem (têm) a função de atuar como identificador(es) único(s) das instâncias da entidade. Dentro da abordagem relacional de banco de dados, denomina-se essa propriedade como chave primária de uma tabela, conceito que vamos utilizar dentro de nosso contexto, quando realizarmos o modelo lógico de dados.

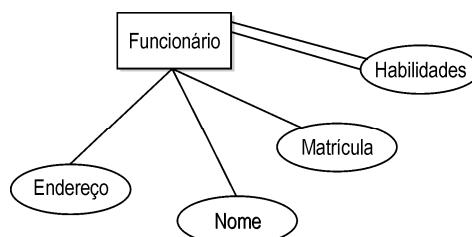
Este conceito não foge de forma nenhuma do que acontece no mundo real, já que um objeto sempre possui uma forma de identificação unívoca.

Quando modelamos, independentemente de estarmos pensando em ambiente de banco de dados relacional, encontramos alguns atributos que são multivvalorados, ou seja, acontecem mais de uma vez para cada ocorrência da entidade. Por exemplo, no caso de funcionário, poderíamos ter um atributo especialidades, ou habilidades, que seria mais de um para cada existência de funcionário.

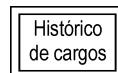
Em um primeiro momento vamos representar esses atributos em destaque com uma conexão de linha dupla com a entidade.



Apesar de afirmarmos a existência de atributos identificadores e descriptivos, encontramos entidades que são derivadas de outras, tendo parte de seus atributos identificadores igual à de outra entidade.

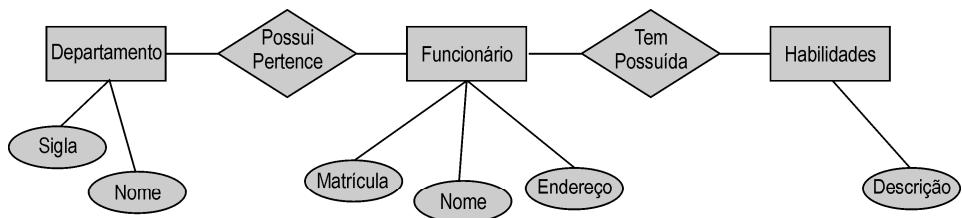


Estas entidades têm existência dependente, ou seja, somente existem se existir a entidade da qual possuem parte dos atributos identificadores. Exemplos claros deste caso são as entidades históricas em um negócio.



No caso da entidade funcionário seria a existência de uma entidade história dos cargos ocupados por cada ocorrência de funcionário. Mas observe que isso é uma derivação, pois inicialmente esses cargos ocupados poderiam ser entendidos como um atributo multivalorado da entidade funcionário.

O objetivo é apresentar as formas de representação de um modelo ER. Representação que exemplifica entidades, relacionamentos e atributos.



Grau de Relacionamento

O grau de um relacionamento é o número de entidades que participam dele.

Existem três tipos básicos de grau de relacionamento: binário, ternário e e-nário que referem a existência de duas, três ou mais entidades envolvidas no fato que o relacionamento representa.

Um relacionamento binário acontece entre duas entidades que é o fato mais comum de acontecer no mundo real.

Podemos ter também um relacionamento binário recursivo construído por meio de conexões com a mesma entidade, acontecendo como se fossem duas coisas diferentes se relacionando na abstração, duas ocorrências da entidade se relacionando, se associando.

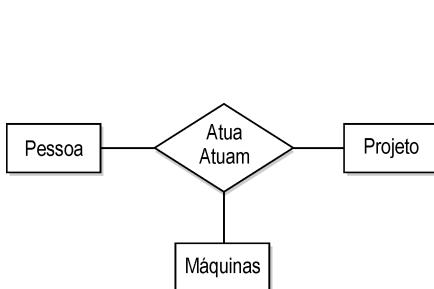
De um relacionamento ternário participam três entidades no mesmo fato. Um relacionamento e-nário é aquele do qual participam múltiplas entidades.

Esta teoria existe nos trabalhos de Peter Chen, porém quando elevamos o nível de abstração, podemos observar que esse tipo de relacionamento não ocorre no mundo real. O que vamos estudar é a existência de relacionamentos dependentes de um relacionamento binário.

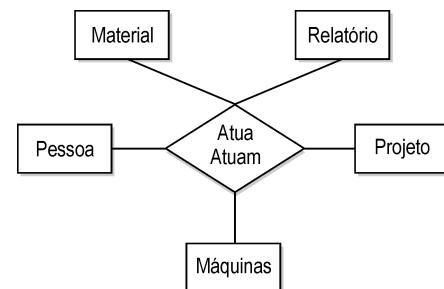
É impossível para o ser humano verbalizar um evento, um fato, um acontecimento de forma ternária. O que sempre interpretamos em uma realidade é um acontecimento como consequência de outro. Logo, não recomendamos nem vamos estudar relacionamentos ternários e múltiplos detalhadamente e sim com poucas apresentações, pois podemos perfeitamente explicá-los e utilizá-los por meio da teoria dos relacionamentos dependentes (agregações), como veremos mais adiante.



Relacionamento Binário



Relacionamento Ternário



Relacionamento E-nário

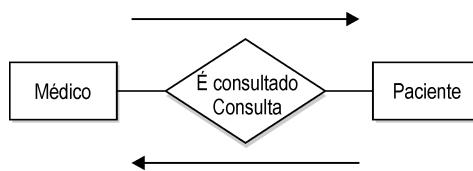
Uma entidade pode estar envolvida em indeterminado número de relacionamentos, assim como podemos ter mais de um relacionamento entre duas entidades, se efetivamente as duas participam de fatos ou acontecimentos distintos, ou essas associações representam formas diferentes de associação entre as suas ocorrências.

Conectividade de um Relacionamento

A conectividade, como gostamos de denominar, descreve as restrições no mapeamento das associações existentes entre as ocorrências de entidades em um relacionamento.

Os valores de conectividade estão sempre entre um ou muitos em um dos lados do relacionamento, da mesma forma que vimos no capítulo 2 sobre abstração em modelagem de dados.

A ocorrência de uma entidade se associa, conecta a uma ocorrência de outra entidade, ou a muitas ocorrências dessa outra entidade.



CRM Médico	Nome Médico	Especialidade	Nome Paciente	Endereço Paciente	Sexo
21113	Luis Paulo Carvalho	Pediatria	Júlio Adamastor	R. Silva Sá, 23/11	Masc
21113	Pedro Estevão	Ginecologista	Carmem Milhor	R. Dias Melhores 334/122	Fem
51024	Mauricio Abreu	Neurologia	Sandra Chu Li	Av. Arapanés 4487/1915	Fem
76004	Simone Almeira	Cardiologia	Álvaro Medeiros Sá	R. Botica do Ouvidor 44/fundos	Masc
			Paulo Alengui	Trav. das Camélias 661	Masc

Observe que não é necessário que todas as ocorrências de uma entidade estejam associadas, conectadas a alguma ocorrência da outra entidade. Podem existir ocorrências em uma entidade que não possuem nenhum relacionamento com ocorrências da outra entidade.

Um relacionamento possui sempre esta característica básica chamada de conectividade ou cardinalidade.

A conectividade de um relacionamento consiste na especificação do sentido da associação entre as entidades envolvidas.



Por exemplo, lendo o relacionamento no sentido de funcionário para departamento, um funcionário pertence sempre a no mínimo um e no máximo um departamento, ou seja, é obrigatório que exista, para uma ocorrência específica da entidade funcionário, tão somente uma ocorrência da entidade departamento associada.

Por outro lado, no sentido de departamento para funcionário, para um determinado departamento é possível que existam vários funcionários relacionados: mais de uma ocorrência da entidade funcionário refere-se à mesma ocorrência da entidade departamento. Dizemos neste caso que a cardinalidade do relacionamento "Pertence" é de 1:N ("um-para-ene" ou "um-para-muitos"). Este é o grau de conectividade ou cardinalidade de um relacionamento.

Vejamos então as conectividades existentes em um modelo de dados.

Conectividade Um-para-Um

Quando entre duas entidades temos um relacionamento em que cada ocorrência da entidade A se associa ou relaciona com uma e somente uma ocorrência da entidade B e cada ocorrência da entidade B se relaciona com uma e somente uma ocorrência da entidade A, então temos um relacionamento com conectividade um-para-um (1:1).



A representação no diagrama é realizada pela inserção do número 1 de cada lado do relacionamento, e deve ser lido da seguinte forma:

- ▶ Um departamento é gerenciado por um e somente um funcionário.
- ▶ Um funcionário gerencia um e somente um departamento.

Observe que colocamos dois verbos para explicar este relacionamento, permitindo a leitura nos dois sentidos.

Vamos buscar outro exemplo de conectividade de um-para-um.

- ▶ Um computador possui um e somente um teclado conectado a ele.
- ▶ Um teclado está ligado a um e somente um computador.



No mundo real e principalmente no ambiente de negócios de uma empresa, que normalmente é onde modelamos, são muito raros os casos de relacionamento de conectividade um-para-um, porém a Lei de Murphy sempre aparece.

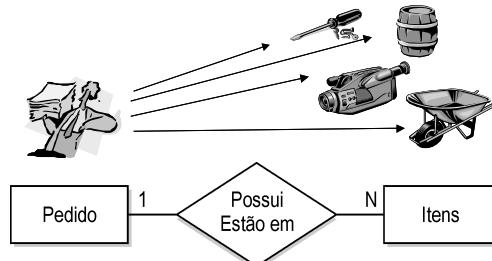
Conectividade Um-para-Muitos

É a conectividade mais comum no mundo real e no mundo dos negócios e a mais utilizada na solução de modelo de dados.

Elá acontece quando uma ocorrência da entidade A se associa ou conecta a mais de uma ocorrência da entidade B, visto que a ocorrência da entidade B está conectada a uma e somente uma ocorrência da entidade A.

Vejamos então alguns exemplos:

- ▶ Um pedido de venda de um cliente tem vários produtos.



A análise e a interpretação de um relacionamento de um-para-muitos devem ser muito cuidadosas, pois é comum observarmos analistas interpretarem como um-para-muitos relacionamentos que em verdade são de muitos-para-muitos.

Observe a figura seguinte que apresenta dois domínios relativos a duas entidades.

Vamos considerar que temos uma entidade prateleira em nosso escritório. Os nossos produtos são guardados nessas prateleiras.

Cada prateleira pode ter vários produtos, e como somos organizados, um produto sempre está somente em uma prateleira.

Sempre temos uma leitura em dois sentidos para validar que esse relacionamento seja efetivamente de um-para-muitos.

Cada ocorrência da entidade prateleira pode estar conectada a muitas ocorrências da entidade produto. Entretanto, cada ocorrência da entidade produto somente está conectada a uma e somente uma ocorrência da entidade prateleira.

Vamos analisar agora os relacionamentos que são de conectividade muitos-para-muitos utilizando a mesma situação, porém considerando que somos muito desorganizados e colocamos um produto em mais de uma prateleira.

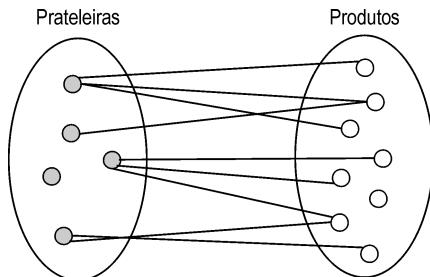
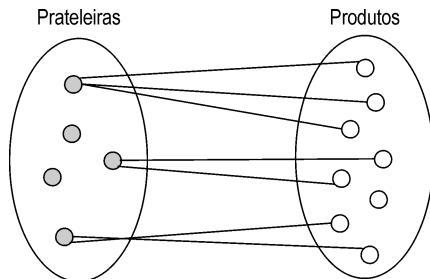
Conectividade Muitos-para-Muitos

Agora nos dois sentidos de leitura encontramos a conectividade de um-para-muitos. Uma prateleira possui muitos produtos e um produto pode estar em muitas prateleiras.

Quando encontramos nos dois sentidos de leitura a conectividade de um-para-muitos, temos então o que se denomina de conectividade muitos-para-muitos.

Um relacionamento com essa conectividade é normalmente um fato, acontecimento de negócios no mundo real.

Vejamos mais alguns exemplos. Essa conectividade é muito comum em negócios. Uma nota fiscal tem muitos produtos, e um produto pode estar em muitas notas fiscais. Este é um fato corriqueiro em qualquer empresa.



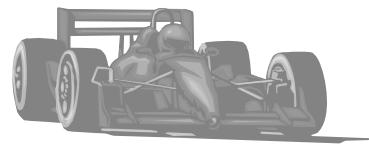
Neste caso podemos afirmar que é um relacionamento muito-para-muitos, pois antecipadamente realizamos a leitura nos dois sentidos, e encontramos nos dois a mesma conectividade de um-para-muitos.



Vamos abstrair e procurar no ambiente doméstico um relacionamento muitos-para-muitos.

Como somos fanáticos por corridas de Fórmula 1, vamos analisar esta situação.

O campeonato de Fórmula 1 tem muitas corridas durante o ano que assistimos confortavelmente instalados em frente ao nosso televisor. As equipes que disputam o campeonato possuem muitos carros, pilotos e mecânicos e esses carros e pilotos participam normalmente de todas as provas do campeonato.



Que modelo de dados podemos obter deste fato?

Também neste caso nos dois sentidos de leitura confirmamos a existência deste tipo de conectividade:

Cada carro de Fórmula 1 participa de muitos Grandes Prêmios.

De um Grande Prêmio participam muitos carros de Fórmula 1.



Nos dois sentidos obtivemos uma conectividade de um-para-muitos, que caracteriza um relacionamento ser de conectividade muitos-para-muitos.

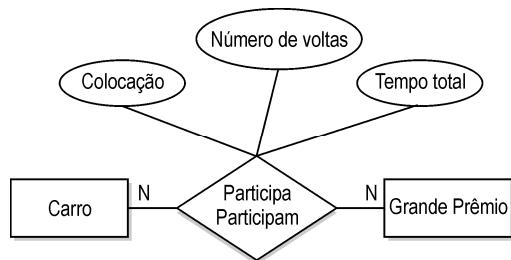
Atributos em um Relacionamento

Como estamos atuando no nível conceitual, temos a obrigação de observar que neste tipo de relacionamento de muitos-para-muitos normalmente ocorre a presença de atributos.

No caso das corridas de Fórmula 1, temos um atributo que é a posição de chegada de um carro, o tempo que levou para completar a prova, quantas voltas realizou (pode ser que tenha abandonado), só para ilustrar alguns atributos.

Por que estes atributos são do relacionamento?

Analisando, entendemos que o relacionamento representa o fato de um carro participar de um Grande Prêmio. Não são informações somente do GP nem tampouco somente de um carro. São referentes à participação do carro em um GP, logo são dados do relacionamento.

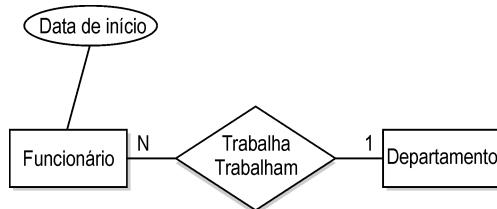


Atributos são normalmente assinalados em relacionamentos muitos-para-muitos. Nunca assinalamos atributos em relacionamentos um-para-um e em relacionamentos um-para-muitos porque no sentido de leitura de um relacionamento deste tipo sempre temos um lado que tem somente um elemento relacionado, e seria ambíguo colocarmos o atributo no relacionamento em vez de colocarmos na entidade. Por exemplo, no relacionamento entre funcionários e departamentos:



Poderíamos dizer que o atributo data de início de atividades pertence ao relacionamento, pois o funcionário começou a trabalhar em um departamento em uma determinada data. Como ele somente trabalha em um departamento e cada ocorrência de funcionário possui a sua data de início em um departamento, essa data é um atributo de funcionário.

Se essa data se referisse à data de início de atividades do departamento, ela seria referente a uma ocorrência de departamento, desta forma também não seria do relacionamento.



Sempre que temos relacionamentos de um-para-um ou um-para-muitos, os atributos que possam existir decorrentes do fato são elementos de descrição de uma das duas entidades.

Opcionalidade de Relacionamento

Analisando o modelo funcionário e departamento, vamos verificar a validade de algumas afirmativas:

- ▶ Todo funcionário trabalha em um departamento?
- ▶ Todo departamento tem funcionários?

Das duas afirmativas podemos dizer que nem todo departamento necessita ter funcionários, logo o relacionamento de departamento no sentido de leitura para funcionário é opcional. Isso quer dizer que podemos ter uma ocorrência de departamentos sem funcionários, ou seja, ainda não foram alocados funcionários para eles.

Como representar essa opcionalidade?

Esta situação quer nos dizer que a conectividade mínima é igual a 0 (zero).

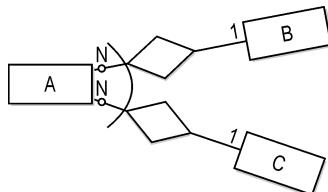
Um pequeno círculo no lado que existe a opcionalidade faz esta representação.



- ▶ Em um departamento trabalha nenhum, ou muitos funcionários.
- ▶ Um funcionário trabalha em no mínimo 1 (um) departamento.

Condisionalidade de um Relacionamento

Um relacionamento pode ser condicional ou ter restrições, ou seja, podemos ter um modelo em que uma entidade se relacione com outras duas, porém com cada uma exclusivamente. Vejamos uma situação para exemplificar.

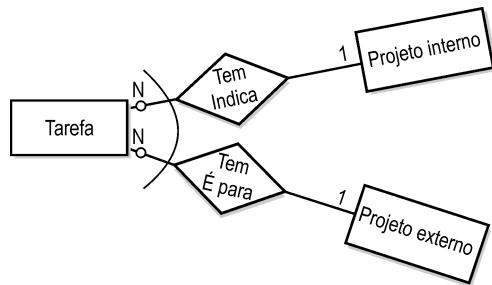


A entidade A se relaciona com a entidade B ou com a entidade C.

Isso quer dizer que cada ocorrência da entidade A pode se relacionar com uma ocorrência da entidade B ou então com uma ocorrência da entidade C, sendo esse relacionamento mutuamente exclusivo, isto é, não é possível existir um relacionamento com as duas ocorrências das entidades B e C.

O desenho torto é somente para ilustrar, pois certamente podemos realizar um *design* melhor esteticamente.

Uma tarefa "indica" um projeto interno, senão ela "é para" um projeto externo. Relacionamento mutuamente exclusivo.



Quando estudarmos os conceitos de generalização e especialização de entidades, vamos entender melhor quando acontecem esses casos.

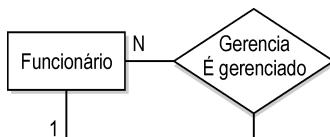
Relacionamentos Reflexivos

São os relacionamentos que acontecem entre as ocorrências de uma mesma entidade.

Normalmente esses relacionamentos representam algum sentido de hierarquia.

Por exemplo, a entidade funcionários representa todos os funcionários de uma empresa. Alguns desses funcionários gerenciam outros funcionários. Este é um autorrelacionamento, pois ocorre dentro da mesma entidade.

Observe a representação.



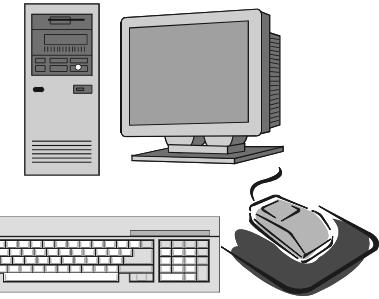
Os relacionamentos reflexivos podem ter qualquer uma das três possibilidades de conectividade, um-para-um, um-para-muitos e muitos-para-muitos.

Quando utilizamos estruturas que representam algum tipo de composição de algo, a conectividade é muitos-para-muitos.

Um computador, por exemplo, é entregue como composto por quatro produtos: monitor, CPU, teclado e mouse.

Vamos supor que temos uma entidade produto e nas suas ocorrências temos todos os produtos de nossa empresa, inclusive os que são compostos por outros produtos. Como representar essa composição?

Por meio de um relacionamento reflexivo de muitos-para-muitos.

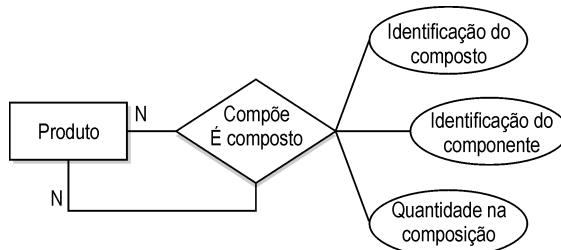


O relacionamento em si tem os atributos que identificam qual é o produto composto e qual é o produto que está participando de uma composição, assim como a quantidade com que ele participa.

São ocorrências distintas na mesma entidade.

A pergunta que você deve fazer é se todos os componentes de um produto composto estão em uma ocorrência do relacionamento. A resposta é não.

Cada componente forma uma ocorrência do relacionamento, logo se um produto é formado por dez outros produtos, existirão dez ocorrências no relacionamento indicando cada uma dessas ocorrências do produto composto e um dos componentes.



Peter Chen especifica a existência de um papel para cada elemento de entidade, participante de um relacionamento, mas é no autorrelacionamento ou relacionamento reflexivo que isso se torna mais evidente e necessário.

Usualmente esses papéis são denominados **ALIAS**.

Código Produto	Descrição
3758	Parafuso
8596	Rosca
4512	Arruela
5532	Bloco
7626	Carburador

Código Composto	Código Componente	Quantidade Componente
5532	3758	10
5532	4512	22
5532	8596	14
7626	3758	65
7626	4512	70

Para obter uma visão dos dados do relacionamento, vamos apresentá-lo no formato tabela.

Isso denota que o relacionamento reflexivo de conectividade muitos-para-muitos implica em uma estrutura de dados para ele, ou seja, ele possui dados da mesma forma que uma entidade.

Na estrutura de atributos do relacionamento existem dois papéis, o papel de produto composto e o papel de produto componente.

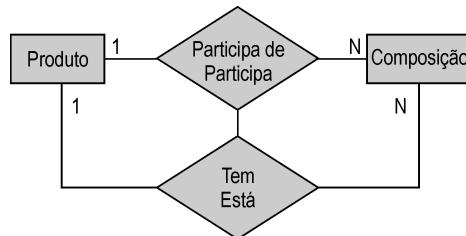
Resolução de Relacionamentos Muitos-para-Muitos

A interpretação de um relacionamento muitos-para-muitos pode variar na análise do minimundo que estamos modelando. Entretanto, podemos conciliar esses casos por meio do que denominamos resolver o relacionamento muitos-para-muitos.

Todo relacionamento muitos-para-muitos pode ser entendido como uma entidade. Essas entidades denominam-se associativas, pois elas representam um fato, um relacionamento muitos-para-muitos.

Considerando os modelos vistos até o momento, as alternativas são as seguintes:

Neste caso de autorrelacionamento, ou relacionamento reflexivo de muitos-para-muitos que apresentamos, a inclusão da entidade composição tem o mesmo efeito de representatividade que o relacionamento reflexivo.



Observe a leitura do modelo:

- ▶ Um produto (componente) participa de 'n' ocorrências de composição.
- ▶ Cada ocorrência de composição participa de um produto (componente).
- ▶ Cada produto (composto) está em 'n' ocorrências de composição.
- ▶ Cada ocorrência de composição tem um produto (composto).

No caso de relacionamentos reflexivos como este, são substituídos por dois relacionamentos um-para-muitos.

Observe a tabela da entidade associativa, idêntica ao relacionamento para um melhor entendimento.

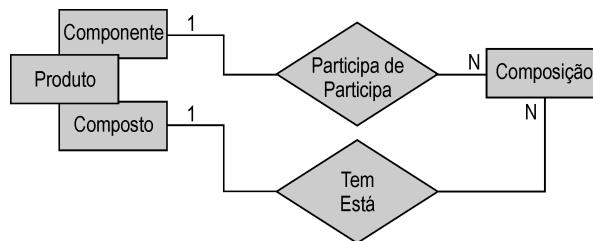
Código Composto	Código Componente	Quantidade Componente
5532	3758	10
5532	4512	22
5532	8596	14
7626	3758	65
7626	4512	70

- ▶ O produto 5532 (composto) ESTÁ na entidade associativa mais de uma vez, logo ele está relacionado "N" vezes com a entidade composição.
- ▶ Cada ocorrência de composição TEM somente UM produto composto.

Código Composto	Código Componente	Quantidade Componente
5532	3758	10
5532	4512	22
5532	8596	14
7626	3758	65
7626	4512	70

- ▶ O produto 3758, como destacamos, aparece mais de uma vez na entidade composição, logo UM produto participa de "N" composição.
- ▶ Cada ocorrência da entidade composição tem UM produto (componente).

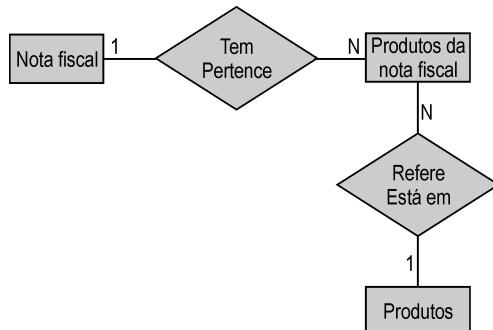
Peter Chen usa um formato de diagramação para representar os papéis que deixam um pouco mais clara essa definição do modelo.



Vamos ver agora o caso do exemplo que apresentamos de nota fiscal e produtos.



Poderia ser resolvido também como:



Procure agora visualizar como são os dados da entidade produtos da nota fiscal.

Nota Fiscal	Produto	Quantidade
2585999	Mouse	2
2585999	Teclado	5
2586000	Tela para Monitor	10
2586001	CD-RW	20
2586001	Mouse	5

- ▶ Toda ocorrência de produtos da nota fiscal tem um produto.
- ▶ Todo produto pode estar em mais de um produto da nota fiscal.
- ▶ Uma nota fiscal pode estar em mais de um produto da nota fiscal.
- ▶ Toda ocorrência de produtos da nota fiscal tem uma nota fiscal.

Vamos agora exercitar um pouco a descoberta de entidades e relacionamentos, assim como descobrir os atributos das entidades e relacionamentos.

Para este exercício vamos utilizar um minimundo, cuja descrição apresentamos em seguida:

Existem empresas identificadas univocamente pelo respectivo número de contribuinte (número do CNPJ). Adicionalmente, são caracterizadas por um nome e um segmento de negócio.

Cada empresa tem uma sede e zero, uma ou mais filiais. Tanto a sede como as filiais têm um endereço e um só número de telefone cada uma. Os empregados das empresas são identificados univocamente por uma matrícula de empregado. São ainda caracterizados pelo respectivo nome, endereço e data de nascimento. Cada empregado trabalha apenas num local (sede ou filial); no seu local de trabalho, existe um número de telefone direto.

Bem, vamos observar o que é candidato ou quais são as entidades existentes neste contexto.

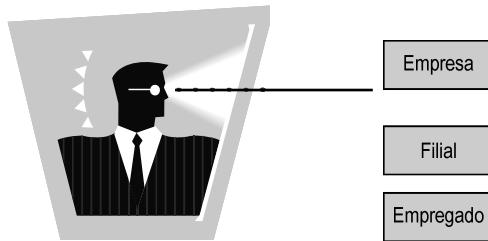
Seguindo um critério de que substantivos indicam entidades, temos neste caso:

- ▶ Empresa
- ▶ Filial
- ▶ Número de contribuinte
- ▶ Número de telefone
- ▶ Endereço
- ▶ Empregados
- ▶ Matrícula de empregado
- ▶ Local de trabalho

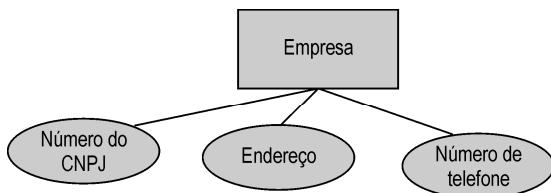
Nesta lista temos dois substantivos compostos que indicam atributos de alguma entidade e não propriamente entidades, pois não têm existência própria: número de telefone e matrícula de empregado.

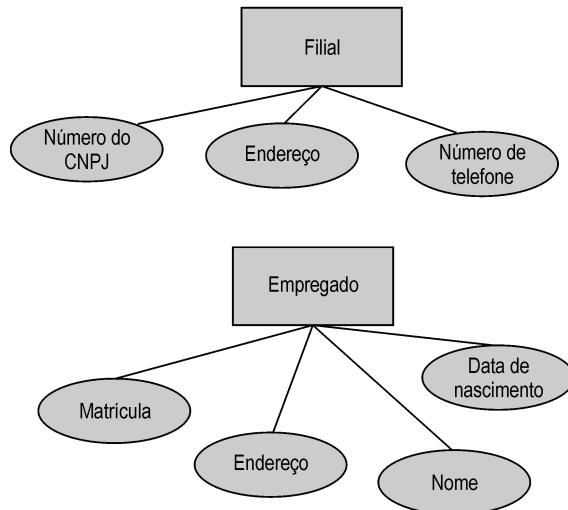
Analizando mais, podemos concluir que endereço é uma propriedade de algo, também não tem existência própria, logo deve ser atributo de alguma entidade e não uma entidade. Local de trabalho também é composto e claramente caracteriza-se como uma propriedade de uma entidade e uma entidade em si.

Logo existem então três entidades neste contexto:



Vamos analisar os atributos de cada uma dessas entidades que podemos entender do minimundo.



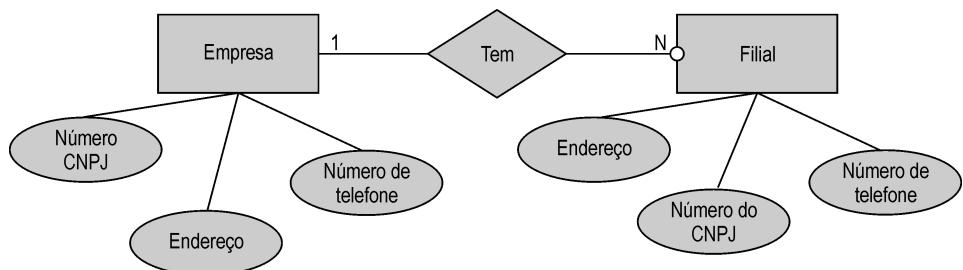


Após termos obtido e identificado as entidades que compõem o contexto do aplicativo que estamos projetando, a próxima etapa consiste na definição dos relacionamentos que existem entre as entidades e que interessam aos propósitos. No momento em que obtivermos o domínio dos relacionamentos e entidades, poderemos traçar um primeiro diagrama ER que servirá de base para as etapas seguintes.

Uma vez que temos as entidades definidas para o caso, vamos analisar os verbos que existem no minimundo e quais fatos descritos podem caracterizar a existência de relacionamentos no modelo de dados.

"Cada empresa possui uma sede e zero, uma ou mais filiais."

Aqui temos claramente a visão de que existe um relacionamento entre empresa e filial, e ainda está indicada a conectividade (cardinalidade) desse relacionamento e sua opcionalidade.

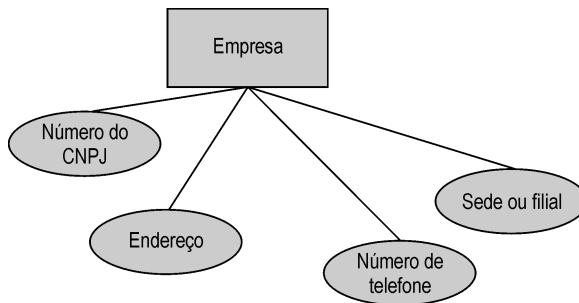


O diagrama agora já apresenta estas duas entidades com o seu relacionamento.

Seguindo, devemos analisar com quem a entidade Empregado se relaciona, porém um detalhe até o momento chama a atenção:

Empresa e filial têm os mesmos atributos.

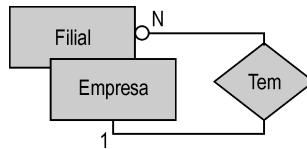
Não são uma única entidade? Ser filial ou sede não será um atributo, uma propriedade dessa entidade?



Vejamos se esta solução não é plausível:

- Quando a empresa for sede, o atributo sede ou filial indicará um valor 'sede'. Quando for uma filial, o valor do atributo será 'filial'.

Desta forma conseguimos simplificar um pouco o nosso modelo, porém como devemos saber quem é filial de quem, vamos colocar um relacionamento reflexivo no modelo.

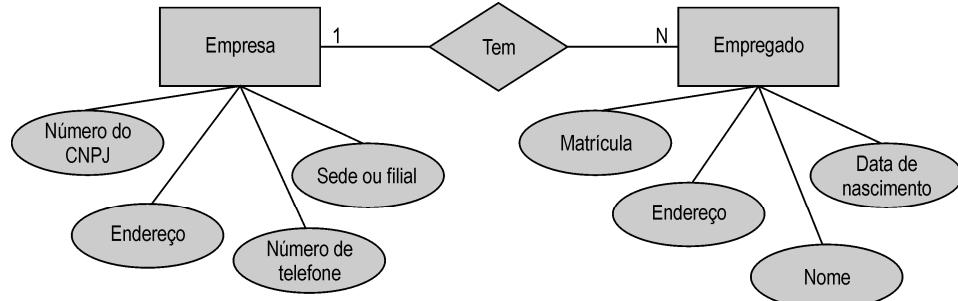


Voltando ao raciocínio de continuidade, qual o relacionamento entre empregado e a entidade empresa?

"Cada empregado trabalha apenas num local (sede ou filial); no seu local de trabalho tem um número de telefone direto."

Logo temos um relacionamento.

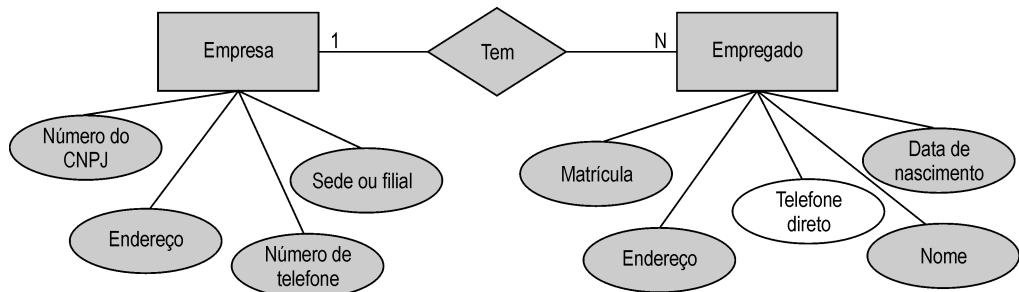
Resolvendo, temos:



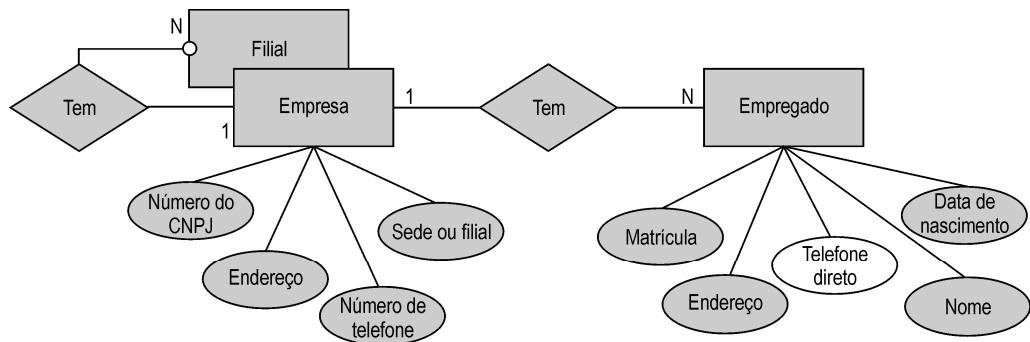
Porém, ainda falta alguma coisa para que este modelo fique correto:

".....no seu local de trabalho tem um número de telefone direto."

Este atributo de o empregado ter um telefone em seu local de trabalho não está visível no modelo até o momento, pois não é o telefone da empresa. Logo:



Vamos agora apresentar o modelo completo segundo a notação de Peter Chen.



Ao analisarmos o contexto, devemos ter em mente uma forma de pensar ER.

Logo, algo que não requeira pelo menos dois atributos para descrevê-lo provavelmente não caracterizará uma entidade e sim um atributo de alguma outra entidade, ou será simplesmente uma variável do contexto.

É importante estabelecermos alguns princípios para que obtenhamos (a partir dos levantamentos e entrevistas com os usuários e na construção do minimundo) as entidades de interesse para a aplicação que está em projeto.

O usuário sempre se refere a processo quando nos descreve uma realidade em que estão situadas suas atividades.

Pense e lembre que o modelo de dados não é fluxograma, logo não tem começo nem fim. Esteja atento a referências a substantivos, pois pode ser a indicação de uma entidade.

Entidade Fraca ou Dependente

É uma entidade que não possui existência própria individualizada no ambiente sem que seja dependente da existência de uma entidade dita forte à qual ela está relacionada.

A entidade fraca no modelo lógico não possui chave primária, pois por definição é uma entidade subordinada. Para formarmos a chave primária de uma entidade fraca, utilizamos a chave primária da entidade forte da qual ela é existencialmente dependente, mais o conjunto mínimo de atributos que possa identificar uma entidade em um conjunto de entidades fracas.

Por exemplo, no relacionamento que apresentamos entre as entidades funcionário e histórico de cargos, podem ser modeladas respectivamente como entidades forte e fraca.

Isso porque a entidade funcionário (matrícula, nome, endereço) tem por natureza atributos que são chaves candidatas e, por consequência, possui uma chave primária.

Já a entidade Histórico de cargos (data, cargo) não possui um conjunto de atributos que possa identificar cada uma de suas instâncias.

Desta forma, para montarmos a chave primária de histórico de cargos, usamos a chave primária da entidade funcionário, definindo assim a entidade histórico de cargos (matrícula, data, cargo).

Como Reconhecer Entidades na Prática

Observe em seguida uma lista de perguntas úteis para identificar entidades em um contexto:

- ▶ Que coisas são trabalhadas?
- ▶ O que pode ser identificado por número, código?
- ▶ Essa coisa tem atributos? Esses atributos são relevantes, pertinentes?
- ▶ Essa coisa pode assumir a forma de uma tabela?
- ▶ É um documento externo (recibo, fatura, nota fiscal)? Se sim, é forte candidato à entidade.
- ▶ Tem significado próprio?
- ▶ Qual a entidade principal do contexto?

Dicas:

- ▶ Substantivos que não possuem atributos podem ser atributos de outras entidades.
- ▶ Adjetivos colocados pelos usuários indicam normalmente atributos de uma entidade.
- ▶ Verbos indicam prováveis relacionamentos.
- ▶ Advérbios temporais indicam prováveis atributos de um relacionamento.

- ▶ Procure sempre visualizar a entidade principal do contexto sob análise.
- ▶ Entidades cujo nome termine por "ento" ou por "ão" geralmente são procedimentos.

Como Reconhecer Relacionamentos

Após obter e identificar as entidades que compõem o contexto de aplicativo que estamos projetando, a próxima etapa consiste na definição dos relacionamentos que existem entre as entidades e que interessam aos nossos propósitos.

No momento em que obtivermos o domínio dos relacionamentos e entidades, podemos traçar um primeiro diagrama ER que servirá de base para as etapas seguintes.

Dicas para reconhecer e inserir relacionamentos no modelo:

- ▶ O relacionamento é necessário?
- ▶ Ele é útil?
- ▶ É redundante?
- ▶ Se redundante, retirar?
- ▶ Qual a sua finalidade? (Documentar)
- ▶ Verbos indicam possíveis relacionamentos.
- ▶ Analisar sempre as entidades aos pares.

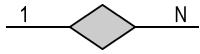
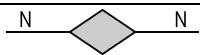
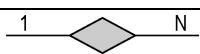
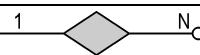
Veja que a notação de Peter Chen é interessante e bastante expressiva, porém quando temos modelos com grande quantidade de entidades e relacionamentos, o espaço ocupado no diagrama pela representação desse relacionamento torna o modelo sensivelmente confuso, com muitos cruzamentos de linha e complicado de ser lido apesar da expressividade dos relacionamentos por meio de losangos.

As ferramentas CASE de mercado não disponibilizam esse tipo de notação para diagramas de Entidade-Relacionamento. A maior parte delas utiliza a notação da engenharia de informações, mais conhecida e divulgada.

Vamos então apresentar neste ponto como é essa notação, para que possamos seguir utilizando-a neste livro.

Chamada de notação da engenharia de informações ou notação James Martin. Para a engenharia de informação (método de desenvolvimento de sistemas de informação) foi definida a seguinte notação gráfica para o que apresentamos até o momento neste livro:

Comparativo entre Notação Peter Chen e Engenharia de Informação

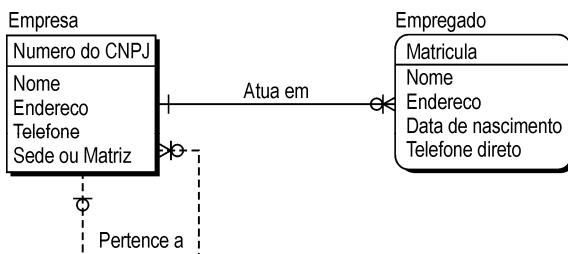
Conectividade	Engenharia de Informação	Peter Chen
1:1	_____	
1:N	_____<	
N:N	>_____<	
Existência		
Obrigatório	_____!<	
Opcional	>-----<	

É simples trabalhar com este tipo de notação, principalmente se pertermos o foco na expressão do relacionamento que pode ser escrita sobre a linha do relacionamento.

Mais importante que descobrir e relacionar os atributos de cada entidade é assinalar as entidades e seus relacionamentos independentemente de atributos quando trabalhamos com modelos conceituais.

Deixe para um segundo momento a análise de atributos.

Outro detalhe na engenharia de informação é que a notação que é a entidade é representada da mesma forma que a notação de Peter Chen, quando tratamos somente da entidade. Entretanto, quando modelamos a entidade de atributos, essa notação passa a ser muito parecida com a de Rumbaugh. Vejamos então como fica o modelo com essa notação:



Para continuarmos a fixar o apresentado até este ponto do livro, vamos estudar outra situação de contexto e realizar o modelo de dados necessário à sua solução.

Uma Distribuidora de Filmes

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

- ▶ A empresa de distribuição possui vários cinemas em diversas localidades.
- ▶ Cada cinema possui uma identificação única (código do cinema), 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.
- ▶ Os cinemas apresentam os mais diversos filmes durante a sua existência. Cada apresentação de filme tem uma data de início de uma data de fim de exibição, que corresponde ao último dia do filme em cartaz.
- ▶ Cada filme é registrado com um título original, e se for 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.
- ▶ Um filme pode ser exibido simultaneamente em vários cinemas de diversas localidades.
- ▶ 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. Em cada filme o ator representa um papel ou personagem. Um ator tem as seguintes características: um nome, uma nacionalidade, sexo e uma idade.

Seguindo as dicas para identificação de entidades, temos neste contexto as seguintes entidades:



Observando os critérios no texto que define o minimundo, encontramos os substantivos:

- ▶ Cinema, Ator, Filme.

Estes elementos têm significado próprio, logo podem ser caracterizados como entidades.

Filme tem 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 tem atributos próprios que o caracterizam, além de ser fácil imaginar a existência de diversos cinemas. O mesmo critério vale para ator.

Vamos analisar e descobrir os relacionamentos existentes por pares de entidades:

Cinema e filme têm algum relacionamento entre elas?

A resposta correta é sim, pois cinema exibe filme.

Logo temos um relacionamento existente entre cinema e filme.

Representando pela notação de Chen:



Porém, temos agora que determinar a conectividade desse relacionamento.

Um cinema exibe quantos filmes?

Quantos filmes você lembra que foram exibidos em um cinema perto de você? (Credo, até parece comercial.)

Um modelo de dados é espacial em relação a tempo. As entidades e os relacionamentos não são temporais, devem ser analisados em um espaço de tempo indeterminado. As ocorrências do relacionamento são temporais, mas o relacionamento não, ele é espacial.

Logo, um cinema exibe muitos filmes, mas lendo esse relacionamento no sentido contrário de filme para cinema, o que obtemos?

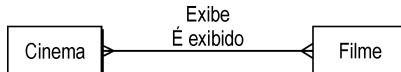
Um filme é exibido em quantos cinemas? Em muitos, é claro!

Então um filme é exibido em muitos cinemas.

Isso faz com que nosso modelo seja na notação de Peter Chen:



Ou na notação de engenharia de informação:



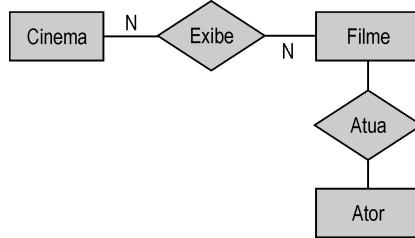
Agora vamos analisar se existe algum relacionamento entre filme e ator.

Como diz o texto:

"Os atores de um filme podem, obviamente, atuar em diversos filmes..."

Logo o relacionamento de ator é com filme, pois na vida real sabemos que os atores não estão no cinema e sim nos filmes aos quais assistimos. Claro que eventualmente eles podem aparecer no cinema (vão assistir a uma estreia), mas não é nesse contexto que estamos trabalhando.

Então temos:

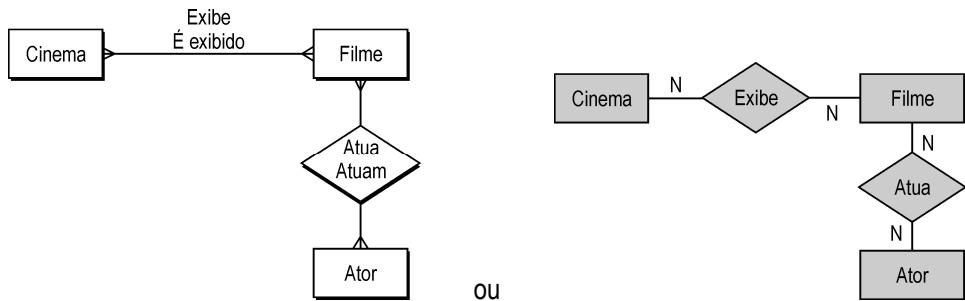


Devemos agora descobrir a conectividade entre a entidade ator e a entidade filme.

"Os atores de um filme podem, obviamente, atuar em diversos filmes... "

A afirmação do contexto é clara, e podemos concluir pela vida real que um filme também tem vários atores, apesar de existirem monólogos.

Logo, nosso modelo agora fica com o seguinte conteúdo:



Podemos agora partir para analisar se existe mais alguma entidade nesse contexto. Seria diretor uma entidade?

"Existirá um único diretor para cada filme."

Com esta afirmativa podemos concluir que cada ocorrência de filme tem uma informação de diretor e um diretor pode estar informado em mais de uma ocorrência de filme, já que sabemos no mundo real que vários filmes são dirigidos pelo mesmo diretor.

Isso caracteriza que diretor até o momento é um atributo de cada ocorrência de filme, não se caracterizando como uma entidade com significado próprio, pois ainda não conseguimos determinar atributos claros que o descrevam.

Vamos agora colocar atributos nesse modelo para deixá-lo mais legível.

Para cinema temos:

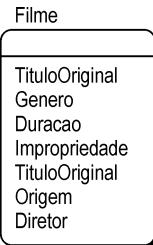
"Cada cinema possui uma identificação única (código do cinema), um nome fantasia, um endereço completo, incluindo rua, avenida, bairro, município, estado e sua capacidade de lotação."

Cinema
Codigo Cinema
Nome Fantasia
Rua
Cidade
Bairro
Estado
Lotação

Para filmes temos:

"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."

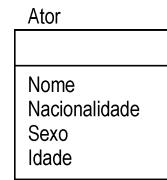
As informações sobre os atores de um filme, ou seja, quem participa do filme, podem ser colocadas no relacionamento entre os dois, uma vez que é um relacionamento de muitos-para-muitos.



Para ator temos:

"Um ator possui as seguintes características: um nome, uma nacionalidade, sexo e uma idade."

Entretanto, podemos analisar a afirmativa seguinte e considerar que diretor tem os mesmos atributos de ator, pelo menos até o momento.



Concluindo o que apresentamos até este ponto do livro, este seria o modelo obtido do ponto de vista conceitual.

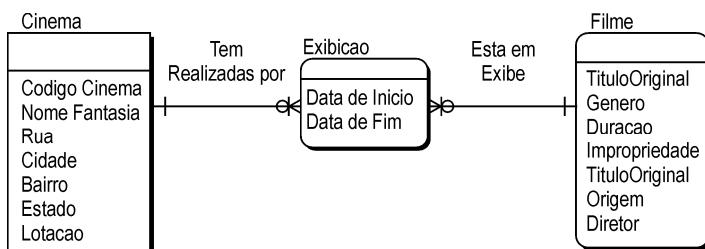
Este modelo de dados deve ser considerado conceitual inicial, pois existem extensões do modelo ER que podem ser aplicadas a ele.

Para que ele fique mais claro, vamos resolver os relacionamentos muitos-para-muitos.

"Cada apresentação de filme tem uma data de início e uma data de fim de exibição, que corresponde ao último dia do filme em cartaz."

Esta afirmação caracteriza atributos do relacionamento exibe.

Logo, data de início e data de fim são atributos da entidade associativa exibe que vai substituir o relacionamento muitos-para-muitos.



Vamos analisar o relacionamento entre ator e filme que também é muitos-para-muitos.

"Em cada filme o ator representa um papel ou personagem."

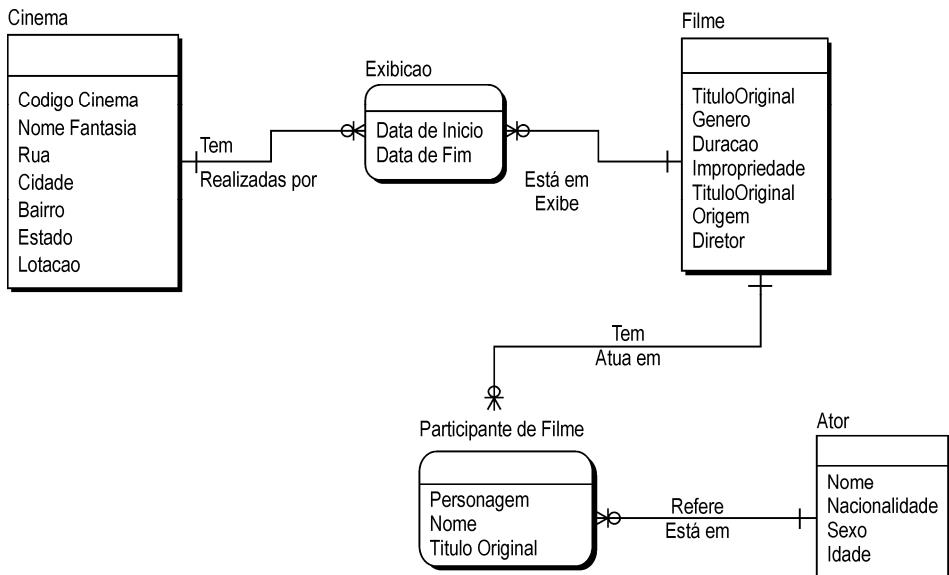
Logo temos um atributo que é do relacionamento entre ator e filme que será agora da entidade associativa que denominaremos de elenco de filme.

Participante de Filme

Personagem
Nome
Titulo Original

Mas como temos de associar e saber quem é o ator e qual o filme, vamos colocar um par desses dados em cada ocorrência de elenco para ficar claro quem é quem.

Nosso modelo então fica completo até o momento:



Este modelo não pode ainda ser considerado definitivo, pois como já citamos, existem extensões ao modelo ER proposto por Peter Chen, as quais podem ser aplicadas a este modelo, deixando efetivamente mais claro e correto. No próximo capítulo vamos estudar essas extensões e aplicá-las no próximo exercício.

Exercícios

Realize o diagrama entidade-relacionamento para os contextos apresentados.

1. Academia de ginástica

Os clientes podem frequentar quantas aulas desejem.

As aulas são identificadas por um número sequencial relativamente à modalidade e são caracterizadas pelo nível de dificuldade, professor e sala.



Os professores são identificados pelo nome e sobre eles é ainda guardado o sobrenome e o telefone.

Sobre os alunos são conhecidos o nome, idade, sexo, instituição (caso pertença a uma instituição que o beneficie de desconto) e telefone.

2. Empresa de transportes

Uma companhia de transportes chamada Carregaki é responsável por transportar encomendas de um conjunto de armazéns até um conjunto de lojas de uma cadeia denominado Venda Tudo.

Atualmente existem seis armazéns e 45 lojas.

Um caminhão pode transportar várias encomendas em cada viagem, identificada por um número de viagem, descarregando em diferentes lojas.



Cada encomenda é identificada por um número e inclui a data da encomenda, o peso, o destino etc.

Os caminhões têm diferentes capacidades, quer relativas ao volume ou ao peso que podem transportar.

A empresa Carregaki detém, de momento, 150 caminhões e cada caminhão tem a capacidade de fazer três a quatro viagens por semana.

Pretende-se projetar um banco de dados, utilizado por ambas as empresas, que mantenha a informação sobre a utilização dos caminhões e distribuição de encomendas, e que possa auxiliar na definição dos horários de distribuição.

3. Revenda de veículos

A AutoStand, que comercializa veículos novos e usados, deseja um banco de dados para gerir a sua informação, que descreveu da seguinte forma:



Sobre o veículo:

- ▶ Matrícula (identificador)
- ▶ Marca e modelo (tabela)
- ▶ Data de matrícula
- ▶ Tipo de veículo: esportivo, passeio, utilitário, transporte de mercadorias etc. (tabela)
- ▶ Cor (tabela)
- ▶ Localização: indicação do local do veículo (que pode ser um estande, um armazém ou uma oficina) (tabela)

- ▶ Data de entrada na AutoStand
- ▶ Indicação de novo ou usado
- ▶ Cilindrada
- ▶ Potência
- ▶ Quilometragem

Sobre o cliente:

- ▶ Nome, endereço, localidade, telefone, CPF, RG
- ▶ Classificação como regular ou frotista
- ▶ Vendedor associado

Sobre o fornecedor e proprietário anterior:

- ▶ Nome, endereço, localidade, telefone, fax, CPF, RG

Sobre os tipos de custos:

- ▶ Custos associados ao veículo (funilaria, pintura, eletricista, combustível, transporte, despesas de legalização, despesas de importação, manutenção etc.)

Sobre os valores:

- ▶ Valor de compra do veículo
- ▶ Indicação do valor pelo qual se pretende vender o veículo
- ▶ Indicação dos custos atribuídos ao veículo (necessários para o cálculo do custo do veículo)
- ▶ Preço da venda, data da venda e condições

Notas: É importante guardar as vendas efetuadas por cada vendedor com as respectivas datas e comissões sobre as vendas para cálculo do prêmio mensal dos vendedores. Os veículos novos são disponibilizados por um fornecedor e os veículos usados são comprados do seu proprietário anterior.

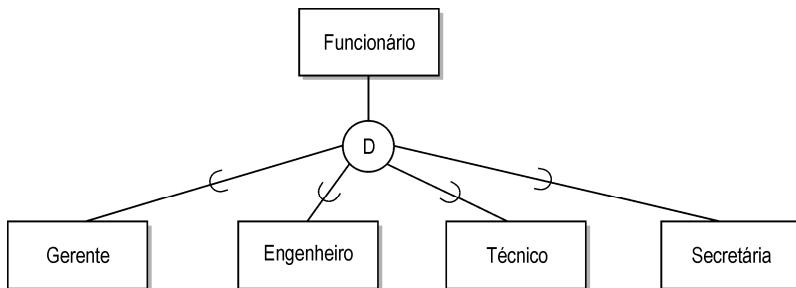
Importante: Elabore esse modelo com calma e não se preocupe com a quantidade de entidades que serão descobertas. Siga os roteiros e dicas apresentadas e você o concluirá com sucesso.

EXTENSÕES DO MODELO ENTIDADE-RELACIONAMENTO

O modelo de dados Entidade-Relacionamento, como proposto por Peter Chen, tem sido usado para a comunicação do usuário final apresentando entidades e relacionamentos. Entretanto, quando usado para integrar diversos modelos conceituais com diferentes usuários finais, fica severamente limitado até que se utilize um conceito de abstração de dados denominado generalização.

Generalização: Supertipos e Subtipos

Relembrando o capítulo em que foram apresentados os conceitos de abstração em modelagem de sistemas, a generalização ocorre quando se define um subconjunto de relacionamentos entre duas ou mais classes de dados.



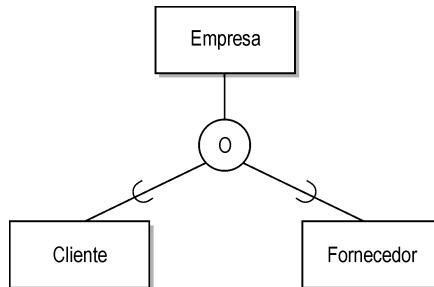
Quando analisamos a entidade funcionários, conforme o enfoque e o ponto de vista de cada grupo de usuários finais, observamos que essa classe de dados que definimos como a entidade funcionários é generalização de outras classes, tais como gerentes, engenheiros, secretária, técnicos em sistemas etc.

Na realidade, a generalização ocorre quando entidades que possuem atributos em comum são generalizadas em alto nível como uma entidade só, como uma entidade genérica ou uma superclasse de dados. As entidades de nível mais baixo que fazem parte desse supertipo de entidade são denominadas de subtipos, e elas refletem a especialização de partes da entidade supertipo.

A simbologia para representar o relacionamento entre a entidade supertipo e a entidade subtipo é muito variada. Apresentaremos a simbologia utilizando um pequeno círculo na intersecção das linhas que levam do supertipo até os subtipos, com uma indicação em seu interior.

Essa indicação objetiva distinguir quando a especialização é exclusiva, ou seja, no nosso exemplo, quando um funcionário é engenheiro, ele não é nenhum dos outros subtipos, ou não pertence a nenhum deles.

Existem casos em que a especialização pode não ser exclusiva, existindo *overlapping* das entidades subtipos dentro da entidade supertipo.

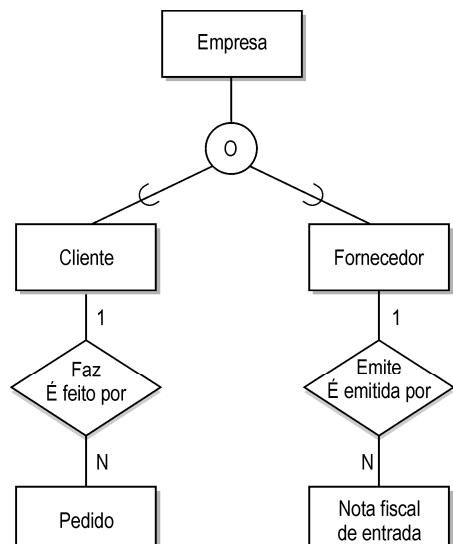


Observe que dentro do círculo do primeiro exemplo colocamos uma letra D, indicando *disjoint* (disjunção) subtipos, o que significa que são subtipos exclusivos, e no segundo exemplo colocamos a letra O, que significa *overlapping* de subtipos.

Quando temos a situação de *overlapping* de subtipos, uma ocorrência da entidade supertipo no caso empresa, pode ser cliente ou fornecedora, dependendo do relacionamento com outras entidades existentes em um modelo de dados.

No exemplo ao lado, quando existe o relacionamento entre nota fiscal de entrada e a entidade empresa, ele é estabelecido com o subtipo fornecedor. E quando existe o relacionamento com pedido, é estabelecido com a entidade subtipo cliente.

Desta forma conseguimos apresentar uma semântica que possibilita ao usuário final entender e discutir o modelo de dados, mesmo existindo áreas de negócios distintas e com usuários finais distintos, com conceitos distintos.



Agora vamos imaginar uma situação em que temos uma entidade de produto de software, e sabemos que programas e manuais compõem cada pacote de produto de software.

Como entender esse relacionamento ou essa generalização?

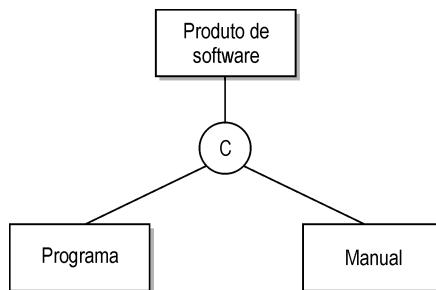
Em primeiro lugar não podemos denominar de generalização este caso, pois a entidade programa e a entidade manual não possuem atributos em comum.



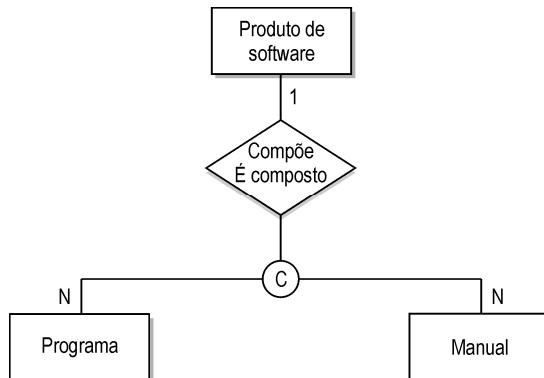
Algumas literaturas denominam de agregação de entidades, porém não vamos utilizar este termo, pois apresentaremos agregação em capítulo à parte, explorando os seus conceitos corretamente.

Vamos denominar de composição, apesar de em abstrações termos estudado que quando uma classe de dados é composta de várias outras, estamos agregando coisas.

Analisando, então, temos que cada ocorrência de produto de software é composta de muitos programas e também de muitos manuais.



Veja que este caso resolve-se simplesmente com dois relacionamentos. É importante destacar que o principal neste momento é entender que estamos utilizando dois relacionamentos para desenhar a composição de um determinado produto, uma vez que os atributos de manual e programa são completamente diferentes neste exemplo.



Utilizamos no primeiro diagrama o círculo de generalização com uma letra C, indicando tratar-se de uma composição de entidades relacionadas, mas num diagrama ER devemos representar os relacionamentos na forma tradicional, exceto pela indicação de que os dois relacionamentos são obrigatórios.

Optamos então por uma simbologia mista que permita expressar esse entendimento de forma simples.

Em realidade existe um só relacionamento no modelo conceitual, pois as duas entidades têm o mesmo relacionamento com a entidade produto de software.

Não pense agora nos aspectos físicos dessa implementação para não atrapalhar o seu raciocínio lógico e conceitual.

Leia-se produto de software é composto de muitos programas e muitos manuais.

Esta é uma interpretação conceitual do diagrama Entidade-Relacionamento e permite apresentar um diagrama de entendimento simples para o usuário final.

Vamos voltar ao exercício de cinemas que estávamos fazendo no capítulo anterior para visualizarmos uma aplicação de generalização e especialização.

"Existirá um único diretor para cada filme."

"Os atores de um filme podem, obviamente, atuar em diversos filmes, assim como o diretor de um filme pode também ser ator neste filme ou ainda mais, ser ator em outro filme. Em cada filme o ator representa um papel ou personagem. Um ator possui as seguintes características: um nome, uma nacionalidade, sexo e uma idade."

A afirmativa de um diretor também pode ser ator permite que visualizemos a seguinte realidade:

Pessoas podem atuar em filmes, neste caso são atores, ou podem dirigir filmes, ou fazer os dois simultaneamente.

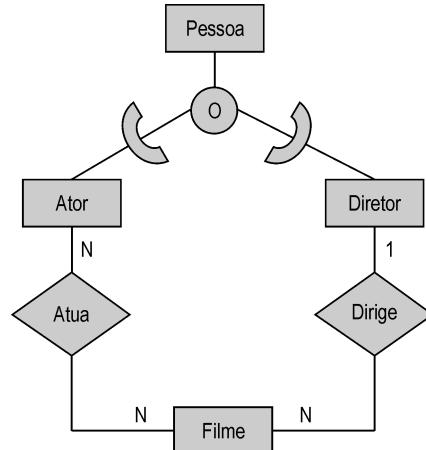
Logo, podemos aplicar a generalização e substituir a entidade ator do modelo anterior por uma entidade, digamos, generalista pessoa.

A entidade pessoa relaciona-se com filme, criando o papel ator quando ela está associada pelo fato de atuar em um filme e relaciona-se com filme da mesma forma por outro fato que é dirige um filme, criando o papel (alias) diretor.

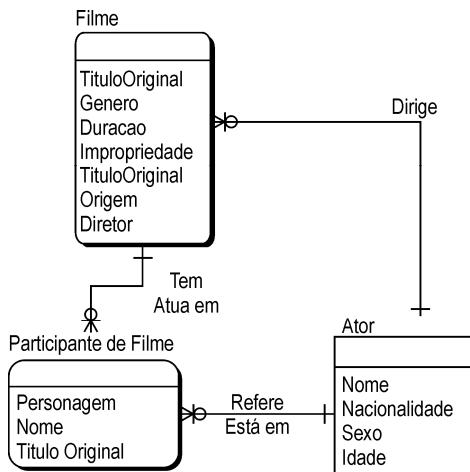
Então sabemos que diretor e ator são papéis de uma mesma entidade, pessoa. Há uma generalização com *overlapping*, sobreposição de papéis.

Vamos então olhar nosso modelo e ver como ele ficaria com esse tipo de solução no tocante a ator, diretor e filme.

Na representação de engenharia de informação temos uma única entidade com dois relacionamentos distintos.



Desta forma agora estão unificadas em uma só entidade tanto as informações de atores como de diretores, já que os dois papéis podem existir em relação a um mesmo filme. Não são redundantes as informações de determinada pessoa que atua e dirige um filme, por exemplo.



Relacionamentos Ternários

São utilizados quando os relacionamentos binários não conseguem semântica para descrever uma associação de três entidades.

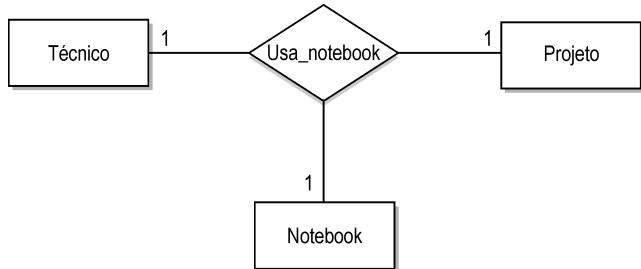
Não acreditamos muito profundamente nesta afirmativa, mas como em nosso primeiro livro, também vamos manter a apresentação, pois pode ser que um dia algum leitor venha em nosso socorro e apresente efetivamente uma utilização real para os relacionamentos ternários.

Em alguns casos podemos dizer que até é possível a utilização e o acontecimento de algum fato ternário, mas como contraria totalmente o princípio da polarização binária da forma de pensar do ser humano, fica difícil aceitá-los.

Um exemplo apresentado em muitas literaturas é o que está ilustrado a seguir entre técnico, projeto e notebook.

Vamos apresentar esta solução e a nossa, contestando a utilização de relacionamento ternário.

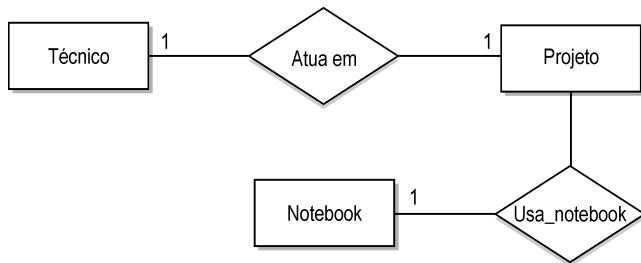
A explicação para este modelo apresentada em outros livros é que um técnico usa exatamente um notebook para cada projeto.



Cada notebook é utilizado por um técnico em cada projeto. Se o técnico estiver em outro projeto, vai utilizar outro notebook.

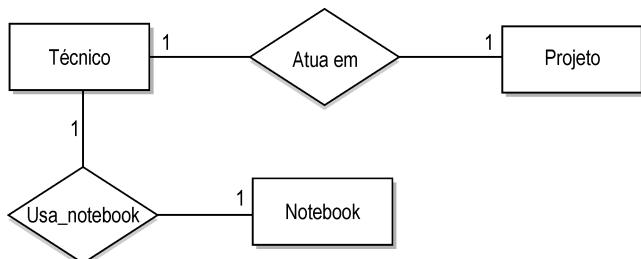
A afirmação invalida esta solução, pois teríamos um lado muitos nesse relacionamento.

Se imaginarmos que cada um dos projetos somente tem um técnico, pois são projetos individuais, estamos colocando o técnico em mais de um projeto, logo o lado de projeto em relação a técnico seria muitos. E se um projeto somente tem um técnico e no projeto ele usa somente um notebook que somente é utilizado nesse projeto, o relacionamento correto de notebook seria com projeto. É um retrato binário da situação.



Se nós mantivermos a afirmação de que a relação entre técnico e projeto é de um-para-um, tanto faz sentido ligarmos o notebook a projeto quanto se ligarmos a técnico.

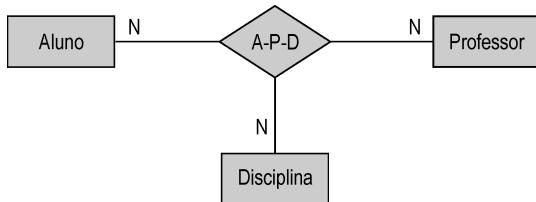
Acreditamos que é um erro de interpretação procurarmos montar estruturas ternárias ou múltiplas para retratar coisas de um ambiente que, por realidade, se relacionam sempre binariamente.



Como veremos no capítulo sobre agregação de relacionamentos, o que podemos ter é um relacionamento dependente da existência anterior de outro relacionamento.

A síntese do pensamento humano está na paridade de raciocínio e entendimento das coisas de nosso universo e do ambiente que analisamos. Sempre existem dois caminhos ou duas opções, além de que sempre é mais simples dividir para entender, ou explicar.

Vamos apresentar outro exemplo oriundo de literaturas diversas que mostra um relacionamento múltiplo ou ternário. É o relacionamento existente entre alunos, professores e disciplinas, clássico da literatura de modelagem de dados.



Observe que não conseguimos um verbo para explicitar o relacionamento. Como denominar um fato que acontece com três coisas simultaneamente?

Dentro da técnica de modelagem de dados isto é considerado válido, apesar do absurdo de nada representar, e ser efetivamente uma solução criada para visão física dos dados, sem preocupação conceitual de retratar o real.

Vamos agora começar a estudar a modelagem dos atributos das entidades e suas implicações nos relacionamentos que aprendemos até o momento, considerando que estamos modelando *relacionalmente*, ou seja, para banco de dados relacional, passamos agora a tratar também no nível de modelo lógico de dados.

Modelagem de Atributos (Modelagem Lógica)

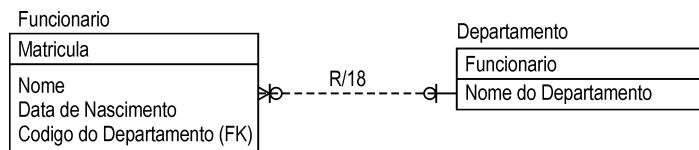
Quando estamos trabalhando com entidades e relacionamentos, devemos manter os princípios relacionais e suas regras de integridade, a integridade de identidade e a integridade referencial, pois as nossas entidades vão se transformar em tabelas relacionais e devemos visualizá-las como tais no modelo lógico.

A partir deste momento vamos iniciar a utilização também da notação gráfica para modelo ER lançada por Gordon Everest, em 1986, que elimina o losango que representa o relacionamento e simplesmente escreve sobre a linha que une as entidades a nomenclatura do relacionamento, além de utilizar os famosos "pés de galinha" para identificar a conectividade de cada lado de um relacionamento. Vamos manter ainda em conjunto a notação de Chen por ser em alguns casos mais claro o seu entendimento.



A razão para apresentarmos este outro tipo de notação justifica-se, como já citamos, pelas ferramentas CASE que dela se utilizam em sua grande maioria.

Outra característica desta notação é que, seguindo alguns preceitos de Rumbagh, os modelos apresentam no interior do retângulo da entidade a relação dos atributos que o descrevem, inclusive com suas chaves primárias em destaque. Na parte superior da caixa da entidade estão os atributos que compõem a chave primária, e os outros atributos na parte inferior da caixa, sendo destacadas as chaves estrangeiras como FK (foreign key).



Observe que o mesmo modelo anterior agora possui somente uma linha pontilhada entre as duas entidades, explicitando o relacionamento.

Como utilizamos um software CASE para este exemplo e não demos nome ao relacionamento, o software inseriu um código (R/18) para representar tal relacionamento.

Outra característica que se pode perceber é que, além das chaves primárias das entidades, o modelo apresenta as chaves estrangeiras, que realizam a efetivação lógica do relacionamento em banco de dados relacional, transformando o modelo conceitual em modelo lógico.

- Muitos / Mínimo = 0
- ◀ Um / Mínimo = 0
- └ Um / Mínimo = 1
- ↗ Muitos / Mínimo = 1

Este símbolo de mínimo zero indica que o relacionamento é opcional no sentido em que apresenta o símbolo.



O importante a destacar é que todas as entidades que modelamos não possuem chave estrangeira no mundo real. Ela é colocada para que se possibilite a efetivação lógica dos relacionamentos de acordo com os princípios relacionais.

Da mesma forma nem todas as entidades que serão encontradas possuem chave primária evidente, sendo muitas vezes necessária a inserção de um atributo código, como realizamos em departamento.

É importante a modelagem de atributos de chave primária, pois eles nos fornecem as restrições lógicas do mundo real para a existência das ocorrências de uma entidade.

Em seguida apresentamos um modelo ER maior para exercitar seu entendimento, pois apresentamos o problema em si e a solução adotada com um modelo ER.

O Problema

Um negócio imaginário que chamaremos de *softball* compreende a participação de clubes de *softball* em diversos campeonatos que são realizados por confederações.

Cada clube pertence a uma confederação somente, entretanto participa de campeonatos de todas as confederações.

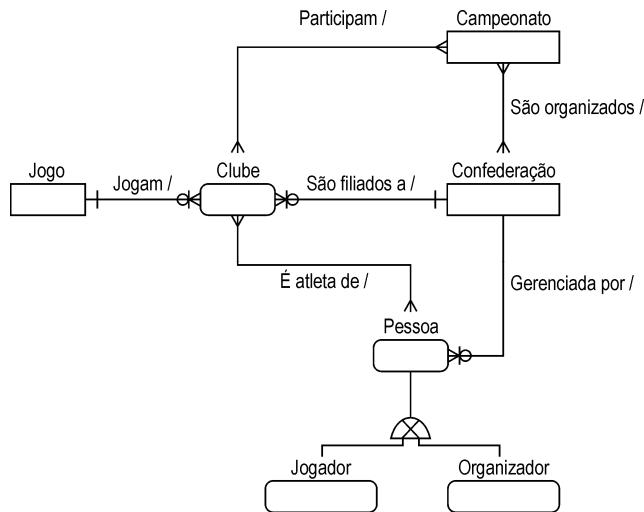
Nesse negócio as pessoas são os jogadores ou os organizadores dos campeonatos. Um campeonato tem muitos jogos.

Um campeonato pode ser organizado por mais de uma confederação. Elas podem realizá-lo em conjunto.

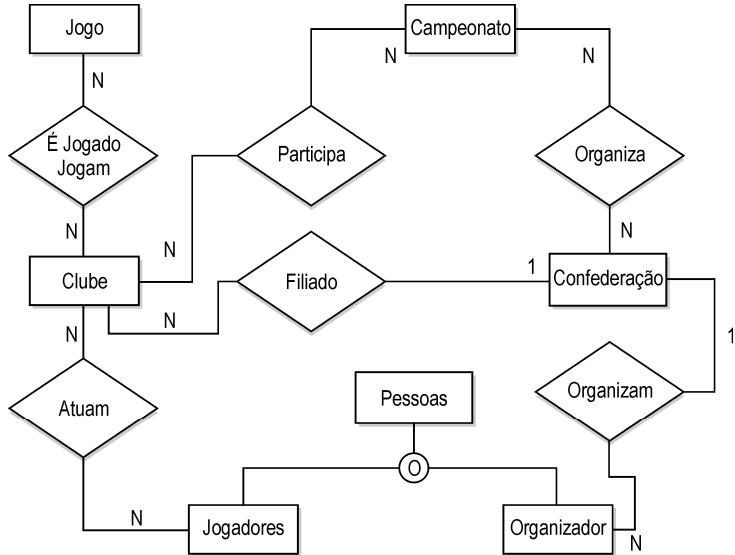
Desculpe se isso parece com o futebol brasileiro, mas não era este o objetivo.

O mesmo modelo visto sob a perspectiva da notação de Chen ficaria como na figura apresentada em seguida.

Interpretando o modelo, é importante que façamos a leitura dele.



Utilizamos nesta solução os conceitos de generalização e especialização, inserindo uma entidade pessoa que engloba tanto jogadores como organizadores, uma vez que permitimos assim que alguns dos jogadores também sejam organizadores de uma confederação.

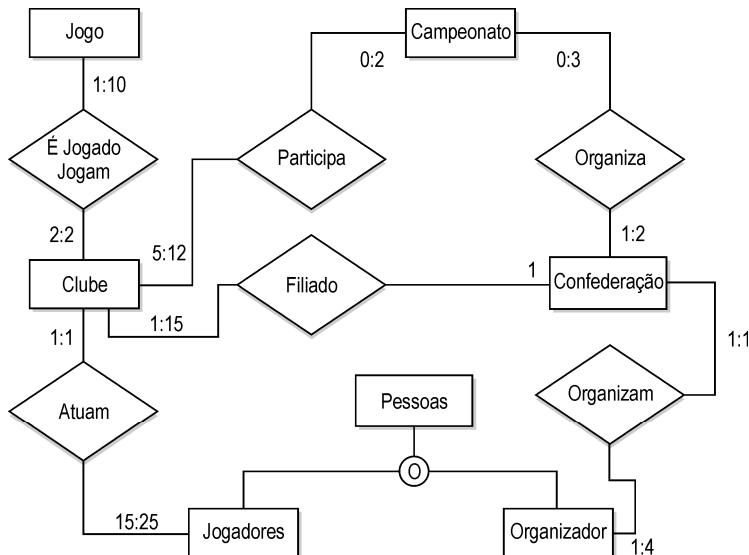


- ▶ Um jogador atua em muitos clubes.
- ▶ Um clube tem muitos jogadores.
- ▶ Um organizador organiza uma confederação somente.
- ▶ Uma confederação é organizada por muitos organizadores.
- ▶ Uma confederação organiza muitos campeonatos.
- ▶ Um campeonato é organizado por muitas confederações.
- ▶ Um clube é filiado a uma e somente uma confederação.
- ▶ Uma confederação tem como filiados muitos clubes.
- ▶ Um clube joga muitos jogos.
- ▶ Um jogo é jogado por muitos clubes (máximo 2).
- ▶ Um clube participa de muitos campeonatos.
- ▶ De um campeonato participam muitos clubes.

Uma crítica que o modelo ER sofre é que restrições de número máximo nos relacionamentos não são expressas.

Existe uma notação que utiliza a expressão dos valores máximos e mínimos em cada lado do relacionamento, deixando assim mais expressivas essas restrições. Lamentavelmente as ferramentas CASE mais conhecidas e utilizadas no mercado não implementam essas possibilidades.

Exemplo

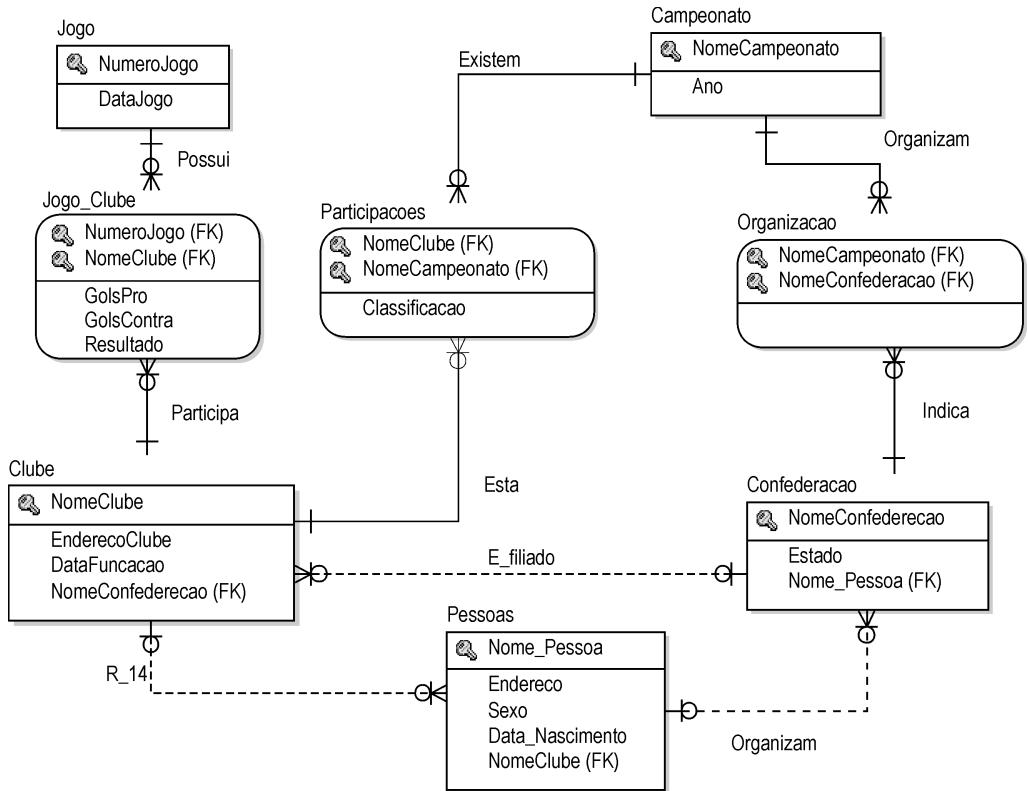


Agora ficam claras as limitações em cada relacionamento:

- Um clube pode ter no mínimo 15 e no máximo 25 jogadores inscritos.
- Um jogador só pode estar inscrito em um e no máximo em um clube.
- Uma confederação é organizada no mínimo por um organizador e no máximo por quatro organizadores.
- Um jogo é jogado por no mínimo duas e no máximo duas equipes.
- Um clube participa de nenhum ou no máximo de dois campeonatos.
- Um campeonato tem no mínimo cinco clubes e no máximo quinze clubes.
- Uma confederação organiza nenhum ou no máximo três campeonatos, que são organizados por no mínimo uma ou no máximo duas confederações.

Vamos então ao modelo lógico desta solução com a visualização dos atributos e das chaves primárias e estrangeiras existentes no modelo.

Importante agora é observar que o modelo de dados possui a identificação de chaves primárias e chaves estrangeiras que efetivam os relacionamentos em um ambiente relacional.



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 dessa entidade, denomina-se, como já vimos no ambiente relacional, chave primária.

Quando em uma entidade temos um campo, que é chave primária de outra entidade, temos então a chave estrangeira dessa entidade.

Essa ligação realiza-se por comparação do valor da chave estrangeira de uma entidade com o valor da chave primária de outra entidade.

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

Por exemplo:

- ▶ **Nome do clube em clube = Nome do clube em pessoa**

Esta é uma expressão de efetivação lógica de um relacionamento.

Se desejarmos saber o nome das pessoas que jogam em um clube específico, basta informarmos o valor do nome desse clube para que sejam selecionadas todas as ocorrências de pessoas cujo campo NomeClube seja igual ao valor informado.

Este é um processo de seleção, ou melhor, uma operação de projeção e seleção relacional sobre o relacionamento entre a entidade clube e a entidade pessoa.

$\pi \text{Nome_pessoa} (\text{CLUBE} \bullet \text{Clube.NomeClube} = \text{Pessoa.NomeClube} \text{PESSOA})$

Veja só, já estamos navegando em um modelo de dados.

Mas vamos aprender isso especificamente em álgebra relacional.

Vamos observar agora as tabelas seguintes, que simulam o conteúdo das duas entidades.

Entidade: Departamento

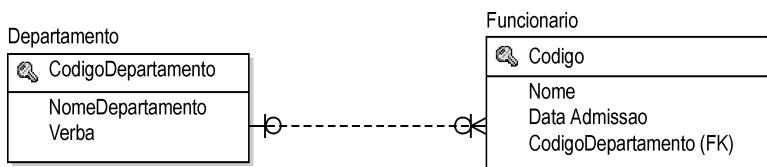
Código Departamento	Nome Departamento	Verba
01	Contabilidade	500,00
10	Vendas	1.000,00
21	Faturamento	250,00

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

Como estas tabelas que representam as duas entidades possuem chaves primárias e estrangeiras, conforme o modelo seguinte, podemos responder às questões:

Quais são os funcionários do departamento de vendas?



Primeiro devemos ver que código do departamento de vendas é igual a 10.

Em segundo lugar é preciso saber quais funcionários que têm código de departamento igual a 10.

Resposta= "Carlos, Eduardo e Luís."

Logo, a existência no modelo lógico de chaves primárias e chaves estrangeiras possibilita a recuperação de informações e efetiva logicamente os relacionamentos.

Gostaríamos de salientar que a preocupação tem sido sempre nos orientarmos para recuperar informações combinadas, não dando ênfase neste momento à inserção, manutenção ou deleção de dados, aspectos que vamos comentar adiante no livro, mas são elementos que auxiliam na descoberta do contexto e sua consolidação.

Para um bom trabalho de modelagem devemos esquecer essas 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 desse problema com vistas ao seu domínio e sua solução.

Para que se solidifiquem os conceitos de uma técnica, não é suficiente apenas a apresentação de um exemplo de situação e sua aplicabilidade; pelo contrário, a massificação de casos analisados é que nos dá a possibilidade de ter segurança no 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.

Quando os Fatos Podem Confundir

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 esse dado se refere é de suma importância para o resultado correto do modelo de dados.

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

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

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 ele vai ficar alocado. O modelo ER que representa este fato é o apresentado na figura seguinte.



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. Esses dados são inerentes ao fato alocação, logo são campos, dados do relacionamento alocado.

Não, 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, esses 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, necessitamos do atributo Código_do_Projeto em sua estrutura.

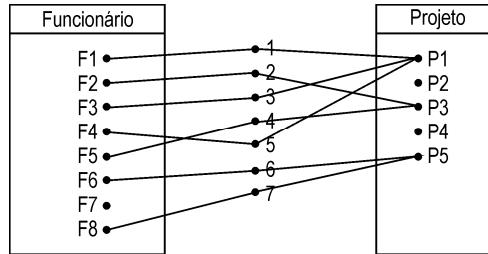
Temos assim a seguinte estrutura de dados:

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

Utilizamos o padrão FK (*foreign key*) para indicar uma chave estrangeira.

Se a realidade colocada em análise fosse a de que um funcionário estivesse alocado a muitos projetos, ela seria uma informação do relacionamento entre funcionário e projeto, já que para cada associação do funcionário com um projeto teríamos esses dados para caracterizá-la. Veremos mais adiante um estudo desse tipo de relacionamento.

A figura seguinte exibe 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 F7 não está alocado a nenhum projeto, assim como os projetos P4 e P5 não possuem nenhum funcionário alocado.

Queremos dizer com isso 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 ela 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 obrigatoriamente condicionais no mínimo para um dos lados do relacionamento.

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

Uma característica para efetivação lógica de um relacionamento um-para-muitos é o fato de ele necessitar da existência de chave estrangeira em uma das entidades.

Tomamos 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.

No exemplo, a ligação da entidade funcionário com a entidade projeto será possível de efetuar pela inserção do atributo *Codigo_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 desse atributo será NULO, isto é, uma informação desconhecida, inexistente.

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

É importante destacar que existem ocorrências que justificam a cardinalidade muitos-para-um, independentemente de existirem ocorrências com cardinalidade um-para-um no conjunto de informações em análise.

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 menos um caso de um projeto ter seu valor de Código_do_Projeto referenciado em mais de uma ocorrência de funcionário, o que 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.

Na entidade projeto existe uma ocorrência (projeto p-32) que não é referenciada por nenhuma ocorrência da tabela da entidade funcionário, assim como a entidade funcionário possui uma ocorrência que tem valor nulo para os atributos relativos ao relacionamento com a entidade projeto: Código_do_Projeto, Data_de_início e Tempo_de_Alocação são a ocorrências do funcionário Carlos Estevão.

Pelo apresentado nos exemplos, é preciso que tenhamos uma massa crítica de dados para simularmos as suas ocorrências 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 modelo de dados, qual alteração sofreria. Teríamos um relacionamento muitos-para-muitos, conforme o diagrama.



Como estudamos na definição de relacionamento com cardinalidade muitos-para-muitos, esse tipo de relacionamento tem a característica de possuir campos para representá-lo, informações que podem ser inerentes ao fato, ao evento, que é o relacionamento.

Para a efetivação lógica desse tipo de relacionamento, necessitamos de no mínimo dois atributos, que são as chaves primárias das entidades participantes do relacionamento.

Neste caso em estudo, `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, esses dados passam a ser únicos por cada ocorrência do relacionamento, passando a ser múltiplos no contexto do modelo.

Como o número de ocorrências do relacionamento é indeterminado, não podemos mais mantê-los como atributos de funcionário, pois não saberíamos quantas ocorrências colocar desses atributos em funcionário, sendo necessário o desdobramento em múltiplos e indefinidos atributos.

Logo, o relacionamento alocado passa a ter existência lógica, ou seja, possui dados ou pode ser transformado em uma entidade associativa, como já vimos outros casos.

As estruturas de dados correspondentes ao modelo apresentado ficam delineadas da seguinte forma:

Entidades	Atributos	Relacionamento ou Entidade Associativa	Atributos
Funcionário	Matrícula_Funcionário Nome_Funcionário	Alocado	Matrícula_Funcionário Código_Projeto Data_Início_no_Projeto Tempo_de_Alocação
Projeto	Código_Projeto Nome_Projeto		

Como se efetiva logicamente este relacionamento?

O relacionamento efetiva-se por meio 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 de Código_do_Projeto na entidade projeto, conjuntamente.

Quando isso 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 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 pela 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 ocorrências relacionando-se com cardinalidade um-para-um não invalidam 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.

Observe que interpretamos, ou melhor, realizamos a leitura pura e simples da tabela que representa esse relacionamento, não considerando ainda neste instante as ocorrências das duas entidades que não figuram no relacionamento.

Essas ocorrências são irrelevantes para a interpretação do relacionamento.

AGREGAÇÃO: UMA EXTENSÃO ESPECIAL

O modelo Entidade-Relacionamento, proposto por Peter Chen na década de 1970, sofreu posteriormente algumas adaptações e extensões, que visaram torná-lo mais completo e abrangente. Uma dessas extensões foi o conceito de agregação, introduzido para viabilizar a modelagem de algumas situações típicas envolvendo relacionamentos de cardinalidade N:N.

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

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

O termo agregação tem sido utilizado pelas técnicas de modelagem de sistemas nos mais variados conceitos, porém o conceito lançado em modelagem de dados refere-se à visão de um relacionamento como um bloco, como alguma coisa que se relaciona com outra.

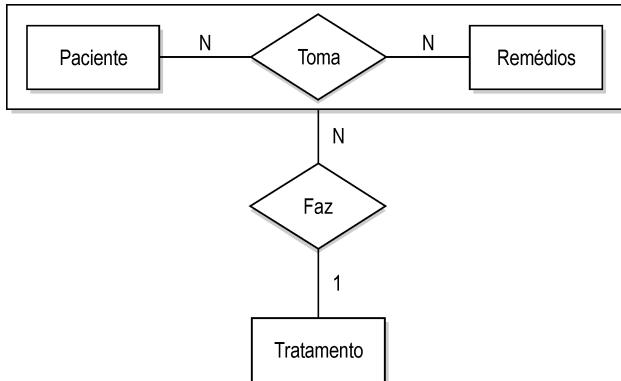


Chamamos também a atenção para o fato da existência de dependência entre os fatos, ou seja, um fato somente acontece após a existência do primeiro fato.

Em muitos trabalhos de consultoria observamos que o analista tem medo ou insegurança quanto à utilização de agregações em seus modelos de dados, seja porque existe a tendência de realizar diretamente o modelo físico ou o lógico, com a visualização da entidade associativa, ou porque cometem erros ao utilizar o conceito.

Quando um paciente toma remédios, isso configura um tratamento.

Tratamento existe sem que paciente tome remédio? Sim, pode ter existência própria.



Tratamento pode ser, por exemplo:

- ▶ Tratamento de diabetes
- ▶ Tratamento de cólica intestinal
- ▶ Tratamento de pressão alta

E assim por diante.

Vamos imaginar então que temos vários tipos de tratamento.

Poderíamos afirmar que quando um paciente toma remédios, pode ter associado um tratamento, que não impede de existir um tratamento que não esteja associado ao fato de o paciente tomar remédios. Lembre-se de que nem todas as ocorrências de uma entidade devem estar associadas.

O resultado de modelo apresentado é uma aplicação certa de agregação, com uma dose maior de interpretação da realidade.

Vamos exemplificar com outras situações.

Um determinado órgão público de planejamento possui alguns escritórios regionais estrategicamente distribuídos pelo interior de um certo estado, de maneira a facilitar o suporte aos inúmeros projetos desenvolvidos por seus funcionários.

Tipicamente os funcionários são alocados a projetos (muitas vezes simultâneos) que atingem várias regiões e tornam-se relativamente comuns situações em que um funcionário ou equipe desempenha algumas tarefas numa determinada região contando com os recursos locais do escritório mais próximo.

Para isso, basta requisitar com antecedência os equipamentos necessários ao projeto. Como existem vários projetos em andamento ao mesmo tempo, com frequência ocorre que um dado recurso seja alocado por mais de um funcionário para projetos diferentes.

Suponha agora que existem dois funcionários A e B, alocados a projetos distintos. A está alocado aos projetos 1 (Censo escolar) e 2 (Censo industrial), enquanto o funcionário B está alocado aos projetos 3 (Censo agropecuário estadual) e 2.

Para o projeto 1 o funcionário A alocou um veículo e um notebook disponíveis no escritório de uma região.

O funcionário B alocou o mesmo veículo para o projeto 3. Considere que, em princípio, não devemos ter problemas ligados ao compartilhamento do automóvel pelos dois funcionários, visto que o projeto 3, digamos, depende da conclusão do projeto 1.

Observe que são válidas, portanto, as seguintes afirmações:

- ▶ Um funcionário pode estar alocado a mais de um projeto.
- ▶ Um projeto pode ser desenvolvido por mais de um funcionário.
- ▶ Um funcionário pode requisitar um recurso para utilização num projeto.
- ▶ Um recurso pode ser utilizado por mais de um funcionário, geralmente em projetos diferentes.



Para associarmos um recurso a um funcionário, devemos levar em conta que é importante que esteja disponível a informação referente também ao projeto para o qual o recurso será utilizado. Desta forma, não devemos relacionar recurso nem unicamente a funcionário nem unicamente a projeto, mas a ambos. Em termos mais práticos, seria o caso de associarmos a entidade recurso ao relacionamento alocado que possui atributos, pois é muitos-para-muitos.

Entretanto, o modelo Entidade-Relacionamento não permite esse tipo de ligação direta com um relacionamento com atributos, pois conceitualmente não existe relacionamento entre relacionamentos.

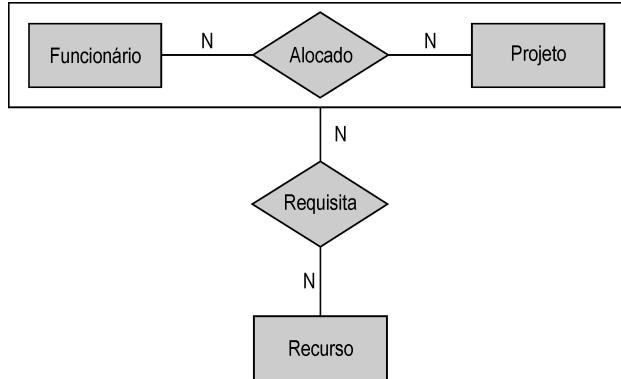
Aqui então entra um parâmetro novo em nossa análise de uma realidade.

Se observarmos o mundo real, a utilização de um recurso somente acontece QUANDO existe o fato de um funcionário estar alocado a um projeto, ou seja, um par do relacionamento descrito.

Este QUANDO coloca condicionalidade à existência da associação a entidade recurso.

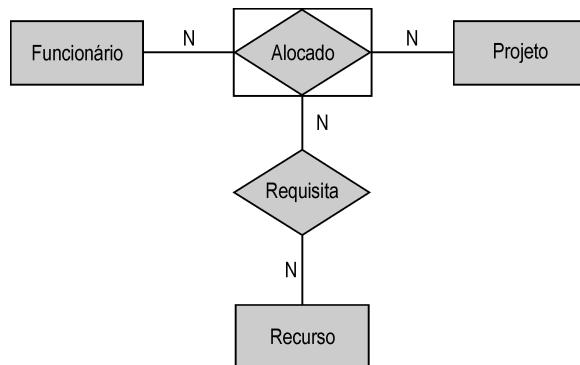
Interpretando mais abrangemente:

- ▶ Quando e somente quando funcionário alocado a projeto, requisita um recurso.
- ▶ Então funcionário alocado a projeto é um bloco do modelo, que neste momento é interpretado como uma coisa única que se relaciona com a entidade recurso.



Esta é a representação conceitual de uma agregação.

Algumas literaturas exibem uma representação que é uma mistura desta com entidades associativas.

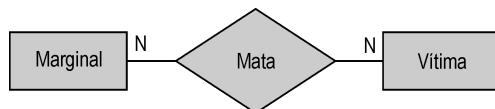


Veremos outra situação.

Vamos iniciar o estudo com um caso policial, como se estivéssemos lendo um jornal diário atual, acontecido no Rio de Janeiro, a que tanto assistimos nos telejornais. Uma típica (credo!) chacina (arhg!!, quando isso vai parar?).

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

Nesta situação, podemos afirmar que um criminoso assassinou várias vítimas e que uma vítima foi assassinada por vários criminosos, já que não podemos determinar com exatidão 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 imaginou isso sendo explicado em um telejornal?



O diagrama da figura apresenta um impasse: temos a entidade marginal e a entidade vítima relacionadas com cardinalidade muitos-para-muitos.

Aparece então em cena uma terceira entidade, denominada de arma,arma do crime, que contém as armas apreendidas.

Como vamos ligar as ocorrências de arma, ao fato, ao evento representado pelo relacionamento mata, este um relacionamento com campos?

As entidades arma têm suas ocorrências relacionadas a marginal, ou está relacionada à vítima?

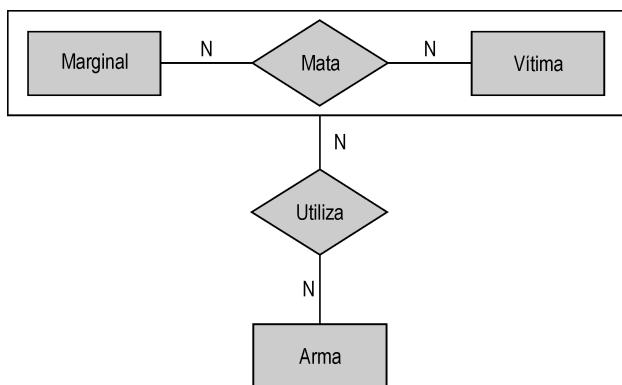
Se ligarmos arma a marginal, não podemos afirmar que ela foi utilizada no assassinato de uma vítima. Se ligarmos à vítima, mais difícil torna-se estabelecermos uma relação com o marginal.

Como resolver esta questão?

É relativamente simples a solução. Basta nos determos em retratar a realidade da mesma forma como ela é expressa em linguagem natural, ou seja:

- ▶ Um marginal mata vítimas. Quando ele mata vítimas, utiliza arma.

Desta forma temos o diagrama Entidade-Relacionamento em seguida:



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

Na realidade são necessários dois relacionamentos para retratar um fato por completo.

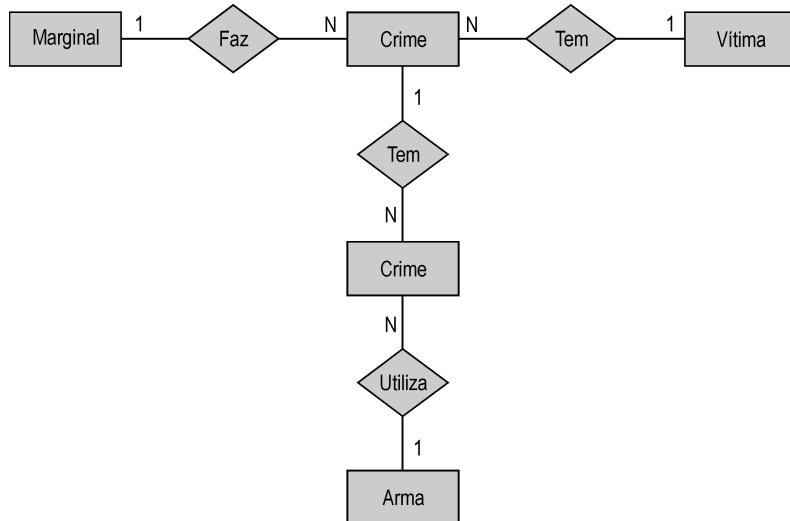
O que desejamos é relacionar uma ocorrência de arma com uma ocorrência do fato, do relacionamento mata.

O relacionamento mata, como colocamos na situação, possui campos. Vamos então relacionar uma ocorrência desse relacionamento com uma ou várias ocorrências da entidade arma, criando um relacionamento entre mata e arma, que denominamos de utiliza.

Como já explicamos neste livro, podemos resolver relacionamentos muitos-para-muitos com a utilização de entidades associativas.

O diagrama de Entidade-Relacionamento com a utilização de entidades associativas, neste caso, apresenta duas destas entidades, uma para substituir o relacionamento muitos-para-muitos mata, e outra para substituir o relacionamento também muitos-para-muitos utiliza.

Tanto no modelo anterior com a utilização de relacionamentos muitos-para-muitos como no seguinte existem atributos nesses relacionamentos ou nas entidades associativas.



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

Entidade	Atributos	Relacionamento
MARGINAL	Registro do meliante Nome Oficial Codinome Idade	Com Vítima através de Mata 1:N (Parcial) (Nem todo Meliante assassinou uma Vítima)
VÍTIMA	Núm. documento da vítima Nome do infeliz Sexo Idade	Com Meliante através de Mata 1:N (Total) (Toda Vítima foi assassinada por um Meliante)
ARMA	Núm. apreensão da arma Marca da arma Tipo da arma Número de série	Com Assassina através de Utiliza 1:N (Parcial) (Nem toda arma apreendida foi usada num assassinato)

Agora vamos observar as tabelas seguintes que apresentam a simulação das tabelas representativas das entidades e dos relacionamentos envolvidos nesta novela policial.

Marginal

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

Vitima

Documento	Nome	Sexo	Idade
1111111111	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 em seguida mostram apenas uma parte dos valores de dados dos relacionamentos possíveis para as ocorrências das entidades marginal e vítima, considerando que os casos não estão relacionados ou não foram até esta edição registrados.

Mata

Data	Vítima	Meliante
11/02/92	1111111111	121212
14/03/92	243387569	121212
03/04/92	806578913	334455

Usa

Vítima	Meliante	Arma
1111111111	121212	191
1111111111	121212	192

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

Ou seja, nesse crime o infeliz matou uma única vítima utilizando-se de duas armas simultaneamente ou juntas.

Um detalhe muito importante é que a agregação não é representada pelas ferramentas CASE mais conhecidas, o que leva, quando se executam trabalhos de modelagem, a não ser

utilizada, porém isso dificulta a expressão e o entendimento do minimundo , motivo pelo qual recomendamos sempre a execução manual de um modelo conceitual para depois iniciarmos os trabalhos com a ferramenta CASE.

O Hotel

Vamos estudar outros casos e explorar mais situações de agregação.

Destacamos aqui que é necessário que os analistas de sistemas modelem sempre em um primeiro nível conceitual, sendo necessário para isso utilizar a capacidade máxima de abstração. É pela visão perceptiva do ambiente que está sendo modelado que conseguimos atingir modelos mais consistentes, que retratem o mundo real.

Considere um modelo de dados como o da figura, que retrata o ambiente de um hotel.



A leitura do modelo diz que um cliente do hotel pode ocupar muitos quartos e um quarto pode ser ocupado por muitos clientes.

Às vezes, quando apresentamos este modelo, os alunos questionam esta espacialidade do modelo de dados.

As afirmativas anteriores estão corretas porque o relacionamento entre cliente e quarto é uma relação espacial, não temporal.

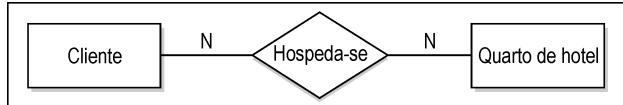
Um cliente em relação ao objeto quarto tem um relacionamento de conectividade muitos-para-muitos porque ele pode vir muitas vezes ao hotel e ficar sempre em quartos diferentes. O quarto de hotel pode ser ocupado por vários clientes diferentes em momentos diferentes.

A noção de tempo está dentro do relacionamento. Cada ocorrência do relacionamento possui informações sobre uma estadia, registro de datas, horas etc.

Seguindo a leitura deste pequeno modelo, coloca-se a seguinte questão:

O cliente do hotel utiliza vários serviços disponíveis, alguns com consumo de produtos como frigobar, restaurante, e outros meramente serviços como diária, por exemplo, que é um dos serviços do hotel.

Como relacionar estas coisas chamadas serviços e produtos com o fato de um cliente hospedar-se no hotel (Cliente hospeda-se em quarto)?



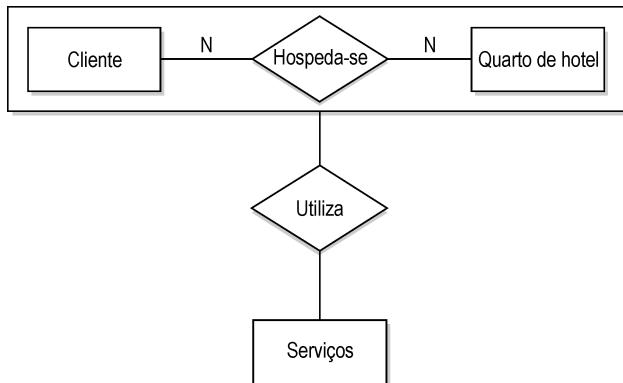
Em primeiro lugar devemos entender que tudo o que acontece a seguir é decorrente, dependente de cliente hospedar-se em quarto de hotel, logo todo e qualquer relacionamento será com esta visão agregada de parte do modelo.

Como serviços é uma coisa sobre a qual interessa manter informações, tem significado próprio, caracteriza uma entidade a mais em nosso modelo. Assim sendo, temos de estudar e entender como isso é no mundo real.

No mundo real os serviços são utilizados quando cliente hospeda-se em quarto de hotel, logo a entidade serviços se relaciona com o bloco superior.

Interpretamos o bloco cliente hospeda-se quarto hotel e relacionamos com a entidade serviços.

Agora vamos buscar entender a conectividade que existe entre as ocorrências de serviços e o bloco cliente no hotel por assim dizer.



Antes, porém, vamos visualizar como seriam os dados numa estruturação de tabelas destas entidades até o momento.

Cliente

CodCliente	NomCliente	RuaCliente	CidadeCliente
1	Luis Sampaio	Rua A	Rio de Janeiro
2	Carlos Pereira	Rua B	Niterói
3	Jose Alves	Rua C	Rio de Janeiro
4	Luis Paulo Souza	Rua B	Niterói

Quarto

NumQuarto	Andar	Tipo Quarto
101	1	Single
102	1	Doble
301	3	Single Luxo
401	4	Single Luxo
1402	14	Suite Presidencial

O relacionamento hospeda-se possui atributos, uma vez que é um muitos-para-muitos.

Hospeda-se

CodCliente	NumQuarto	DataEntrada	DataSaida
1	101	21/12/2003	22/12/2003
2	102	15/2/2004	17/2/2004
3	301	10/1/2004	18/2/2004
4	101	11/1/2004	30/1/2004
2	401	15/2/2004	16/2/2004
3	1402	25/12/2003	3/1/2004
1	102	15/3/2004	1/4/2004

Acompanhe a tabela de serviços para que fique claro o que ela representa.

Serviços

CodServiço	Nome Serviço
1	frigobar
2	Restaurante
5	Serviço de Quarto
7	Sauna
10	Diária

Um cliente hospeda-se em quarto utiliza quantos serviços? Muitos, é claro.

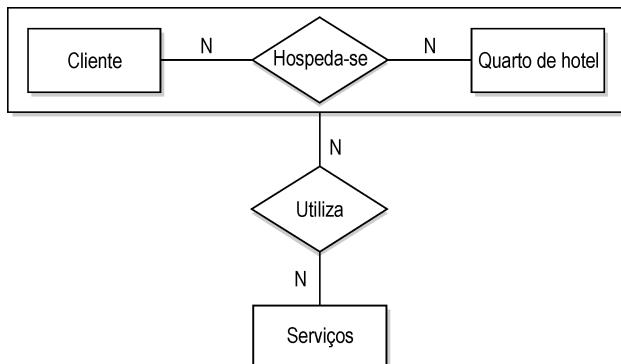
Um serviço é utilizado por quantos clientes que se hospedam em quarto de hotel?

Também a resposta é muitos, logo temos um relacionamento muitos-para-muitos entre serviços e clientes que se hospedam em quarto de hotel.

Este é mais um relacionamento que deve ter atributos, pois devem estar registradas nesse relacionamento as datas de utilização, hora, valor do serviço.

Utiliza

CodCliente	NumQuarto	CodServiço	Data	Hora	Valor
1	101	1	21/12/2003	15:30	50,00
2	102	5	15/2/2004	18:00	75,00
3	301	10	10/1/2004	12:00	150,00
4	101	1	11/1/2004	8:00	8,00
2	102	10	15/2/2004	12:00	220,00
3	1402	2	25/12/2003	21:45	150,00
1	102	10	15/3/2004	12:00	90,00
2	401	5	16/2/2004	22:00	18,00
3	1402	2	31/12/2003	19:00	1.500,00



Logo você vai se perguntar: mas os produtos não são atributos do relacionamento utiliza?

Antes de respondermos, vamos analisar um pouco mais a realidade.

Quando produtos são consumidos?

Os produtos são consumidos normalmente quando se utiliza algum serviço do hotel. No restaurante, no bar, no frigobar e principalmente o fato de estar no hotel, pois a diárida é nada mais que o valor cobrado por cada dia de estada e também é um serviço, o serviço de hospedagem propriamente dito.

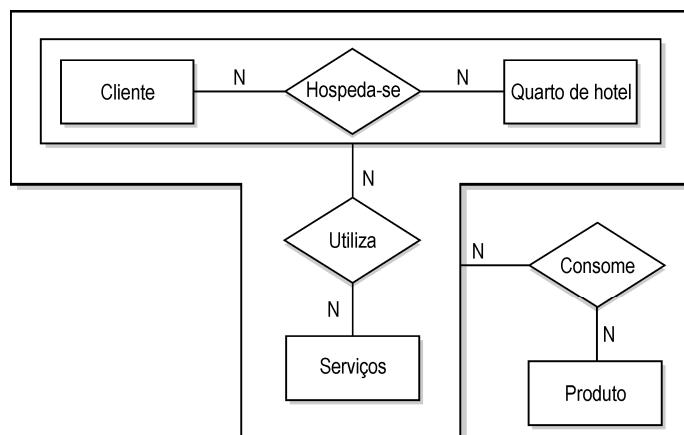


Então, produto está relacionado com o relacionamento utilizा?

Claro, pois nada impede ou limita o número de agregações em um modelo de dados. Elas podem ser superpostas sem nenhum problema, pois estamos nada mais nada menos que retratando uma realidade.

Leia o modelo solução apresentado, interprete-o e procure visualizar o mundo real para validá-lo. Lembre-se de que deve ser lido da mesma forma que foi construído: em camadas superpostas.

Claro que os relacionamentos são opcionais neste caso, e como apresentamos no capítulo, não é possível explicitar no modelo na notação de Chen as regras da optionalidade quanto à sua temporalidade.



O utiliza é um relacionamento de muitos-para-muitos, pois uma ocorrência de serviço pode ser utilizada por muitas ocorrências de cliente hospeda-se em quarto.

Agregação é o bloco cliente hospeda-se em quarto, um fato multirrelacional nos dois sentidos de leitura.

O fato dependente é utiliza, pois ele somente existe, ou melhor, acontece, quando o primeiro fato já existe.

Após agrarmos cliente hospeda-se em quarto e relacionarmos com serviços, vamos então agragar este fato (utiliza) com o anterior (hospeda-se) e tratar este fato composto como um único bloco, uma única coisa, e relacioná-lo com o consumo de produtos.

Lendo o modelo final, temos:

Cliente hospeda-se em quarto. Quando se hospeda utiliza serviços. Quando utiliza serviços consome produtos.

Observe que os três relacionamentos muitos-para-muitos possuem dados.

Hospeda-se pode ter os dados de data de chegada do hóspede, data de saída, além de informações padrão Embratur etc.

Utiliza tem informações sobre quando foi utilizado o serviço, conforme apresentamos na tabela.

Consome indica as quantidades de produtos consumidas pelo serviço utilizado.

Vejamos então a tabela representativa de produto e de consumo.

Produto

CodProduto	Descrição	Valor Unitario
1	Coca-Cola	R\$ 2,00
5	Cerveja Long Neck	R\$ 3,50
52	Mate	R\$ 1,50
84	Café	R\$ 1,00
21	Suco	R\$ 3,00
16	Água Min.	R\$ 1,80
85	Steack T-Bone	R\$ 25,00
12	Camarão Empanado	R\$ 32,00
9	Cheesburger	R\$ 12,00
1	Castanhas de Caju	R\$ 15,00
7	Vinho Tinto Importado	R\$ 80,00
25	Sobremesas	R\$ 10,00
99	Ceia Fim de Ano	R\$ 900,00
59	Champagne Importado	R\$ 200,00

Consumo

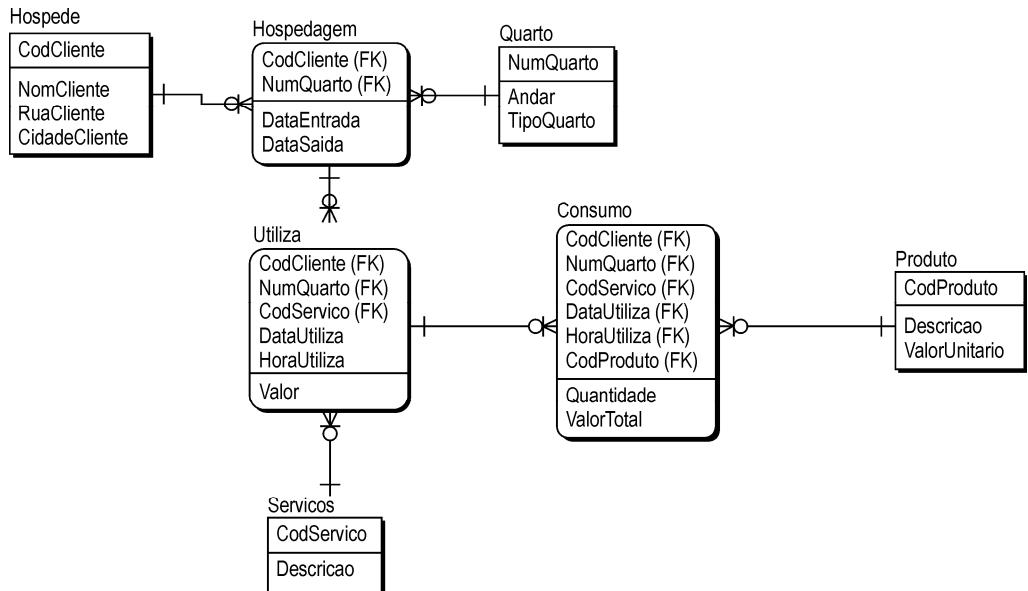
CodCliente	NumQuarto	CodServiço	Data	Hora	CodProduto	Quantidade	Valor Total
1	101	1	21/12/2003	15:30	52	1	1,50
1	101	1	21/12/2003	15:30	5	5	17,50
1	101	1	21/12/2003	15:30	1	2	30,00
4	101	1	11/1/2004	8:00	5	2	7,00
4	101	1	11/1/2004	8:00	84	1	1,00
2	102	5	15/2/2004	18:00	85	3	75,00
3	1402	2	25/12/2003	21:45	7	1	80,00
3	1402	2	25/12/2003	21:45	85	2	50,00
3	1402	2	25/12/2003	21:45	25	2	20,00

CodCliente	NumQuarto	CodServiço	Data	Hora	CodProduto	Quantidade	Valor Total
2	102	5	16/2/2004	22:00	5	4	14,00
2	102	5	16/2/2004	22:00	21	1	3,00
2	102	5	16/2/2004	22:00	84	1	1,00
3	1402	2	31/12/2003	19:00	7	6	480,00
3	1402	2	31/12/2003	19:00	99	1	900,00
3	1402	2	31/12/2003	19:00	25	2	20,00
3	1402	2	31/12/2003	19:00	59	1	200,00

Para entendermos agregação, voltamos a insistir que é necessário modelar o ambiente sem pensar em procedimentos e não pensar no banco de dados físico.

Para fins de esclarecimento apresentamos o modelo de tabelas para a implantação desse modelo do hotel, mas somente para você conseguir visualizar os dados e sua organização em um modelo de dados conceitual, pois como sempre destacamos, não devemos modelar diretamente para o ambiente físico, pois retira a capacidade de visualizar abstrações e modelar corretamente o mundo real.

Observe que nas tabelas referentes ao modelo os relacionamentos aparecem como tabelas também, pois se possuem dados, logo são tabelas relacionais. Como são relacionamentos muitos-para-muitos, as chaves estrangeiras também estão neles, pois são as efetivações lógicas dos relacionamentos.



Com colunas comuns podemos realizar junções de tabelas para a recuperação de informações do negócio.

Explicação Adicional sobre as Chaves

Quando partimos para um modelo de dados em que inserimos chaves primárias e chaves estrangeiras, estamos realizando modelos lógicos, pois fazemos a efetivação dos relacionamentos agora voltados para regras do modelo relacional, com o uso de colunas comuns entre tabelas associadas a um fato ou por um fato.

Vamos continuar a estudar mais agregações.

Uma pergunta que já foi feita várias vezes: uma agregação pode estar relacionada à outra agregação?

Pode, pois são dois objetos que se relacionam no mundo real, logo é possível.

Vamos estudar rapidamente um caso com esse tipo de aplicação de modelo.

Clínicas Médicas

Vamos expor a situação e o modelo de solução para este caso.

Uma clínica médica que é uma empresa realiza atendimentos médicos nas mais diversas especialidades e situa-se em determinados endereços, que denominaremos de local.

Em cada endereço, uma clínica médica atende uma e somente uma especialidade médica.

As clínicas médicas têm médicos contratados, que atendem somente uma especialidade médica na clínica na qual são contratados. Uma especialidade médica pode ser exercida por vários médicos, mas um médico pode exercer somente uma especialidade em uma clínica médica.

Um médico somente pode ser contratado por uma clínica.

Uma clínica pode disponibilizar atendimento na mesma especialidade ou especialidades diferentes em locais diferentes.

Um único local e suas instalações podem ser compartilhados por duas ou mais clínicas médicas por meio de um contrato de compartilhamento.

Por exemplo, uma clínica atende na parte da manhã em um determinado local, e outra clínica atende nesse local na parte da tarde, ou então as salas do local são divididas no mesmo horário ou turno.

Sempre que atende um paciente, o médico preenche um receituário por paciente atendido, que pode conter de ZERO a N (muitos) remédios, e no mínimo uma prescrição médica ou um atestado, portanto sempre é obrigatório o preenchimento. Todos os pacientes são registrados independentemente da clínica que os atendeu.

Um mesmo paciente em datas diferentes pode ter receitas com o mesmo remédio, com médicos diferentes ou com o mesmo médico.

O código e o nome dos remédios são padrões internos do sistema, obedecem ao critério de formulação genérica, portanto podem ser supridos por qualquer laboratório farmacêutico. São controlados pela sua substância ativa e não por marca.

Um remédio só pode ser receitado uma única vez na mesma data para o mesmo paciente.

Vamos analisar este modelo tentando isolar as partes do negócio para facilitar a nossa conquista do modelo de negócio.

Como já dizia Jack, o estripador, vamos por partes.

O primeiro bloco de modelo que vamos analisar talvez seja o centro do negócio de clínicas médicas, o atendimento, a consulta de pacientes.

Um médico atende muitos pacientes.

Um paciente consulta com muitos médicos

Logo, nosso relacionamento no centro do negócio está correto como na figura seguinte.



Entretanto, dependendo da interpretação do ambiente e da capacidade de abstração do analista, também é válida a interpretação de que existe um objeto consulta neste contexto.

Desta forma devemos nos preocupar em entender como são os relacionamentos entre as entidades do contexto: médico, paciente e consulta.

- ▶ Um médico faz muitas consultas.
- ▶ Cada consulta tem um e somente um médico.
- ▶ Um paciente tem muitas consultas.
- ▶ Cada consulta tem um e somente um paciente.



Esta também é uma interpretação válida para este problema.

Bem, vamos quebrar outro bloco para o modelo.

"Uma clínica médica, que é empresa, realiza atendimentos médicos nas mais diversas especialidades e situa-se em determinados endereços, que denominaremos de local."

Então temos a entidade clínica e uma entidade local, e existe um relacionamento entre as duas entidades.



Analizando as afirmativas seguintes sobre clínicas e locais:

"Um único local e suas instalações podem ser compartilhados por duas ou mais clínicas médicas por meio de um contrato de compartilhamento."

Então um local pode ter mais de uma clínica.

Podemos concluir que uma clínica situa-se em muitos locais e em um local situam-se muitas clínicas?

Ainda não. Até o momento somente podemos concluir que um local pode ter mais de uma clínica.

"Uma clínica pode disponibilizar atendimento na mesma especialidade ou especialidades diferentes em locais diferentes."

"Uma clínica médica... situa-se em determinados endereços, que denominaremos de local."

"Porém, em cada endereço uma clínica médica atende uma e somente uma especialidade médica."

Analizando estas três assertivas, podemos afirmar que uma clínica pode situar-se em muitos locais de atendimento.

Temos agora a conectividade deste relacionamento, que é muitos-para-muitos também.



Da mesma forma podemos visualizar ou entender estas associações sem o uso de agregação, simplesmente pelo entendimento de que existe um objeto contrato que associa clínica e local.



O leitor deve estar perguntando: onde foram colocados os remédios e as especialidades?

Sim, porque estão caracterizados pelo critério e dicas como entidades.

Calma, chegaremos lá.

Existem afirmativas neste caso que estabelecem relações entre médicos e clínicas e ainda por cima destaca-se especialidade como algo relevante. Senão, vejamos:

"As clínicas **médicas** têm médicos contratados, e estes atendem somente uma especialidade médica na clínica pela qual estão contratados."

"Porém, em cada endereço atende (a clínica) uma e somente uma especialidade médica."

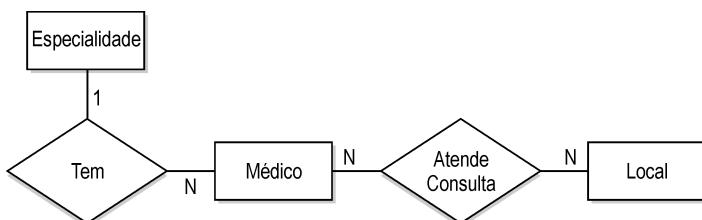
"Um médico somente pode ser contratado por uma clínica."

Logo, temos relações entre médicos, clínicas e especialidades e ainda por cima entre especialidades e clínicas.

- ▶ Todo médico tem uma e somente uma especialidade quando está associado, contratado por uma clínica, que é a especialidade em que ele vai atuar nessa clínica.
- ▶ A clínica em um local atende uma e somente uma especialidade
- ▶ Um médico só pode estar relacionado a uma clínica.

Como o médico só pode atuar em uma especialidade na clínica que o contratou, e somente pode estar vinculado a uma clínica, somente nos interessa a especialidade em que ele atua na clínica.

Vejamos a solução para a primeira afirmativa.



Um médico tem uma e somente uma especialidade, e uma especialidade tem muitos médicos.

Continuando o modelo.

Quem mais tem especialidade? A clínica? O local?

A resposta correta é que a clínica no local tem especialidade.

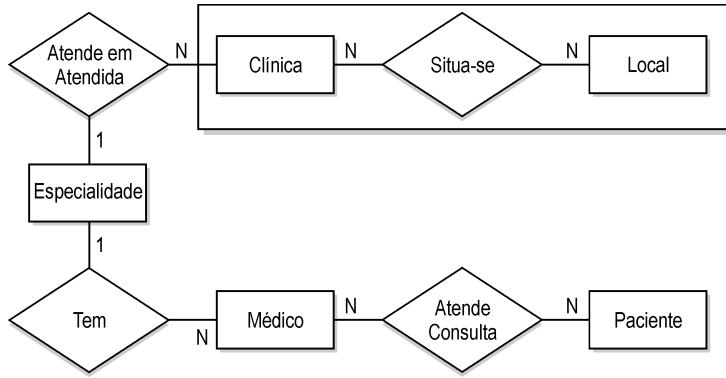
"Porém, em cada endereço atende (a clínica) uma e somente uma especialidade médica."

Logo podemos usar mais uma agregação no modelo.

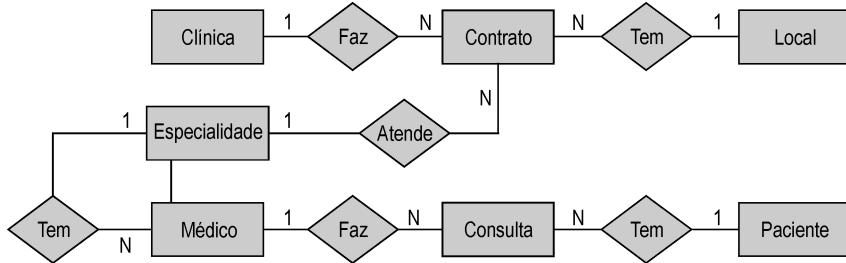
O conjunto, bloco de agregação "clínica situa-se local" relaciona-se com especialidade.

"Uma clínica situa-se local" atende em uma especialidade, e uma especialidade é atendida em muitos "uma clínica situa-se local".

O relacionamento de especialidade com o bloco é feito com a agregação apresentada.



Continuando com as alternativas de interpretação, se estamos com o modelo aberto anteriormente sem agregação, como efetivar esses relacionamentos?



O diagrama apresenta o relacionamento de especialidade com o equivalente da agregação anterior, que é a entidade associativa contrato. O relacionamento de especialidade com médico é idêntico ao modelo com agregação.

Vamos analisar e estudar agora o que fazemos com os remédios citados neste caso.

"O código e o nome dos remédios são padrões internos do sistema, obedecem ao critério de formulação genérica, portanto podem ser supridos por qualquer laboratório farmacêutico. São controlados pela sua substância ativa e não por marca."

Logo, remédio é uma entidade, pois tem significado próprio e dados que o descrevem.

Analisando outra afirmativa do problema: *"Sempre que atende um paciente o médico preenche um receituário por paciente atendido, que pode conter de zero a N (muitos) remédios, e no mínimo uma prescrição médica ou um atestado, portanto sempre é obrigatório esse preenchimento. Todos os pacientes são registrados independentemente da clínica que o atendeu."*

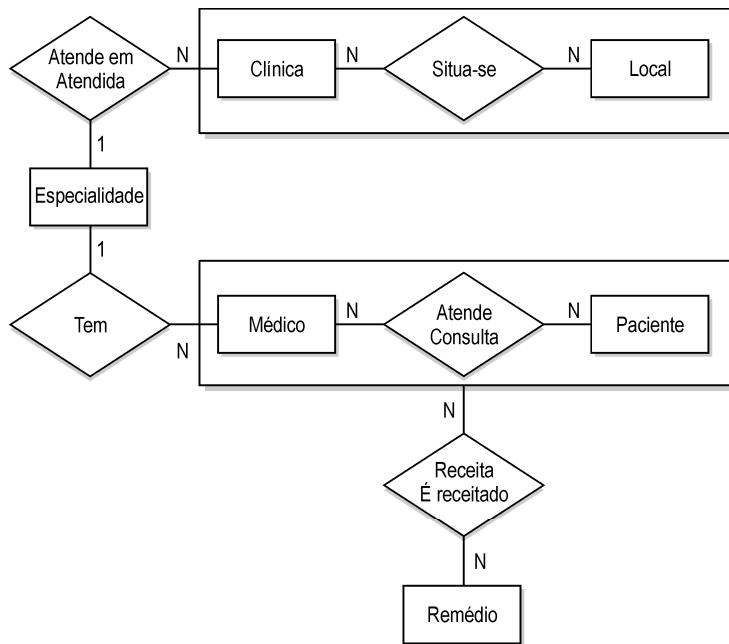
Isso denota que existe um relacionamento entre remédio e o atendimento de um médico a um cliente, a consulta propriamente dita.

Como temos um relacionamento de muitos-para-muitos entre médico e paciente, e representa o fato da consulta, podemos realizar uma agregação e relacioná-la com remédios, pois quando existe consulta (agregação) são receitados de zero a N remédios.

Destacamos no modelo apresentado em seguida as ligações entre remédios e a agregação e a ligação entre especialidade e médico.

O objetivo é destacar que um relacionamento ocorre entre uma entidade e um bloco, uma agregação, enquanto outro (especialidade tem médico) ocorre entre duas entidades diretamente.

Observe no modelo que a linha que une o relacionamento tem e a entidade médico é contínua até a caixa da entidade, enquanto a linha do relacionamento entre remédio e o fato consulta é contínua até a caixa que representa, destaca a agregação.



Agora será que nosso modelo está completo? Não representamos ainda neste modelo a associação, o relacionamento entre médico e clínica.

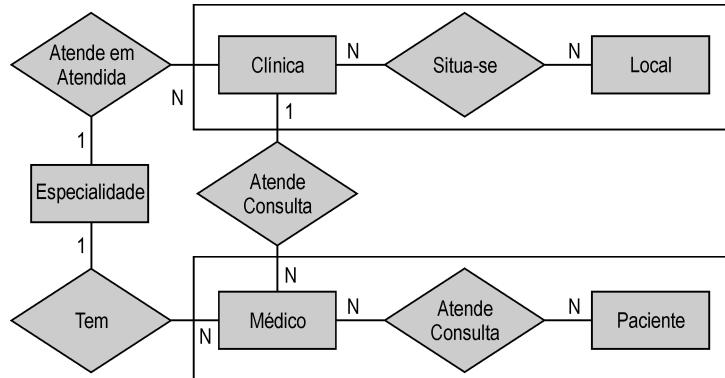
"As clínicas médicas têm médicos contratados, e estes atendem somente uma especialidade médica na clínica pela qual são contratados. Uma especialidade médica pode ser exercida por vários médicos, mas um médico pode exercer somente uma especialidade em uma clínica médica."

"Um médico somente pode ser contratado por uma clínica."

Logo, devemos ainda inserir mais um relacionamento no modelo obtido até o momento.

- ▶ Um médico atende em uma clínica.
- ▶ Uma clínica pode ter muitos médicos atendendo.

O relacionamento é entre as entidades médico e clínica.



Neste ponto destaca-se a seguinte questão:

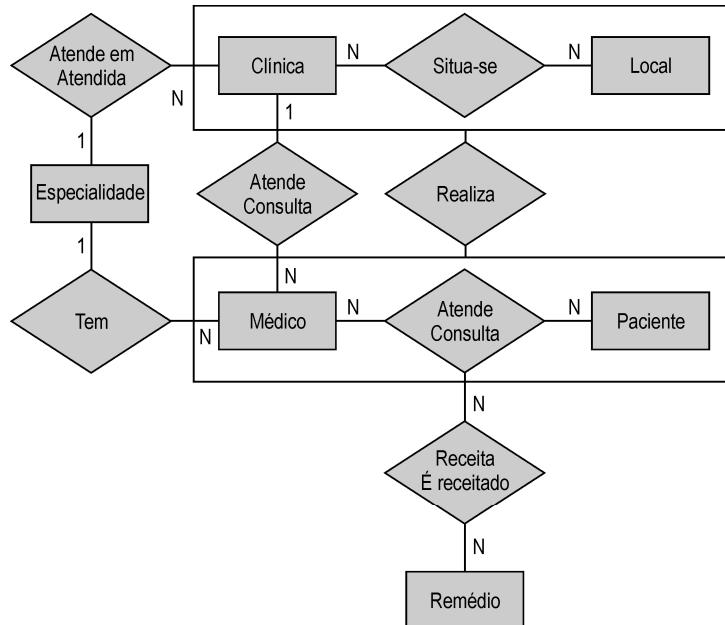
Neste modelo, como podemos descobrir onde o médico atende paciente?

Do jeito que está não existe nenhum caminho de navegação que possibilite chegar a esta resposta.

Se obtivermos os dados do paciente, do médico e da consulta, podemos identificar uma especialidade. Seguindo por especialidade, vamos obter mais de uma clínica, sem falar nos locais múltiplos que podemos encontrar para cada clínica. Se navegarmos por médico e clínica, também vamos encontrar muitos locais.

Como resolver então este problema do modelo?

A solução vem de um relacionamento entre duas agregações neste modelo.



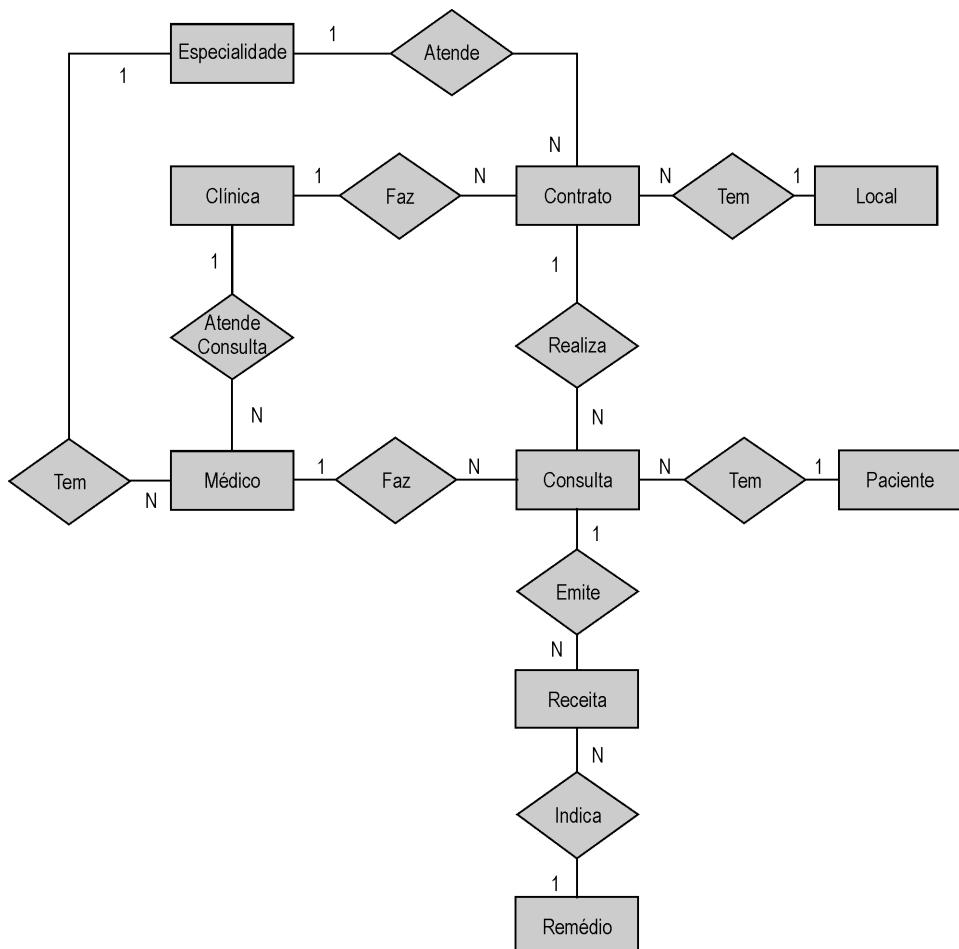
Observe a ligação do relacionamento com os dois blocos que representam a agregação.

Este mesmo modelo, no momento em que formos utilizar uma ferramenta de modelagem, como estas estão direcionadas para bancos de dados relacionais e não implementam o conceito de agregação nos seus templates de modelagem lógica, causará a necessidade de utilizar tabelas associativas no lugar de relacionamento muitos-para-muitos.

Vamos mostrar agora o mesmo modelo lógico de duas formas diferentes: sem os relacionamentos muitos-para-muitos com agregações, realizado em uma ferramenta CASE, e o modelo a partir dos exemplos de entendimento do minimundo sem agregações que vínhamos já apresentando neste caso.

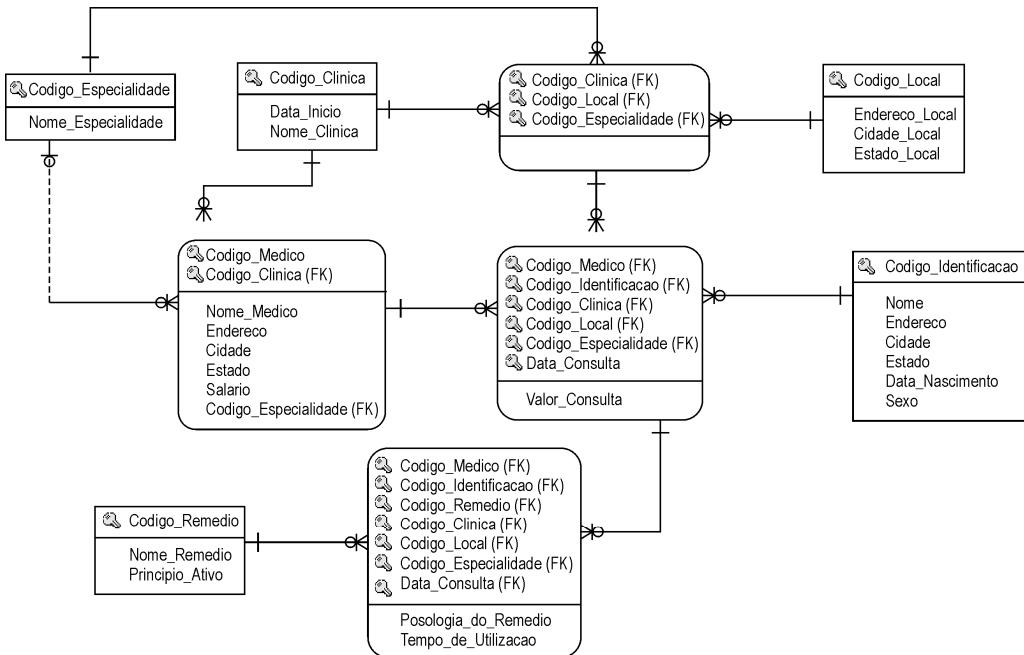
Todos os modelos apresentados satisfazem as necessidades explicitadas para o minimundo.

Modelo com notação de Chen sem relacionamentos muitos-para-muitos



O modelo seguinte é gerado em uma ferramenta CASE com a especificação dos atributos de todas as entidades.

Como detalhe não estamos apresentando os nomes das entidades para que o leitor exerceite e identifique cada uma das nossas entidades e relacionamentos.



Observe que os três relacionamentos muitos-para-muitos que se transformaram em entidades possuem um número acentuado de atributos, principalmente em suas chaves primárias, pois herdam as chaves primárias, constituindo-se em chaves estrangeiras dentro das próprias primárias, das entidades relacionadas.

O que desejamos mostrar é que a utilização de agregação é um fato natural na maioria dos modelos de dados. Sempre temos algum fato que se relaciona com uma outra entidade ou com outro fato.

Em empresas industriais é um fato muito comum, como vamos demonstrar no próximo exemplo em estudo, uma fábrica de produtos sob encomenda.

A Fábrica

Vamos analisar outra situação de agregação bem prática, em uma indústria que fabrica seus produtos sob encomenda, ou seja, não existe estoque, os pedidos de produtos são recebidos e tabulados. Existe então a emissão de muitas ordens de produção. A ordem de produção depende da quantidade de cada produto solicitado nos pedidos, executada para atender vários pedidos ou somente um.

Considerando que as ordens de produção executam a fabricação de somente um produto cada uma, podemos ter várias ordens de produção sendo executadas para atender cada pedido.



Vamos iniciar a nossa interpretação da realidade pela relação existente entre pedido e produto, muito apresentada em literaturas que falam (falam, mas não ensinam) sobre modelo de dados Entidade-Relacionamento. A figura mostra este relacionamento.



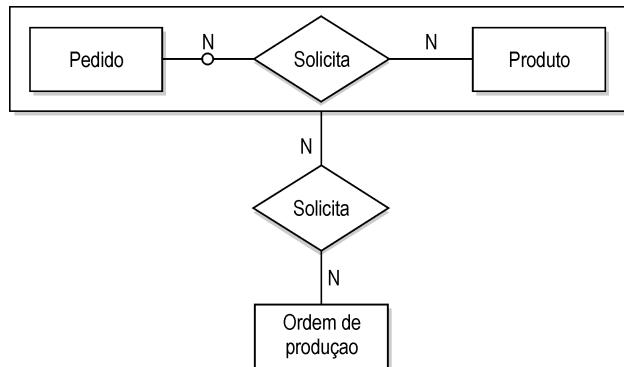
- ▶ Um pedido solicita um ou vários produtos.
- ▶ Um produto é solicitado por um, vários ou nenhum pedido.

Novamente destacamos que este relacionamento é opcional no sentido de produto para pedido, pois podemos ter ocorrências de produto que não estejam solicitadas por nenhuma ocorrência de pedido. Somente não podemos dizer quando e como isso ocorre na vida real.

E como relacionamos a ordem de produção comentada com o fato?

Se você acompanhou este livro com calma até este ponto, já sabe o resultado da questão proposta.

Ordem de produção se relaciona com a agregação formada por pedido solicita produto.



Este relacionamento é muitos-para-muitos.

Como? Vamos ler o modelo e entender a realidade.

Quando um pedido solicita produtos, então é atendido por uma ou várias ordens de produção.

Uma ordem de produção atende um ou vários pedidos que solicitam produtos.

Por que não relacionar a ordem de produção diretamente com pedido?

Porque a ordem de produção vai atender a solicitação de produtos especificados no relacionamento entre pedido e produto e não ao pedido todo.

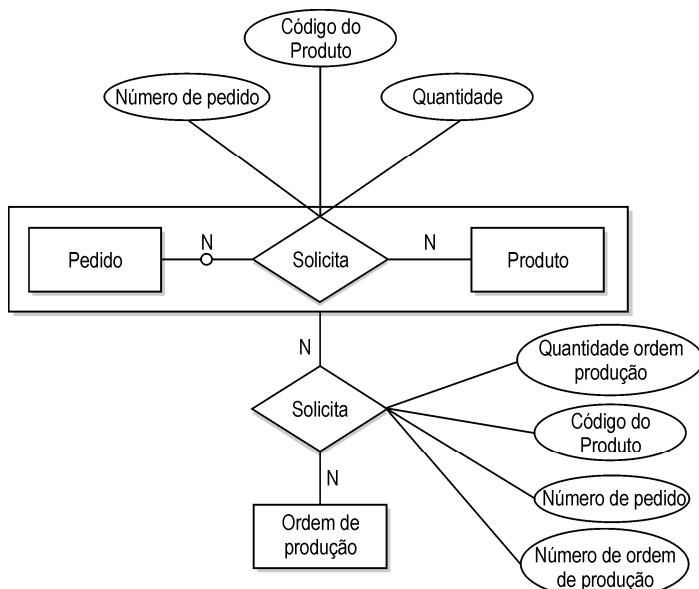
Onde está a informação sobre os produtos em cada pedido?

Está no relacionamento, que possui campos, dados.

Observe o modelo com atributos básicos nos relacionamentos e exerçite sua leitura de um modelo de dados.

Existe uma regra para todo vendedor: você só vende aquilo que compraria.

Aplicada a este caso:



Você só aprova o modelo de dados que entende.

Regras para Identificar e Utilizar Agregação

Mesmo apresentando exemplos, sabemos que no dia a dia dos analistas de sistemas a atividade de modelar dados é realizada sob intensa pressão de gerentes e usuários, logo necessitamos passar aos leitores algumas regras e dicas para identificar rapidamente as possibilidades de existência de agregações nos modelos de dados em construção.

Regra 1

Somente podem existir agregações em relacionamentos de muitos-para-muitos.

Um relacionamento muitos-para-muitos sempre representa um fato, um acontecimento entre duas coisas, duas entidades.

Ora, na vida real sabemos que quando algo acontece, podem existir coisas ou fatos que são decorrentes do acontecido.

Por exemplo, na vida real:

Acidentes de Trânsito

Acidentes ocorrem em vias de circulação, sejam estradas, ruas ou avenidas, e envolvem sempre veículos e pessoas.

Modelando isso, sabemos que acidente é um fato, e poderíamos afirmar que:

Veículos accidentam-se em locais.



Como nosso trânsito é muito maluco, ou melhor, mal-educado mesmo, em um determinado local podem ocorrer muitos acidentes com muitos veículos.

Um veículo pode sofrer muitos acidentes em vários locais.

Já temos neste momento a regra número 1 satisfeita: é um relacionamento de muitos-para-muitos entre veículos e locais.



Regra 2

Sempre que obtivermos em um modelo de dados um relacionamento muitos-para-muitos, devemos questionar, ou melhor, dirigir nossa análise para o que existe neste mundo real que seja decorrente do fato representado.

Se existir alguma coisa associada, decorrente do fato, estamos já com a definição de que algo se agrava ao relacionamento definido.

Senão, vejamos o exemplo real.

Quando veículo acidenta-se em local, o que acontece?

O que nos interessa saber?

O que perguntaríamos logo que obtivéssemos esta informação?

O sádico perguntaria:

Quantos morreram?

O técnico perguntaria: foi imprudência ou excesso de velocidade?

O policial perguntaria:

Já registraram a ocorrência?

Já chamaram socorro?

Os envolvidos perguntariam, se estiverem vivos:

Você tem seguro?



Observa-se que existiriam, na vida real, muitas questões que naturalmente nos levariam a encontrar alternativas de coisas relacionadas ao fato.

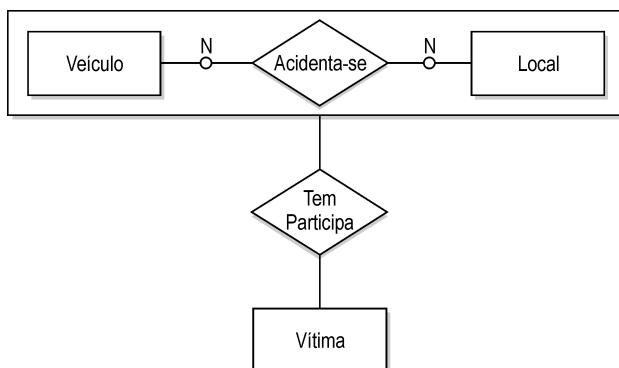
Vamos analisar somente uma: vítimas (sádico).

Quando veículo acidenta-se em local, existem vítimas.



Vítimas participam de veículo acidenta-se em local.

Com base nas afirmativas podemos desenhar um modelo de dados com agregação como solução, como na figura seguinte.



Regra 3

A última regra é determinarmos a conectividade do relacionamento entre a entidade e a agregação.

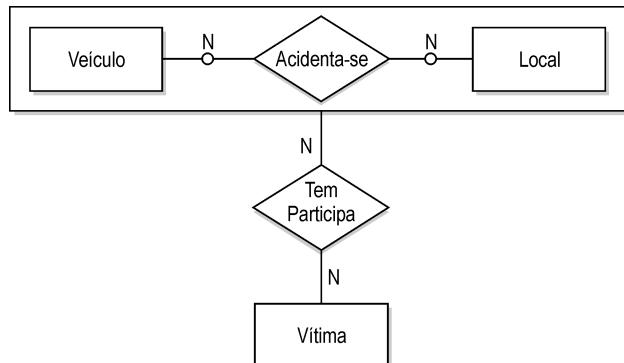
Por que isso é importante para colocarmos como regra?

Porque em certos casos a conectividade vai nos apresentar ou fazer entender que a "coisa" é um atributo do relacionamento e não uma entidade, o que elimina a utilização da agregação.

No caso, quando um veículo acidenta-se em local, pode existir uma ou várias vítimas.

E uma vítima é vítima de um acidente, ou pode também ser muito azarada e aparecer como vítima em muitos acidentes.

Se existir conectividade desta forma, temos um relacionamento entre a entidade vítimas e o bloco de modelo veículo acidenta-se em local do tipo muitos-para-muitos.



E como colocamos as outras perguntas feitas?

Vejamos:

Imprudência ou excesso de velocidade dá a entender que toda vez que houver um acidente existe uma causa.

Descobrimos mais um objeto, as causas possíveis de um acidente.

Mas pode haver mais de uma causa em um acidente? Claro que pode, pois se o acidente for com dois veículos, pode ser que um estava em excesso de velocidade, o motorista alcoolizado, e o outro participante foi imprudente, pois entrou em uma via preferencial.

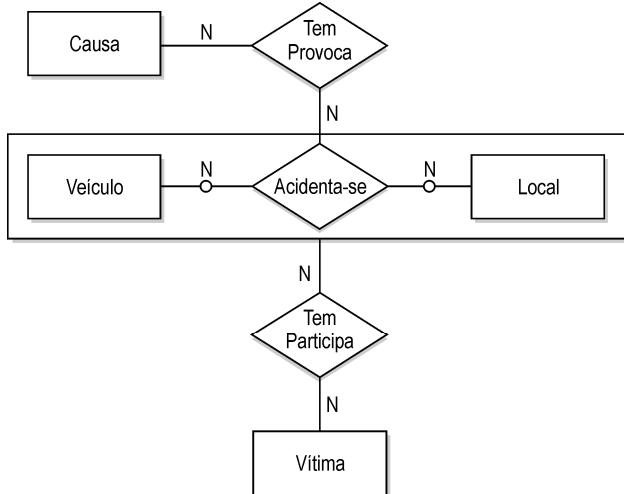
Logo, podemos ter outra entidade relacionada com a agregação, a entidade causas de acidentes de trânsito. Está achando o nome muito grande? Isso não deve ser importante ao modelar dados, o importante é expressar a realidade.

Veja adiante a figura que apresenta mais uma entidade relacionada com a agregação.

Duas entidades relacionadas a uma agregação.

Lembre que os relacionamentos de muitos-para-muitos são representados por tabelas associativas que fazem o mesmo papel do relacionamento.

Fica fácil a implementação de um aspecto conceitual na forma de tabelas relacionais, porém é de grande importância no entendimento de uma realidade trabalharmos com este recurso de agregação.



Recomendamos mais uma vez que os analistas procurem sempre modelar no nível e abstração que é o modelo conceitual, para somente após entender e expressar a realidade, para elaborar o modelo lógico e o modelo físico.

As regras que lançamos para descobrir e utilizar agregações servem como orientação para que o leitor possa trabalhar os dados num nível mais alto de abstração, e facilitar o direcionamento de sua visão e observação de um ambiente que está sendo modelado.

A regra número 1 apresenta a condição básica para a existência de uma agregação, porém nada indica que sempre que houver um relacionamento de muitos-para-muitos haja uma agregação.

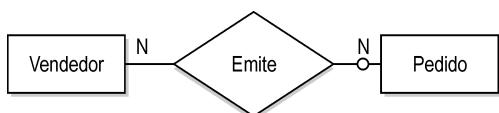
Agregação é enxergar um bloco de modelo que possui um relacionamento de muitos-para-muitos como uma entidade, relacionado com outras entidades, ou então relacionado com outro bloco de modelo, isto é, com outra agregação.

Destacamos no livro que *nunca existe agregação quando da existência de relacionamentos um-para-muitos*. Nunca, não esqueça!

Vamos agora apresentar uma situação de relacionamento um-para-muitos e discutir por que nunca existe a aplicação do conceito de agregação para estes casos.

Uma situação típica de vendas:

Vendedor emite pedido como na figura.

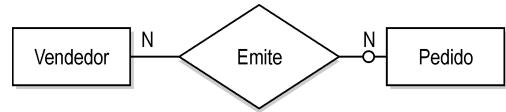


Este relacionamento tem a cardinalidade de um-para-muitos, ou seja, cada ocorrência de vendedor pode ter emitido vários pedidos. Por seu lado uma ocorrência de pedido foi emitida por uma e somente uma ocorrência de vendedor.

Vamos então procurar se existem coisas associadas ao fato de vendedor emite pedido.

Supondo que afirmem que quando vendedor emite pedido implica em ter cliente.

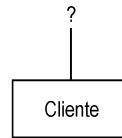
Até parece que estamos no mesmo caminho.



Quem tem cliente?

Pedido ou vendedor.

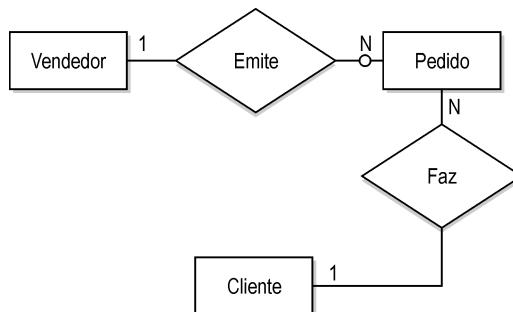
Só existe cliente quando vendedor emite pedido?



Não, a verdade é que cliente é uma informação, um atributo do pedido e não da relação entre as duas entidades.

Na vida real se prestarmos bem atenção ao que acontece, veremos que as informações são de uma entidade, ou de outra, ou ainda caracterizam outra entidade que se relaciona com uma das duas em análise, mas não com o fato representado pelo relacionamento entre elas.

Vejamos na próxima figura a solução correta.



Cliente, vendedor e pedido.

No mundo real um cliente faz um pedido a um vendedor, que o emite, ou melhor, que o encaminha à empresa.

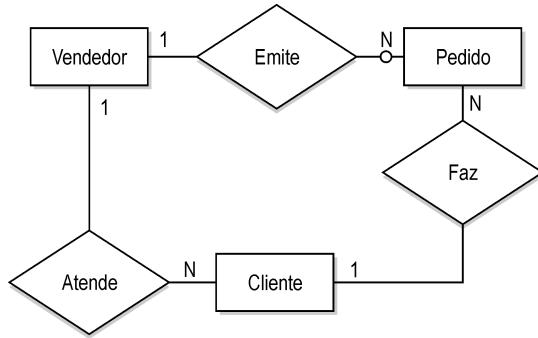
Logo, temos as três entidades como apresentadas no modelo da figura.

Lendo o modelo, vamos encontrar as mesmas afirmações do mundo real.

- ▶ Cliente faz pedido
- ▶ Vendedor emite pedido

E a relação, ou melhor, relacionamento entre vendedor e cliente.

Vendedor atende cliente.



Cuidado com vendedor atende cliente, pois dependendo das regras de negócio da empresa, isso não existirá.

Se a regra de negócio for que os vendedores da empresa não têm exclusividade sobre os clientes, isto é, mais de um vendedor pode atender um mesmo cliente, este relacionamento perde a sua razão de ser.

Neste caso somente através da entidade pedido podemos saber que vendedor atendeu a um determinado cliente.

Esta solução da figura parece ter dois relacionamentos redundantes, porém são dois caminhos alternativos para pedido e cliente em relação a vendedor.

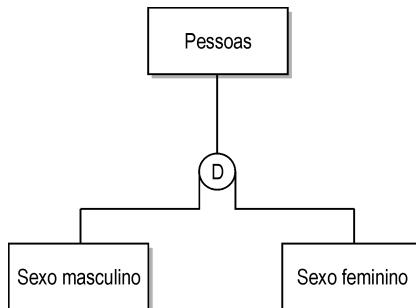
Agregação Reflexiva

Este é um dos casos mais trabalhosos de visualização de realidades.

Vamos colocar exemplos bem reais, assim como uma brincadeira simulada para que você possa gravar, mentalizar os caminhos de solução.

Vamos iniciar este estudo com uma pequena brincadeira.

Suponha que em uma cidade, de poucos habitantes, resolvemos fazer um controle das pessoas que namoram. Aquele negócio de ninguém é de ninguém.

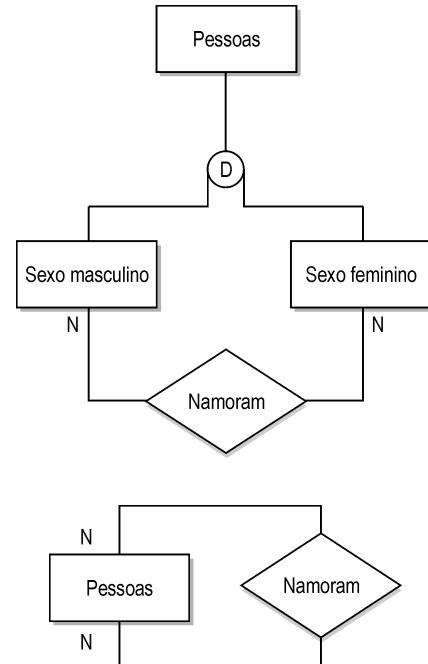


Temos a entidade pessoas da cidade na qual estão, por assim dizer, todos os habitantes, seja do sexo masculino, feminino.

Considerando que uma ocorrência do sexo masculino pode namorar com mais de uma do sexo oposto, e que o caminho no sentido contrário também é verdadeiro, ou seja, uma ocorrência do sexo feminino também pode namorar com mais de uma do sexo oposto, logo temos um relacionamento reflexivo de cardinalidade muitos-para-muitos.

Observe, mesmo rindo, que estamos lidando com duas visões de dados da entidade pessoa. A visão ocorrências do sexo masculino e a visão ocorrências do sexo feminino. Temos dois subtipos distintos neste caso.

Veja a figura que apresenta este relacionamento, que possui dados, pois ali estão a data e a hora e talvez até a indicação do local em que o fato aconteceu.



Observe duas representações gráficas, uma com o entendimento generalista e outra com visões de dados (especializações) colocadas como subtipos, o que vai facilitar o entendimento da realidade.

Mas vamos seguir para obter o resultado de entendimento que buscamos, apesar de ser possível afirmar que não estamos realizando modelagem sexual.

Na referida pequena cidade, como é normal neste Brasil afora, existem vários motéis, muito frequentados.

Analisando o modelo obtido até este instante, podemos colocar a questão:

As pessoas quando namoram utilizam um motel?

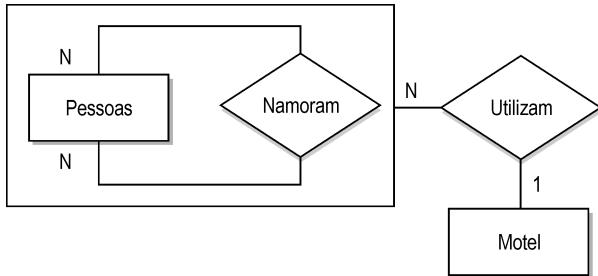
Pelo que falam do movimento que eles têm, certamente utilizam.

Bem, a nossa entidade motel então se relaciona com o quê?

A questão que colocamos tem como resposta:

Motel se relaciona com a agregação resultante do relacionamento entre duas visões de dados da entidade pessoas.

Observe o resultado no diagrama Entidade-Relacionamento.



Pessoas e motel.

Lendo o modelo, temos:

Pessoas do sexo masculino namoram com várias pessoas do sexo feminino, e pessoas do sexo feminino igualmente namoram com várias pessoas do sexo masculino. Continuando.

Quando pessoas namoram com pessoas utilizam um motel.

Este relacionamento, da agregação namora com motel, é de muitos-para-um, pois uma relação só pode ocorrer cada vez em um lugar.

Certamente este exemplo não sairá mais da sua memória, pois quando ele é apresentado, o resultado de fixação é muito alto.

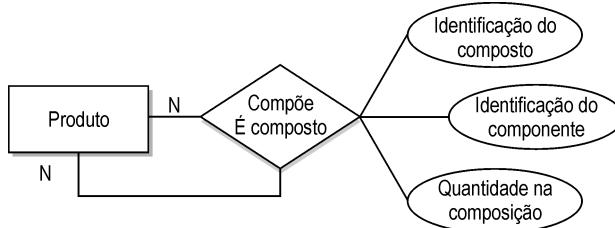
O resultado é bom para entender, isto é!

Um aluno absorveu bem este modelo de dados, inclusive incrementou o seu resultado, com a inserção da entidade marcas de camisinha, pois segundo ele, quando alguém namora com alguém, ainda mais nestes tempos de AIDS, sempre deve usar camisinha de alguma marca e até cor. Concordamos que esta entidade poderia existir, pois teria fins estatísticos.

Este modelo retrata mesmo uma realidade. O problema é, se implementado, como se obteriam os dados para os relacionamentos se efetuarem? Difícil, não acha?

Vamos agora para um exemplo muito utilizado em ambiente industrial, a questão dos produtos compostos e produtos componentes.

A figura apresenta o modelo de relacionamento reflexivo que vamos estudar e agragar.



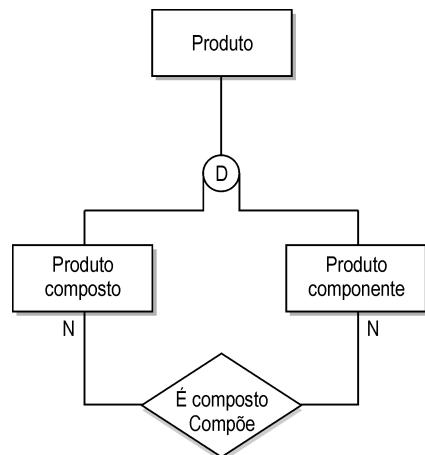
Produto Composto e Componente

Em uma entidade produto suas ocorrências são compostas de produtos completos e componentes.

Um produto acabado é composto de vários componentes.

Um determinado componente participa da composição de vários produtos acabados.

Esses fatos estão retratados no modelo de dados da figura ao lado na forma de subtipos com um relacionamento muitos-para-muitos entre eles.



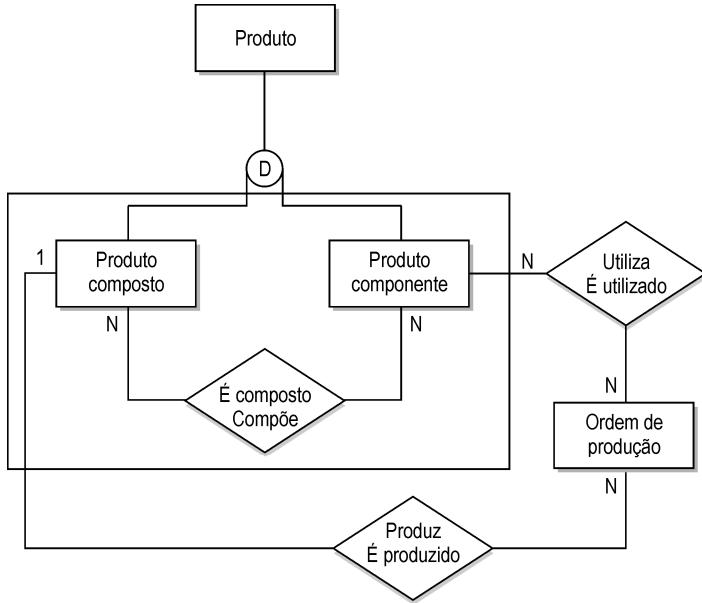
Este relacionamento possui os dados de quantidade de cada componente na composição do produto acabado, conforme destacamos na figura anterior.

Produto

Código Produto	Descrição
3758	Parafuso
8596	Rosca
4512	Arruela
5532	Bloco
7626	Carburador

É Composto - Compõe

Código Composto	Código Componente	Quantidade Componente
5532	3758	10
5532	4512	22
5532	8596	14
7626	3758	65
7626	4512	70



Quando há uma ordem de produção para produzir uma quantidade determinada de um produto acabado, essa ordem provoca a retirada, dos estoques da empresa, das quantidades necessárias de cada componente.

Essa retirada é realizada para atender uma ordem de produção, que produz somente um item acabado, o qual tem componentes.

Essa retirada, ou melhor, requisição de componentes é visualizada pelo relacionamento entre ordem de produção e uma agregação que é o relacionamento entre o subtipo produto acabado e produto componente, o relacionamento compõe/é composto, conforme apresentamos no diagrama ER.

Desta forma, permite-se que se associe a quantidade necessária de cada produto componente para ser utilizada em uma ordem de produção determinada.

Vamos inserir a demonstração das tabelas que representariam com dados este relacionamento utiliza e a ordem de produção para um melhor entendimento.

Ordem de Produção

NumOrdProd	ProdutoOrdem	Qtde	Data Ordem
200152	5532	2	20/2/2004
200155	7626	2	21/3/2004

Utiliza / É utilizado

NumOrdProd	ProdutoOrdem	Componente	Qtde Requisitada
200152	5532	3758	20
200152	5532	4512	44
200152	5532	8596	28
200155	7626	3758	130
200155	7626	4512	140

A tabela que representa o relacionamento entre ordem de produção e a agregação possui como chave estrangeira a concatenação das colunas ProdutoOrdem que é o produto composto a ser fabricado e a coluna Componente que é o produto componente que participa desse produto a ser fabricado.

Voltando ao caso que estávamos estudando no modelo ER, de cinemas e filmes, vamos concluir o contexto, inserindo uma assertiva que não colocamos para poder utilizar os recursos de agregação.

Uma Distribuidora de Filmes - O Retorno

Vamos considerar agora adicionalmente que desejamos controlar o público que assiste aos filmes exibidos nos cinemas.

Queremos saber por horário de sessão, se na sessão das 14 horas, ou das 22 horas, por exemplo, e as datas em que ocorreu maior público.

Entrou um objeto neste contexto que não existia anteriormente: a sessão de cinema.

Se lhe perguntarem quando você foi ao cinema pela última vez, você certamente responderá que foi talvez ao cinema A, passando o filme B, no dia tal, na sessão das X horas.

Bem, olhando por cima, já sabemos que sessão tem informação sobre dia e horário, além de indicar um filme e um cinema.

Ou seja, existe um objeto sessão, só que devemos considerar que não é um objeto com existência independente, pois não existe sessão sem filme e tampouco sessão sem cinema.

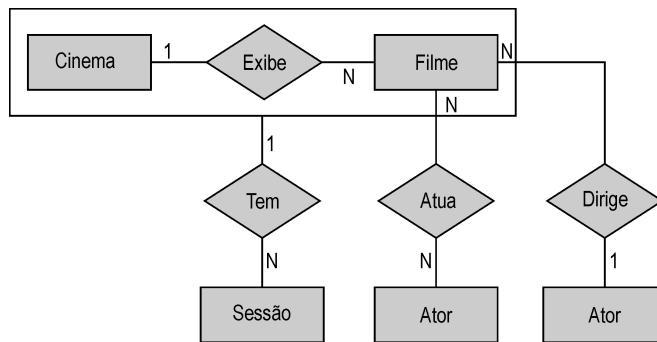
Logo, sessão somente existe quando cinema exibe filme.

A expressão, quando aqui colocada, resulta que temos uma aplicação de agregação entre uma entidade sessão e o conjunto, bloco de modelo cinema passa filme, que já havíamos modelado.

Vamos então ver como fica o modelo de dados agora.

Antes vamos colocar mais uma questão: as pessoas são identificadas e registradas quando entram para assistir a uma sessão de cinema? É claro que não, logo nossa sessão de

cinema deve ter um atributo para que seja possível identificar o público no sentido de quantidade de pessoas que assistiram a um filme em exibição em um cinema.



Uma sessão de cinema tem um cinema e um filme, ou seja, sua conectividade em relação ao bloco agregado é um. E um cinema exibe filme tem muitas sessões, logo a conectividade neste sentido é muitos.

Conclusão

O nosso objetivo neste capítulo foi apresentar uma visão mais ampla da técnica de modelagem de dados, ampla no sentido de retratar com fidelidade as situações do mundo real.

Agregação não é considerada por nós somente uma técnica a mais, porém uma abstração que utilizamos no dia a dia sem perceber que estamos modelando alguma coisa.

O raciocínio do homem é normalmente binário. Observamos e lemos todas as nossas realidades aos pares de coisas.

E quando uma terceira ou quarta coisa entra no meio, utilizamos sempre a expressão **quando**.

Mas dificilmente você interpreta ou lê algo com mais de duas coisas ao mesmo tempo.

No Velho Oeste os pistoleiros conseguiam sempre atirar com duas armas, mas nunca atiravam num determinado instante com mais de duas. Isso faz parte da polaridade binária do sistema nervoso do ser humano.

Neste ponto do livro podemos apresentar exercícios mais complexos, pois consideramos que você adquiriu conhecimentos sobre o modelo ER, inclusive com todas as suas extensões propostas, o que permite a construção de qualquer modelo necessário como os que vamos apresentar.

TRATAMENTO DE INTERPRETAÇÕES DE DADOS

Pontos de Vista Diferentes

A interpretação da realidade difere muito de uma pessoa para outra. Mesmo estando no mesmo lugar e assistindo a um mesmo fato, pessoas diferentes, de ambientes diferentes, podem interpretar e denominar fatos e coisas de forma desigual, além de relacionarem essas coisas com outras completamente diversas.

O desenho ao lado apresenta um cometa, mas que pode ser uma estrela cadente para outras pessoas.



Assim como algumas relacionam este objeto com o fim do mundo, outras relacionam com seres extraterrestres.

Os modelos de dados devem sempre refletir a interpretação que pessoas de uma organização têm de coisas que significam algo para elas. Quando estamos modelando, é muito comum encontrarmos muitas interpretações diferentes em departamentos da empresa sobre a mesma coisa.

Por outro lado, em função das ferramentas existentes para modelagem de dados não implementarem essas interpretações, assim como a existência de direcionamento para o modelo físico, leva o analista a pensar imediatamente em SQL e seus recursos para tratamento de interpretações de uma entidade.

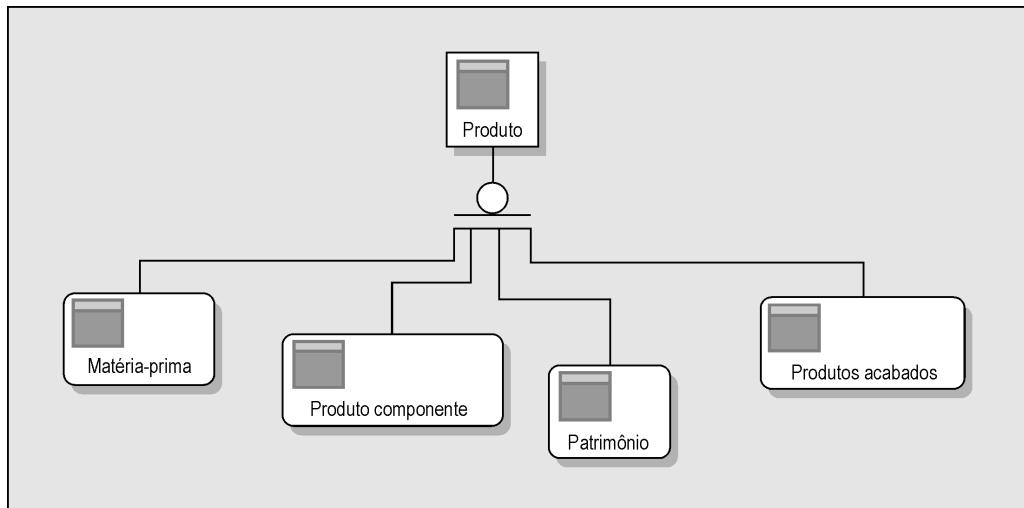


Cria-se então uma distância enorme entre o modelo físico, que é um esquema de tabelas e colunas, e o modelo conceitual, que vai ser apresentado aos usuários, o qual é a representação de um ambiente real, externo ao SGBD.

O que buscamos é uma atenção maior à questão da representação e modelagem das interpretações de dados que existem, não só em departamentos de uma empresa, como os diversos fatos que estão relacionados com interpretações de uma determinada coisa em uma empresa.

Lembre-se de que muitas vezes o não tratamento e modelagem dessas interpretações dentro de um modelo nos levam a não definir relacionamentos entre entidades, uma vez existindo relacionamentos específicos de uma interpretação de dados com outra entidade, ou com outra interpretação. Este problema faz com que fatos do ambiente sejam retratados de maneira confusa, gerando no futuro divergências entre a realidade da empresa e o que o sistema retrata com falta de informações.

A figura seguinte apresenta uma situação de interpretações de dados simples, em que temos a entidade produto e que possui os mais variados tipos de interpretação dentro de uma empresa. Como tratar isso?



Toda empresa, seja comercial, industrial ou de serviços, tem essa coisa chamada produto, porém as interpretações em cada negócio são em maior ou em menor número, mas podemos afirmar que sempre existirá um conjunto de interpretações idêntico.

Devemos resolver o problema dando atenção e explorando mais acentuadamente as possibilidades de interpretações dentro de uma entidade.

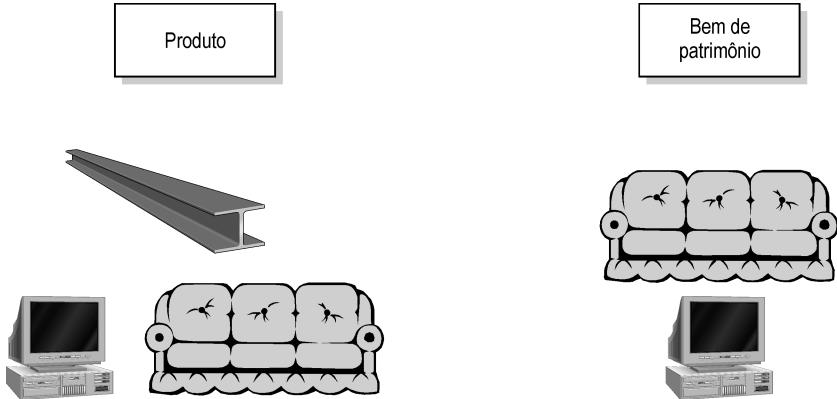
No nosso exemplo, temos ainda o fato de usuários diferentes utilizarem nomenclaturas diferentes para suas coisas. Por exemplo, materiais de consumo e não produtos de consumo.

Muitas vezes esta coisa chamada produto provoca erros graves de modelagem, ou modelos não otimizados, pois coisas iguais são modeladas separadamente.

Produto acabado e matéria-prima.

Ora, os dois não são coisas iguais, por que possuem tratamento diferente?

Ou qual a diferença entre produto acabado e item de patrimônio? São coisas distintas dentro de um negócio, apesar de um produto fornecido por alguém ser um item imobilizado, um bem patrimonial.



A figura ilustra este problema, principalmente em empresas do segmento industrial. Podemos ter um produto acabado que, mudando de área de negócio, é um bem do patrimônio. Se produzirmos poltronas, ou computadores, quando um produto destes está no estoque, é um produto acabado; quando utilizamos um desses produtos na empresa, ele passa a ser um bem imobilizado.

Veja bem que não estamos falando de processos, mas em análise de dados, de entendimento do papel dos objetos no mundo de negócios.

Vamos apresentar conceitos sobre esses conflitos de nomes e entendimento da semântica dos objetos no mundo real.

Existem duas fontes de conflito de nomes em um ambiente real em análise: sinônimos e antônimos.

Sinônimo é quando um mesmo objeto de um ambiente é representado por nomes diferentes, mesmo sendo o mesmo objeto.

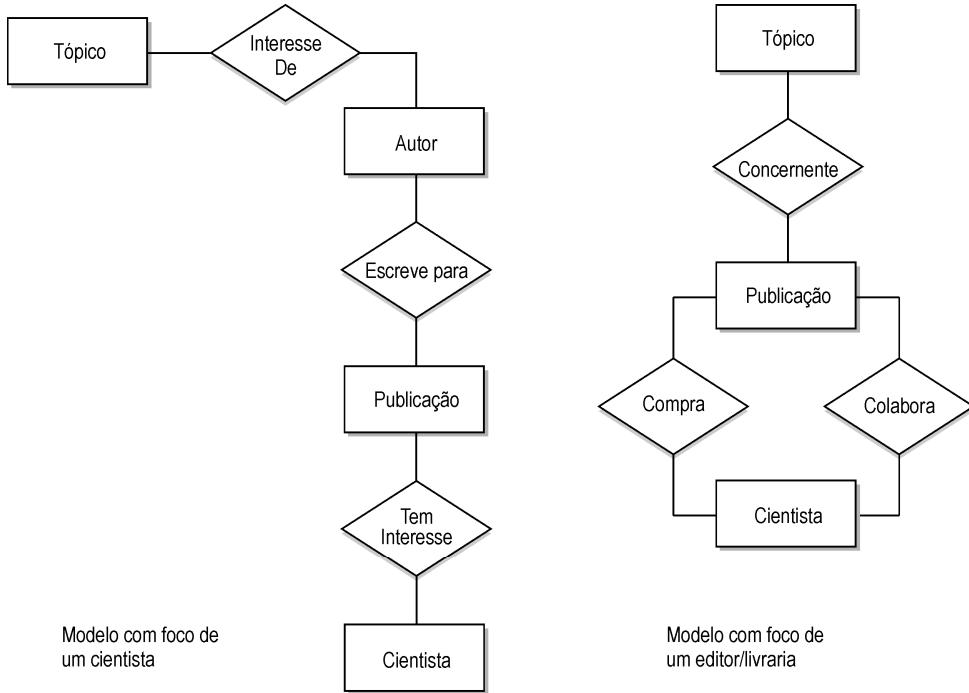
Sinônimo indica o mesmo objeto com nomes diferentes.

Antônimo é quando objetos diferentes são representados pelo mesmo nome no ambiente.

Antônimo, mesmo nome, coisas diferentes.

Devemos estar atentos ao executarmos um modelo de dados para conseguir obter clareza no entendimento dessas similaridades de conceitos do usuário que nos fornece informações.

Os dois modelos de conceitos da figura seguinte apresentam os mesmos objetos com foco de entendimento de negócio diferenciado, e inclusive com conceitos diferentes em relação aos objetos que compõem o ambiente de negócios. Isso não impede que seja realizada uma unificação de conceitos. Porém, destaca-se a necessidade de utilizarmos o que denominamos de renomear os conceitos, removendo as ambiguidades do modelo causadas por sinônimos, e nos casos de antônimos apresentar sempre uma segunda interpretação com os objetos renomeados.



Esta análise de similaridade deve ser realizada principalmente quando conceitos com nomes diferentes possuem as mesmas propriedades, o que permitirá definirmos objetos únicos ou diferentes.

Devemos estar atentos aos conceitos com mesmo nome e que possuam propriedades e restrições diferentes. Lembrando que restrições para nós querem dizer regras e condições que restringem os conjuntos de valores válidos para o objeto.

A prática tem mostrado que exemplos mais simples podem ser mais fáceis de visualizar, porém devemos nos orientar para apresentar os objetos e conceitos que existem no ambiente do usuário e da forma mais próxima possível de seu vocabulário e conceitos.

Devemos buscar uma forma de apresentar, integrar e valorizar essas visões de dados para possibilitar que os modelos sejam realmente legíveis, não somente por um grupo de usuários de um futuro sistema, mas por todos os envolvidos na utilização desse sistema.

Relacionamentos entre Interpretações

No modelo de dados é importante tratar os relacionamentos que essas interpretações têm entre si e com outras entidades.

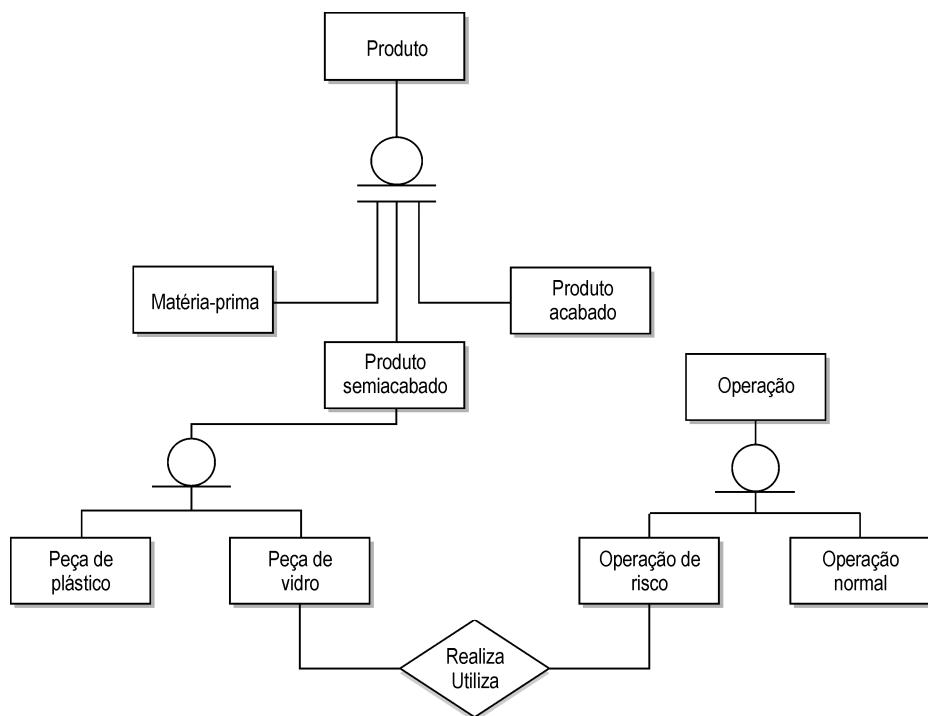
Vamos iniciar pelos casos de relacionamentos entre interpretações de uma entidade. Observe a figura em que apresentamos o relacionamento entre interpretações de produto.

Na entidade produto temos as seguintes interpretações: material de consumo, produtos acabados, produtos semiacabados, componentes de fabricação, matérias-primas e material de marketing. Temos ainda as subinterpretações de produtos semiacabados: peças de plástico e peças de vidro.

Peças de vidro é uma interpretação de um determinado grupo de pessoas dentro da empresa, do departamento de engenharia, e que possui forma de classificação e visualização de coisas totalmente diferente da administração, por exemplo.

Da mesma forma existem as interpretações de operação. Dois tipos de operação são considerados, as operações normais e as de risco, uma qualificação para diferenciar as operações em uma fábrica hipotética.

No nosso exemplo, peças de vidro se relacionam com as interpretações de operações denominadas de operações de risco.



Por exemplo, produto de consumo, que não aparece na figura, poderia relacionar-se com o quê?

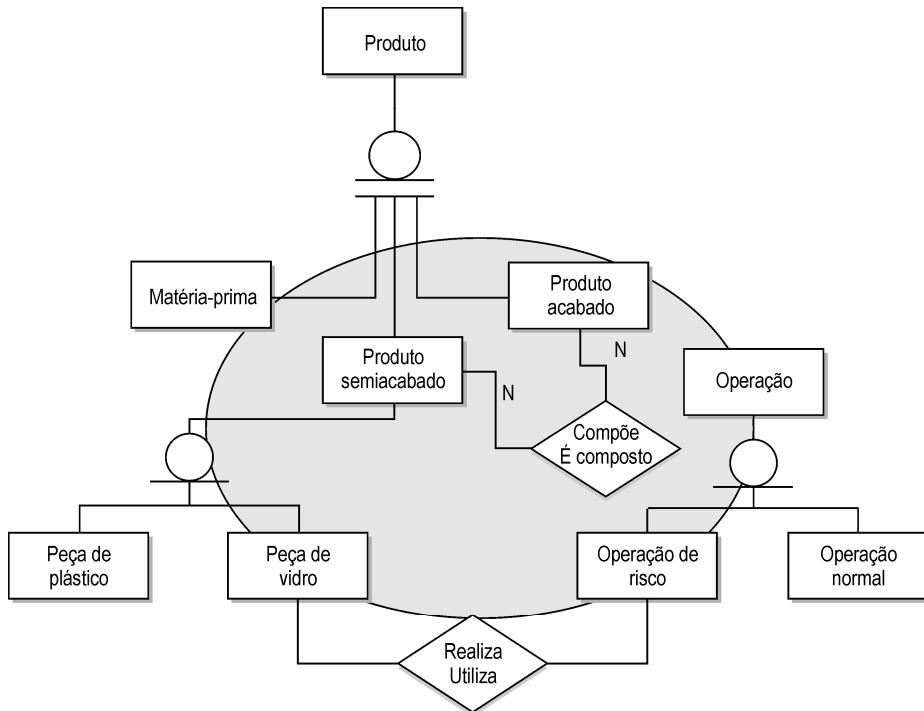
Com nenhuma das interpretações apresentadas, porém, pode existir relacionamento entre produtos acabados e produtos semiacabados?

Sim, pois um produto acabado considerando-se que nossa empresa modelo fabrica, por exemplo, garrafas térmicas, possuiria em sua composição produtos semiacabados.

São produzidos diversos produtos em uma linha de montagem os quais vão formar uma garrafa térmica.

Logo, a figura seguinte apresenta o relacionamento entre as duas interpretações.

Um relacionamento de muitos-para-muitos, que colocamos conceitualmente com dados, contém a informação de quem é o componente e quem ele compõe, ou seja, qual o produto final de cuja composição ele participa.



Aprofundando a análise, qual será a jogada com matéria-prima?

Produto acabado não está relacionado com matéria-prima?

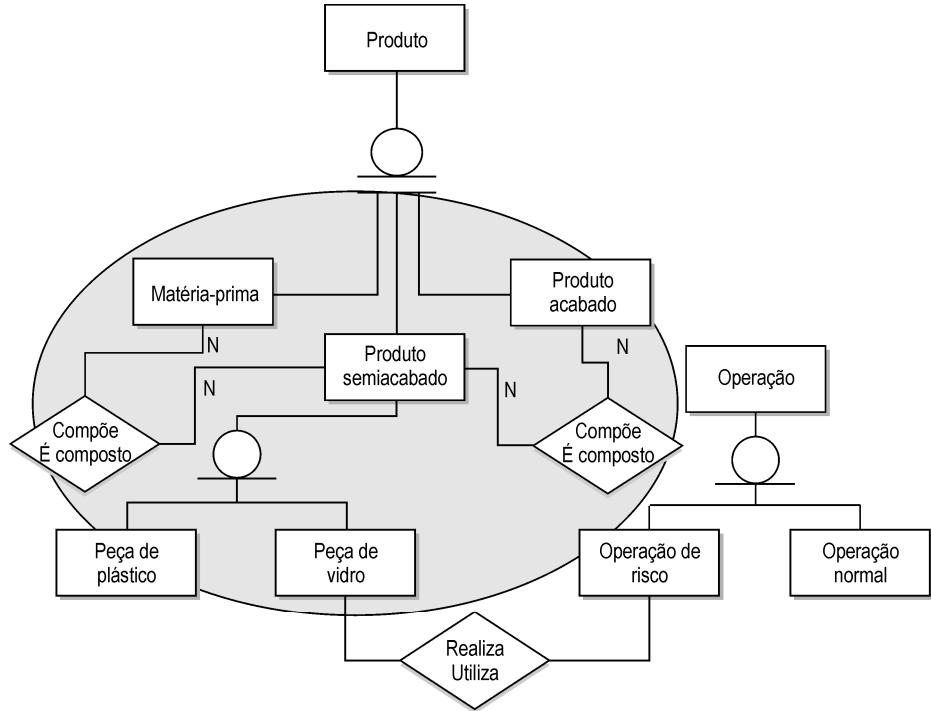
Neste exemplo vamos inicialmente considerar que matéria-prima está sendo utilizada para produzir os componentes de um produto acabado.

Logo, teríamos um relacionamento entre as interpretações produto semiacabado e matéria-prima da mesma forma que temos entre produto acabado e produto semiacabado.

Mas estes dois relacionamentos são os mesmos?

Evidente que são.

Na realidade existe um único grande relacionamento que é produto compõe produto, e que pode ser visto em grupos separados, porém estruturalmente é o mesmo relacionamento. O que temos na realidade é uma hierarquia embutida na entidade produto.



Tratamento de Subinterpretações

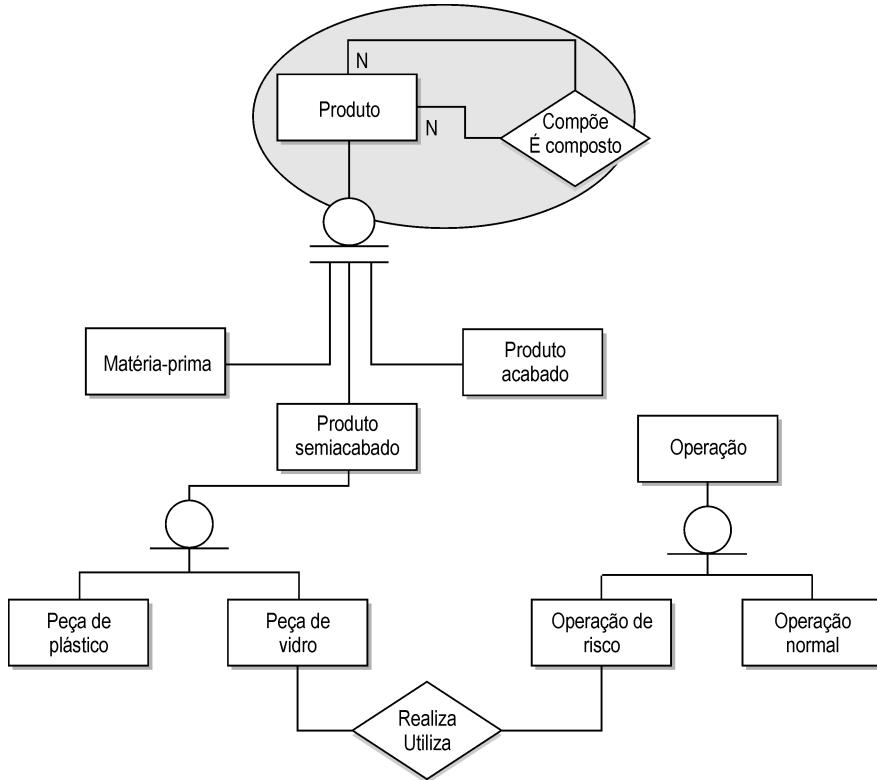
Mas poderíamos agora nos perguntar: e a interpretação peças de vidro não faz parte de uma garrafa térmica? (a parte interna, o bojo da garrafa). Como relacioná-la?

O importante é entendermos que nem todas as interpretações estão no mesmo nível hierárquico. Podemos ter interpretações dentro de outra interpretação.

Neste caso peças de vidro é uma interpretação dentro da interpretação produto semiacabado. Observando a realidade em questão, seu relacionamento se dá com a interpretação produto acabado, conforme apresentamos, porém nada mais é que um segundo grau de especialização do modelo em questão.

Veja a figura em que vamos simplificar o modelo e apresentar uma existência dessa subinterpretação, além de, é claro, apresentar um relacionamento reflexivo geral denominado Compõe entre Produto e o próprio Produto (reflexivo). Esta é uma linguagem que o departamento de engenharia e seus profissionais entendem.

Para que o modelo não tenha sua legibilidade prejudicada, devemos desenhar as interpretações com os relacionamentos específicos de cada uma, mas sem perder a apresentação dos relacionamentos que não são exclusivos de uma interpretação.



Necessariamente uma interpretação, ou melhor, subinterpretação de dados, não precisa ser apresentada através de um relacionamento com a entidade na qual ela está inserida. Ou seja, não há necessidade de perdemos tempo em relacionar, por exemplo, peças de vidro com produto acabado.

O modelo não perde seu entendimento por não ser inserido um relacionamento específico para cada subinterpretação.

O mais importante é que peças de vidro têm este destaque como uma subinterpretação de produtos semiacabados através do entendimento da área de engenharia. Entretanto, também há relacionamentos com outros objetos do modelo, e que são específicos dessa subinterpretação, pois nem todas as ocorrências da interpretação denominada de produtos semiacabados se relacionam com a interpretação denominada de operações de risco.

Quando é apresentada uma interpretação ou uma subinterpretação de dados, devemos observar que ela existe porque possui algum tipo de tratamento significativo no negócio.

Isso quer dizer que quando uma coisa tem sua existência afirmada, ela está relacionada de forma peculiar com as outras coisas significativas do negócio.

Mas todos serão unânimes em afirmar que as ferramentas de modelagem não permitem representar essas interpretações de forma ampla e ainda com subinterpretação.

A maioria dos cursos de modelagem de dados não apresenta corretamente a questão da subinterpretação, o que provoca que a maioria dos analistas não explore em seus modelos de dados esse recurso.

Muitos são os casos que analisamos em que não foi tratada a subinterpretação, ficando o modelo sem retratar a realidade, e o que é mais grave, quando da modelagem dos eventos cria-se uma confusão, pois os eventos estão suportados exatamente pela subinterpretação. Perde-se o que de mais importante existe para um modelo de dados, a sua expressividade do mundo real.

Quando da modelagem de algum objeto, coisa significativa de uma empresa, gostamos de explorar as possibilidades de classificação desse objeto, pois isso leva a identificar subtipos e suas divisões existentes, assim como as hierarquias que podem existir na relação entre os conceitos levantados.

Vamos então explorar mais alguns casos normais que encontrarmos de identificação de interpretação de dados.

A implementação de visões mesmo em SQL é pouco utilizada na implementação do esquema físico do banco de dados por provocar problemas de performance, mas ainda não concordamos que performance seja o fator fundamental em hipótese nenhuma no momento de modelarmos.

Se sua aplicação tratar seus programas através das visões de dados, é porque você as definiu, logo não se preocupe com a forma de implementação no momento da modelagem. Siga o princípio básico de um bom modelista de dados. Não pense em processos enquanto modela; retrate a realidade das coisas de uma empresa.

Mais Interpretação

A maioria das literaturas do mercado somente explora interpretação ou subtipos e supertipos dentro de um exemplo comum e simples de um primeiro entendimento, a entidade pessoas.

Não vamos fugir deste exemplo, mas vamos procurar explorá-lo mais profundamente, considerando não o conceito puro de subtipo, porém o conceito amplo de especialização, que nada mais é que a técnica de representação das interpretações.

Existe um ponto em que a representação de uma interpretação fica mais complexa em função de que as ferramentas de modelagem são dirigidas a ambiente relacional, e as interpretações resultantes de uma entidade e um relacionamento necessitem de vários níveis de hierarquia.

Como não existe implementação desse tipo de interpretação em SGBDs relacionais, muitas vezes ela provoca a inserção de flags muito estranhos em domínio dentro dos modelos de dados, gerando resultados ambíguos e de pouco controle.

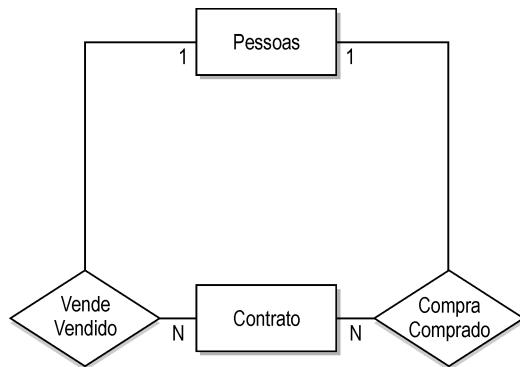
Seja como na figura a entidade pessoas e entidade contratos de venda.

Qualquer pessoa existente na entidade pessoa pode efetuar uma venda de um contrato, assim como pode ser cliente de um contrato de venda.

Num Contrato	Vendedor	Cliente
45				12	45
85				11	74
32				84	56
21				101	94

Identificação	Nome
12				Adão
11				João
45				Cláudia
74				Pedro
84				Sérgio
101				Ivan
56				Maurício
94				José

O relacionamento existe entre pessoas e contratos, porém a interpretação vendedor é derivada exclusivamente do relacionamento entre as duas entidades. Não é uma interpretação exclusiva e caracterizada por um agrupamento específico que possua um identificador de categoria, mas uma interpretação resultante somente da existência de um relacionamento, em que ocorrências desta entidade possuem um papel denominado de vendedor.



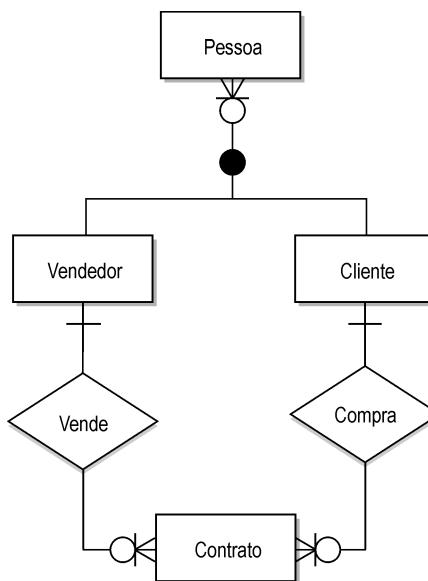
Uma pessoa pertence à interpretação vendedor somente quando tiver realizado uma venda, ou seja, quando existir uma ocorrência de contrato de venda que indique aquela pessoa como o vendedor.

Observe as tabelas demonstrativas para um melhor entendimento.

No contrato de número 32, o vendedor é a ocorrência de pessoa com identificação 84 (Sérgio) e o cliente é a ocorrência de pessoa com identificação 56 (Mauricio).

Como apresentar este modelo, como apresentar a interpretação vendedor se não é simplesmente um subtipo, uma especialização?

Infelizmente as ferramentas de modelagem não permitem que se criem alias ou roles de entidades para permitir essa representação, porém o nosso desenho, o nosso diagrama ER, deve apresentar um submodelo (subject area) em que existirá um subtipo com os mesmos atributos da entidade funcionário, e sem que exista nenhum atributo identificando o papel vendedor, podendo, isto sim, apresentar que existe o papel por meio de uma cláusula no modelo, especificando-o:



Pessoa é vendedor, quando em contrato de venda a identificação do vendedor for igual à identificação de pessoa.

Vamos ampliar o problema de interpretação, ampliando o universo da entidade pessoas, em função do tratamento e da forma como são considerados os clientes de uma empresa.

Se considerarmos que os clientes em um primeiro momento são pessoas físicas e não empresas, a entidade possui dados de pessoas.

Logo, pessoa é uma entidade generalizadora que possui ocorrências de clientes, funcionários e vendedores por assim dizer.

Num diagrama de mais alto nível há, como na figura apresentada, dois relacionamentos entre contrato e pessoas. O relacionamento compra e o relacionamento vende.

Pessoas estão relacionadas com contratos pelo fato de serem vendedores de ocorrências de contratos, e por outro lado também existe outro relacionamento que é pessoas que são clientes de um contrato.

Sem que se abra no diagrama a representação das visões resultantes e condicionadas à existência do relacionamento, nosso modelo poderia suscitar perguntas como: mas alguém pode ser vendedor e cliente de um mesmo contrato?

No momento em que apresentamos as interpretações e os relacionamentos com cada uma delas, estamos com um resultado de interpretação da realidade que não será questionado, pois estamos qualificando e destacando os papéis, perfeitamente como no mundo real.

Este último diagrama apresentado tem um nível semântico de maior grau, permitindo desta forma seu entendimento e representatividade do mundo real de melhor qualidade.

Quando retratamos em um modelo de dados um ambiente de negócios, não devemos poupar a inserção de interpretações, pois elas são fundamentais ao entendimento desse ambiente.

A utilização de generalização e especialização em modelos de dados é um dos artifícios da técnica de maior poder de expressão. Infelizmente, voltamos a salientar, é pouco explorado e utilizado pelos analistas de sistemas.

Vamos avançar as interpretações de dados.

Explorar potenciais interpretações pode tornar uma ferramenta muito útil no domínio de negócios, quando de sua modelagem.

Vamos discutir um enfoque de interpretação de dados muito comum na maioria dos sistemas corporativos existentes hoje, a questão cliente e fornecedor para começarmos.



Em uma organização, cliente é ocorrência de alguma coisa que mantém relação de compra de produtos ou serviços.

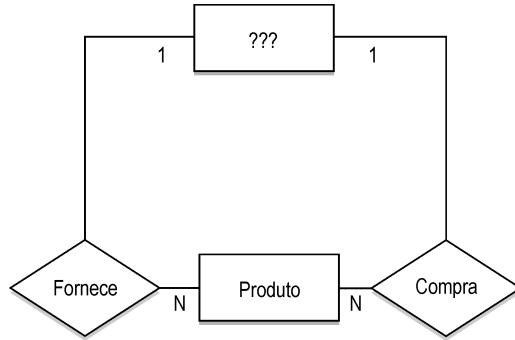
Fornecedor é a ocorrência de alguma coisa de quem a organização adquire bens ou serviços por assim dizer.

A questão que sempre é colocada é que uma ocorrência de cliente pode ser em determinado momento uma ocorrência de fornecedor e vice-versa.

O que impede de que, quem me forneça serviços, também compre meus produtos e seja meu cliente? Nada!

Logo, a existência de venda ou de compra é que determina a interpretação de dados, e não existe neste caso a caracterização específica da entidade cliente e da entidade fornecedor.

Não se trata de querer otimizar redundâncias!



Somente estamos analisando que existe uma entidade ainda sem nomenclatura, que engloba ocorrências de clientes e de fornecedores.

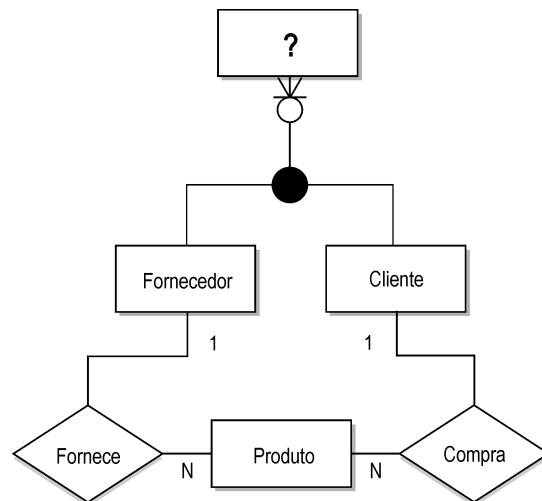
A maioria dos analistas tem o mau costume de dar o nome de CliFor a essa entidade.

Urg! Que coisa horrorosa!

Nada diz nada este nome de significativo, pois continua limitado.

Observe a próxima figura e veja que o conceito de cada interpretação é dependente do relacionamento que cada uma possui.

Qual nomenclatura daríamos a essa entidade?



Vamos continuar a explorar as possíveis interpretações ou descobrir coisas significantes no ambiente em análise.

Quando estamos tratando com compras de produtos, ou com venda, temos um elemento que se denomina normalmente de transporte de mercadorias. Logo, temos quem transporta, os transportadores.

Mas transportador não é fornecedor?

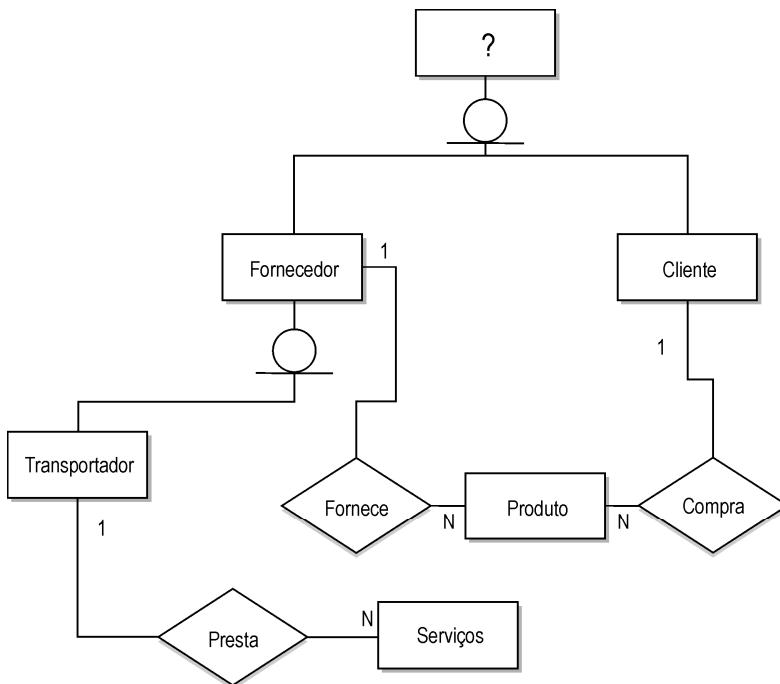
Sim, é fornecedor de serviços.

Da mesma forma que a oficina do Zezinho na esquina, em que nós pagamos e temos a comprovação com recibos!

Então temos na mesma entidade mais uma ou duas interpretações, ou melhor, temos uma nova interpretação e subinterpretações e assim por diante.

A figura seguinte mostra o nível de especialização a que chegamos somente com esta rápida análise.

Mas os clientes podem ser pessoas físicas e pessoas jurídicas.



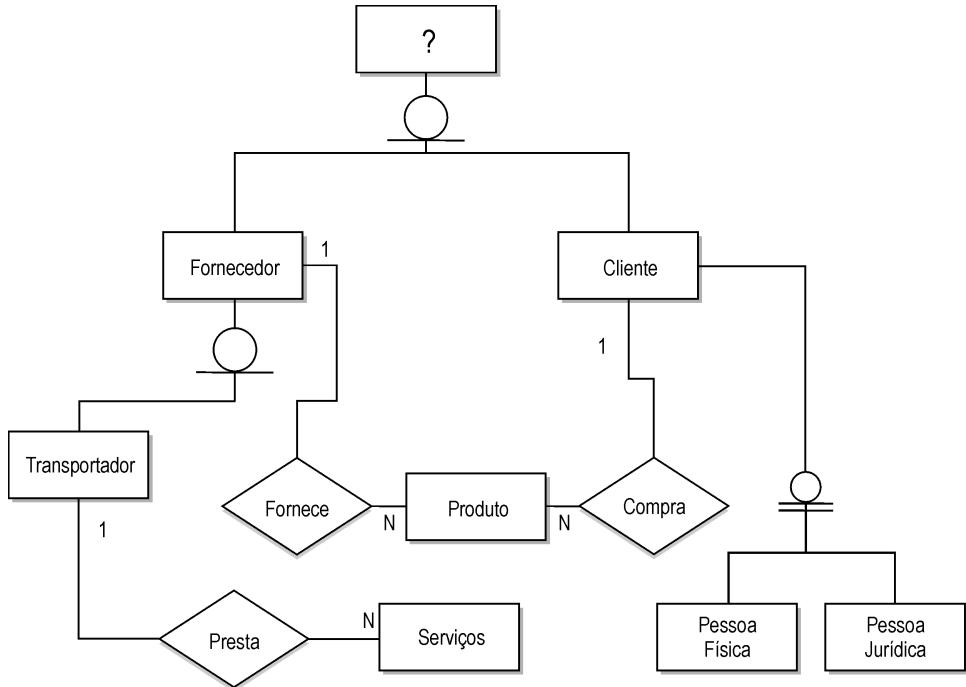
Complicou mais, porque existem tratamentos diferentes na área comercial.

As vendas a pessoas físicas são realizadas por departamento específico da empresa, que vende, por exemplo, a varejo, separadamente das vendas realizadas a pessoas jurídicas, as empresas, as quais compram por atacado (grandes quantidades).

Logo, a interpretação clientes possui suas especializações, suas subinterpretações.

Devemos considerar que estamos realizando a determinação ainda macro de entidades e interpretações, pois não demonstramos até este instante nenhum atributo específico de cada entidade ou interpretação de dados.

Modelo com Cliente Pessoas Jurídicas e Pessoas Físicas



Neste diagrama somente inserimos a classificação de entidade subtipo para pessoa física e pessoa jurídica.

O relacionamento compra não foi afetado, pois a realidade continua a mesma, somente temos uma classificação diferenciada para clientes.

A análise das tipologias de um objeto é pouco explorada inclusive nas técnicas orientadas a objetos, considerando-se que, dentro dos conceitos de mensagem, essas mensagens são específicas de cada tipo de interpretação.

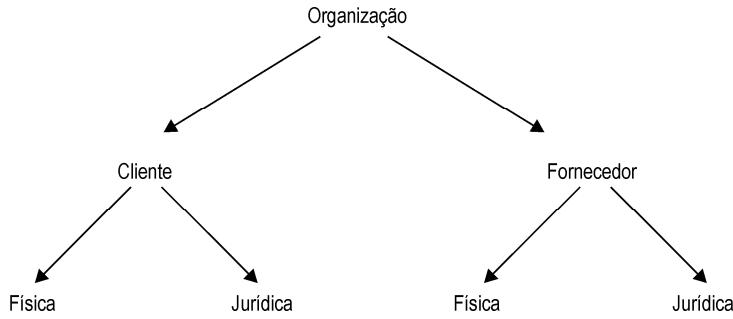
Vamos ler o modelo apresentado e interpretá-lo.

Antes acreditamos que já é possível revelar o nome desta entidade generalizadora: entidade pessoa.

Então, pessoa pode ser Pessoa Física ou Pessoa Jurídica.

Fornecedor é uma especialização que absorve tanto pessoa física quanto jurídica. Da mesma forma cliente absorve tanto pessoa física quanto jurídica.

É importante que se faça um diagrama de estrutura inicial com a árvore de visões que oriente na forma hierárquica em que elas estão inseridas.



Porque a interpretação de subinterpretações somente é possível se entendermos a sua relação de encapsulamento com a entidade básica do modelo. A figura exibe este diagrama de forma simples, somente estruturando essa hierarquia.

Diagrama Hierárquico de Interpretações

O modelo de dados que vai representar esta situação está no modelo ER a seguir, em que simulamos de forma macro os relacionamentos que criam as interpretações de cliente, fornecedor e transportador. Os subtipos de cada uma dessas interpretações são suas especializações através de características de atributos que indicam se é organização física ou jurídica.

Vamos apresentar uma leitura simples deste modelo para que você compreenda a realidade que estamos apresentando:

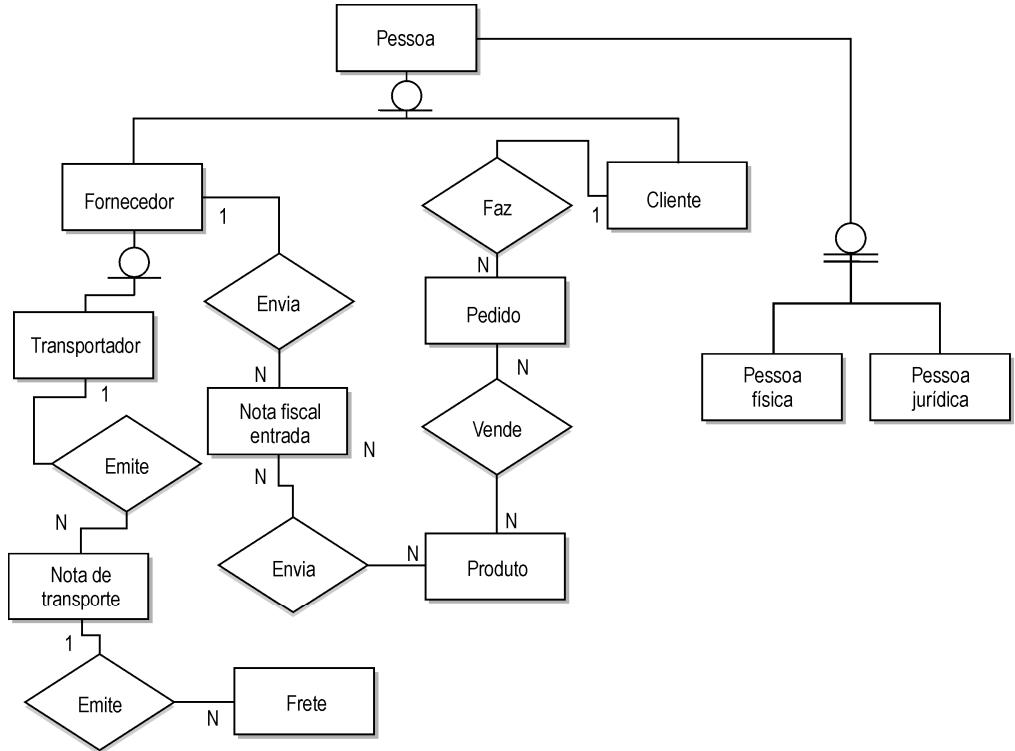
Uma ocorrência de pessoa é **cliente** quando se relaciona com pedido de venda, podendo ser jurídica se o atributo do tipo da organização for, por exemplo, "J".

Uma ocorrência de pessoa é fornecedor quando se relaciona com nota fiscal de entrada.

Expandindo o modelo em grau de especialização, podemos ter o resultado da figura apresentada em seguida.

As nossas notas fiscais de entrada poderiam também ser controladas ou identificadas como dois tipos: notas fiscais de transporte ou notas fiscais de entrada de produtos.

Logo, podemos ter que uma ocorrência do subtipo fornecedor é pertencente à interpretação transportador quando está relacionada com a interpretação notas fiscais de transporte.



A amplitude do número de interpretações é inerente às formas de tratamento da informação nos diversos departamentos de uma empresa.

Nosso modelo, ou melhor, o diagrama ER deve representar sempre que possível essas interpretações, pois elas são esclarecedoras do ambiente que está sendo modelado.

Poderíamos ainda apresentar um modelo com o desdobramento mais detalhado de fornecedor, inserindo um subtipo como fornecedor de produtos para deixar completa a classificação da coisa fornecedor.

Conclusão

Interpretações de dados são estratégicas em um modelo de dados.



A não prospecção das interpretações durante os levantamentos para obtenção e no refinamento de um modelo de dados, ou simplesmente a aplicação dos conceitos de generalização não enfocando acentuadamente as interpretações de dados, por conceitos de SQL, ou de implementação física de bancos de dados, seguramente vai conduzir à construção de modelo de dados falho e incompleto.

Além disso, não vai permitir ganhos de produtividade na análise de eventos do sistema, pois se você não tem todas as classificações de um conceito, também não terá a absorção e entendimento completo dos eventos que são específicos de cada uma dessas classificações.

A seguir estão algumas questões que, se respondidas, podem ajudar a descobrir interpretações e subinterpretações de dados nos trabalhos de modelagem de dados.

1. Que fatos estão relacionados com somente algum subconjunto de ocorrências e uma entidade?
2. Que interpretação de dados no modelo se relaciona com um subconjunto da entidade que estamos analisando?
3. Existe algum fato que se relate com somente parte de um subconjunto de dados?
4. Pessoas diferentes e de departamentos diversos se referenciam a uma coisa com nomes diferentes?
5. Coisas de nomes diferentes não são subconjuntos de uma única coisa?

Cuidado, Abra o Olho!

Coisas de nomes iguais podem ser totalmente diferentes e caracterizarem duas entidades, assim como destacamos que coisas de nomes diferentes podem representar o mesmo conceito. Nesse momento devemos parar e observar os atributos que descrevem esses objetos.



Desta forma podemos entender dois objetos que não são interpretações, mas duas entidades diferentes em nosso modelo, ou entender a existência de classificações em múltiplos níveis em um modelo de dados Entidade-Relacionamento.

NORMALIZAÇÃO

Conceituação

O objetivo da normalização é evitar os problemas que podem provocar falhas no projeto do banco de dados, bem como eliminar a **mistura de assuntos** e as correspondentes redundâncias desnecessárias de dados.

Uma regra que devemos observar quando do projeto de um banco de dados orientado para o modelo relacional é **não misturar assuntos em uma mesma tabela**.

O processo de normalização aplica uma série de regras sobre as tabelas de um banco de dados, para verificar se estão corretamente projetadas. Embora exista um conjunto de cinco formas normais (ou regras de normalização), na prática usamos desse conjunto efetivamente três formas normais.

As tabelas de um banco de dados relacional são derivadas de um modelo Entidade-Relacionamento, e muitas vezes nessa derivação temos muitos problemas de performance, integridade e manutenção de dados.

Normalmente após a aplicação das formas normais, algumas tabelas acabam sendo divididas em duas ou mais tabelas, o que no final gera um número maior de tabelas do que o originalmente existente.

Esse processo causa a simplificação dos atributos de uma tabela, colaborando significativamente para a estabilidade do modelo de dados, reduzindo consideravelmente as necessidades de manutenção.

Neste capítulo vamos entender o processo de normalização na prática por meio de exemplos e estudos de caso para a sua aplicação.

Por exemplo, a tabela que apresentamos em seguida traz os dados produtos, vendedores, clientes e pedidos de venda.

Observe na tabela que as informações de produto e cliente estão armazenadas de forma redundante, desperdiçando espaço.

Nome Produto	Pedido Número	Nome Cliente	Endereço Cliente	Limite de Crédito	Data	Nome Vendedor
Limpadora a Vácuo	1458	Davi Bachmamm	Rio de Janeiro	US\$ 5,000	05/05/00	Carlos Book
Computador	2730	Helena Daudt	Vancouver	US\$ 2,000	05/06/00	João Hans
Refrigerador	2461	José Stolaruck	Chicago	US\$ 2,500	07/03/00	Silvio Phergüns
Televisão	456	Pedro Albuquerque	São Paulo	US\$ 4,500	09/05/00	Frederico Raposo
Rádio	1986	Carlos Antonelli	Porto Alegre	US\$ 2,000	18/09/00	Rui Ments
CD Player	1815	Davi Bachmamm	Rio de Janeiro	US\$ 5,000	18/04/00	Silvio Phergüns
Limpadora a vácuo	1963	C.V. Ravishandar	Bombaim	US\$ 7,000	03/01/00	Carlos Book
Limpadora a vácuo	1855	Carlos Antonelli	Porto Alegre	US\$ 2,000	13/05/00	João Hans
Refrigerador	1943	Davi Bachmamm	Rio de Janeiro	US\$ 5,000	19/06/00	Silvio Phergüns
CD Player	2315	Davi Bachmamm	Rio de Janeiro	US\$ 5,000	15/07/00	João Hans

Para uma seleção do tipo: quais os clientes que têm pedido de limpadora a vácuo no último mês, é preciso fazer uma pesquisa em toda a tabela, comparando linha a linha a coluna produto, para encontrar as linhas em que essa coluna tenha valor de limpadora a vácuo.

Por outro lado, uma atualização do endereço de um cliente da mesma forma exige que se faça a consulta de todas as linhas em que o nome do cliente seja igual ao desejado e se realize a alteração nessas linhas. Da mesma forma a deleção do único pedido da cliente Helena vai apagar a única cópia de seus dados de endereço e limite de crédito.

Estes fatos são considerados anomalias de inclusão, alteração e deleção de dados.

Então podemos definir que normalização consiste em definir o formato lógico adequado para as estruturas de dados das tabelas de um banco de dados relacional, identificadas no projeto lógico do sistema, com objetivo de minimizar o espaço utilizado pelos dados e garantir a integridade e confiabilidade das informações.

A normalização é feita pela análise dos dados que compõem as estruturas utilizando o conceito chamado formas normais (FN).

As formas normais são conjuntos de restrições as quais os dados devem satisfazer.

Por exemplo, pode-se dizer que a estrutura está na primeira forma normal (1FN), se os dados que a compõem satisfizerem as restrições definidas para esta etapa.

A normalização completa dos dados é feita seguindo as restrições das cinco formas normais existentes. A passagem de uma forma normal para outra é feita tendo como base o resultado obtido na etapa anterior, ou seja, na forma normal anterior.

Para realizar a normalização dos dados, é primordial que seja definido para a tabela objeto da normalização um campo de chave primária para a sua estrutura de dados, o qual permite identificar os demais campos da estrutura.

O conceito de normalização foi introduzido para o modelo relacional por Edgar F. Codd, em 1970 (primeira forma normal). Essa técnica é baseada em um processo matemático formal, que tem seus fundamentos na teoria dos conjuntos.

Primeira Forma Normal (1FN)

Uma tabela de dados relacional, como a que apresentamos, não tem colunas repetidas, pois cada coluna tem exatamente uma definição na tabela.

A tabela é então considerada na primeira forma normal. É evidente que como este é o mais baixo nível de normalização, a tabela na primeira forma normal pode ainda conter outras anomalias.

Definição de Primeira Forma Normal

Uma tabela está na primeira forma normal se e somente se todas as colunas possuem um único valor, e não existem grupos repetitivos (colunas) em uma linha ou atributos compostos.

Aplicar a primeira forma normal consiste em retirar da estrutura os elementos repetitivos, ou seja, aqueles dados que podem compor uma estrutura do tipo vetor.

Número da NF	Código do Cliente	Nome do Cliente	Endereço do Cliente	CNPJ do Cliente	Data de Emissão	Total Geral da Nota
456123	1458	Davi Bachmamm	Rio de Janeiro	60890837/0001-85	05/05/00	R\$ 5.421,00
859632	2730	Helena Daudt	Fortaleza	80890575/0001-70	05/06/00	R\$ 6.612,00
859631	2461	José Stolaruck	Maceió	33016338/0002-71	07/03/00	R\$ 1.820,00
745689	456	Pedro Albuquerque	São Paulo	68596006/0001-07	09/05/00	R\$ 453,00
745692	1986	Carlos Antonelli	Porto Alegre	02930076/0002-22	18/09/00	R\$ 184,00
745693	1815	Davi Bachmamm	Rio de Janeiro	71039655/0001-11	18/04/00	R\$ 2.365,00
745694	1963	C.V. Ravishandar	Recife	60890837/0001-85	03/01/00	R\$ 1.112,00
745695	1855	Carlos Antonelli	Porto Alegre	02930076/0002-22	13/05/00	R\$ 1.235,00
745696	1943	Davi Bachmamm	Rio de Janeiro	60890837/0001-85	19/06/00	R\$ 4.150,00
745699	2315	Davi Bachmamm	Rio de Janeiro	60890837/0001-85	15/07/00	R\$ 2.225,00

Código do Produto	Descrição do Produto	Quantidade de Produto	Valor Unitário	Valor Total
45	Limpadora a Vácuo	1	R\$ 600,00	R\$ 600,00
25	Computador	1	R\$ 1.800,00	R\$ 1.800,00
32	Refrigerador	2	R\$ 800,00	R\$ 1.600,00
27	Televisão	3	R\$ 950,00	R\$ 2.850,00
...
32	Refrigerador	4	R\$ 1.235,00	R\$ 4.940,00
27	Televisão	2	R\$ 4.150,00	R\$ 8.300,00
25	Computador	4	R\$ 2.225,00	R\$ 8.900,00

Podemos afirmar que uma estrutura está normalizada na 1FN, se não tiver elementos repetitivos. Acompanhe um exemplo.

Estrutura Original de Notas Fiscais de Venda de Mercadorias

{NUMERO DA NOTA FISCAL, SERIE, DATA EMISSÃO, CODIGO DO CLIENTE, NOME DO CLIENTE, ENDEREÇO DO CLIENTE, CNPJ DO CLIENTE, RELACAO DOS PRODUTOS VENDIDOS (ONDE PARA CADA PRODUTO TEMOS: CÓDIGO DO PRODUTO, DESCRIÇÃO DO PRODUTO, QUANTIDADE VENDIDA, PREÇO UNITÁRIO DE VENDA E TOTAL DA VENDA DESTE PRODUTO) E TOTAL GERAL DA NOTA)}

Já destacamos nesta estrutura de dados uma coluna para identificação única, como chave primária do número da nota fiscal.

Analisando a estrutura, observamos que existem vários produtos em uma única nota fiscal, sendo, portanto, elementos repetitivos que devem ser retirados da estrutura.

Resultado da estrutura após a aplicação da primeira forma normal (1FN):

Tabela de Notas Fiscais

{NUM. NF, SERIE, DATA EMISSAO, CODIGO DO CLIENTE, NOME DO CLIENTE, ENDERECO DO CLIENTE, CNPJ DO CLIENTE E TOTAL GERAL DA NOTA}

Tabela de Item de Nota Fiscal

{NUM. NF, CODIGO DO PRODUTO, DESCRICAO DO PRODUTO, QUANTIDADE VENDIDA, PRECO DE VENDA E TOTAL DA VENDA DESTE PRODUTO}

Observação: Os campos sublinhados identificam as chaves das estruturas.

Número da NF	Código do Cliente	Nome do Cliente	Endereço do Cliente	CNPJ do Cliente	Data de Emissão	Total Geral da Nota
456123	1458	Davi Bachmann	Rio de Janeiro	60890837/0001-85	05/05/00	R\$ 5.421,00
859632	2730	Helena Daudt	Fortaleza	80890575/0001-70	05/06/00	R\$ 6.612,00
859631	2461	José Stolaruck	Maceió	33016338/0002-71	07/03/00	R\$ 1.820,00
745689	456	Pedro Albuquerque	São Paulo	68596006/0001-07	09/05/00	R\$ 453,00
745692	1986	Carlos Antonelli	Porto Alegre	02930076/0002-22	18/09/00	R\$ 184,00
745693	1815	Davi Bachmann	Rio de Janeiro	71039655/0001-11	18/04/00	R\$ 2.365,00
745694	1963	C.V. Ravishandar	Recife	60890837/0001-85	03/01/00	R\$ 1.112,00
745695	1855	Carlos Antonelli	Porto Alegre	02930076/0002-22	13/05/00	R\$ 1.235,00
745696	1943	Davi Bachmann	Rio de Janeiro	60890837/0001-85	19/06/00	R\$ 4.150,00
745699	2315	Davi Bachmann	Rio de Janeiro	60890837/0001-85	15/07/00	R\$ 2.225,00

Número da NF	Código do Produto	Descrição do Produto	Quantidade do Produto	Valor Unitário	Valor Total
456123	45	Limpadora a Vácuo	1	R\$ 600,00	R\$ 600,00
859632	25	Computador	1	R\$ 1.800,00	R\$ 1.800,00
859631	32	Refrigerador	2	R\$ 800,00	R\$ 1.600,00
745689	27	Televisão	3	R\$ 950,00	R\$ 2.850,00

Como resultado desta etapa ocorre um desdobramento dos dados em duas estruturas, a saber:

- ▶ **Primeira estrutura (tabela de nota fiscal):** dados que compõem a estrutura original, excluindo os elementos repetitivos.
- ▶ **Segunda estrutura (tabela de item de nota fiscal):** dados que compõem os elementos repetitivos da estrutura original, tendo como chave o campo-chave da estrutura original (Num. NF) o qual é herdado por dependência, e um campo-chave da estrutura de repetição (Código do Produto).

Vamos ver mais um exemplo de aplicação da primeira forma normal.

Utilizamos um documento que é preenchido por uma suposta instituição bancária.

Ficha de Cliente		
NomCliente		
Endereço		
Rua		
Cidade		
Bairro		
Empréstimos		
Agência	Número Empréstimo	Valor
1	902230	500,00
4	902231	1500,00
3	902240	1200,00

Se visualizarmos agora como uma entidade e, consequentemente, uma tabela, ela teria uma estrutura de dados como segue:

Cliente {NOMCLIENTE, RUA, CIDADE, BAIRRO, (AGENCIA₁, NUMERO EMPRESTIMO₁, VALOR₁, AGÊNCIA₁, AGÊNCIA₂, NUMERO EMPRESTIMO₂, VALOR₂,.....AGENCIA_n, NUMERO EMPRESTIMO_n, VALOR_n)}

em que se observa que o conjunto de atributos AGENCIA, NÚMERO EMPRESTIMO₁, VALOR repete-se indefinidas vezes, ou seja, é um conjunto de atributos multivalorados.

Definimos NOMCLIENTE como a chave primária que identifica o conjunto de informações. Entretanto, temos neste caso uma mistura de assuntos se colocarmos todos estes atributos em uma mesma tabela, além do aspecto de multivaloração.

Ou os dados são de clientes ou são de empréstimos dos clientes.

Aplicando a 1FN, obtêm-se então duas estruturas de dados ou tabelas que representam a realidade do formulário.

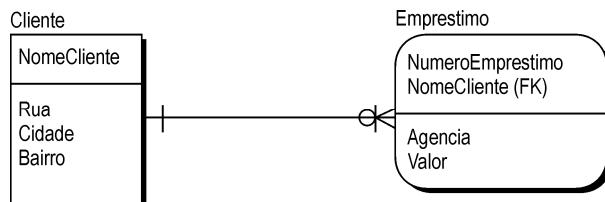
Os atributos repetitivos, ou multivalorados, passam a compor uma entidade, a qual é derivada da entidade principal que seria cliente, com seus dados sem repetição.

Teremos então duas tabelas ou entidades neste caso:

Cliente {NOMCLIENTE, RUA, CIDADE, BAIRRO} e

EmprestimoCliente {NOMCLIENTE, AGENCIA, NUMEROEMPRESTIMO, VALOR}

Logo, de um formulário obtivemos um pequeno modelo de dados.



Observe que a entidade que é derivada no processo de normalização através da aplicação da primeira forma normal herda os atributos da chave primária da entidade, na qual estavam anteriormente inseridos esses dados, ou seja, da qual essa entidade é derivada.

No exemplo, herdou o atributo NOMCLIENTE e compôs com o atributo mais significativo e sem possibilidade de repetição de valor NUMERO EMPRESTIMO.

A primeira forma normal então não permite a existência de atributos multivalorados ou compostos.

Uma relação está na primeira forma normal se possui apenas atributos com valores atômicos (indivisíveis).

Ficha de Empréstimo de Livros					
Histórico de Empréstimos					
Matrícula	Nome	Data Empréstimo	Data Devolução	Data Prevista Devolução	Assinatura

Vamos aplicar agora a primeira forma normal no exemplo de uma ficha de empréstimo de livros em uma biblioteca.

Qual seria a estrutura inicial? Temos assuntos misturados em um único objeto?

Temos ocorrência de valores repetitivos?

Sim, temos informações sobre o livro e seu empréstimo e os valores dos empréstimos são repetidos indeterminadas vezes para cada livro.

Início

Ficha de Empréstimo= {ISBN, NÚMERO DO EXEMPLAR, TÍTULO DO LIVRO, N. VEZES (MATRÍCULA, NOME, DATA EMPRÉSTIMO, DATA DEVOLUÇÃO, DATA PREVISTA DEVOLUÇÃO, ASSINATURA)}

Aplicando a primeira forma normal, extraímos o conjunto de dados repetitivos para compor uma segunda entidade ou tabela, considerando que herdaremos para a chave primária dessa entidade a chave primária original da estrutura que é ISBN, e acrescentamos um dado que seja significativo como identificador para esse conjunto de repetições. Neste caso o dado mais significativo seria a matrícula de quem retira o livro.

Logo, temos como resultado da aplicação da primeira forma normal neste exemplo duas entidades:

Livro {ISBN, NÚMERO DO EXEMPLAR, TÍTULO DO LIVRO} e Emprestimo {ISBN, MATRÍCULA, NOME, DATA EMPRÉSTIMO, DATA DEVOLUÇÃO, DATA PREVISTA DEVOLUÇÃO, ASSINATURA}

Não pergunte como será o valor da informação assinatura, pois não é o objeto deste processo.

Para que seja correto o entendimento das formas normais seguintes, vamos apresentar os conceitos de dependência funcional entre dados.

Dados dois conjuntos de atributos A e B de uma entidade, diz-se que:

- ▶ B é funcionalmente dependente de A ou
- ▶ A determina B ou
- ▶ B depende de A

Vamos ver isto com um exemplo:

Matrícula	Nome	Sobrenome	Departamento
1021	João	Barros	900
1145	Pedro	Silva	700
1689	Antonio	Jardim	900

Departamento determina matrícula? Ou seja, matrícula depende de departamento?

Não, pois departamento 900 está em 1021 e 1689.

Matrícula determina departamento?

Sim, pois se conhecer a matrícula, é possível saber o departamento, já um funcionário só tem um departamento.

Isso quer dizer que as informações de uma entidade devem ser dependentes da informação que é a sua identificadora, sua chave primária.

Papelaria	Artigo	Preço
Colmeia	Caneta Bic	2,50
Central	Cola	1,50
Colmeia	Borracha	1,00
Catedral	Caneta Bic	2,00

Preço é funcionalmente dependente de artigo?

Não, pois o preço pode ser diferente em diferentes papelarias.

Preço é dependente sim do conjunto papelaria e artigo.

Preço é dependente de papelaria?

Não, pois existem tantos valores de preços quantos os artigos vendidos na papelaria.

Dada uma entidade qualquer, dizemos que um atributo ou conjunto de atributos A pertencente a ela é dependente funcional de 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.

Existe uma **dependência funcional** quando o valor de um dado ou dados determina os valores de outros dados.

No exemplo:

Livro {ISBN, NUMERO DO EXEMPLAR, TITULO DO LIVRO}

O número do exemplar e o título do livro dependem do ISBN, pois se mudarmos o valor do ISBN, mudarão o número do exemplar e o título do livro.

Vamos adiante com a aplicação da segunda forma normal que entenderemos melhor a questão de dependência funcional.

Segunda Forma Normal (2FN)

Consiste em retirar das estruturas de dados que possuem chaves compostas (chave primária formada por mais de um campo) todos os dados que são funcionalmente dependentes de somente alguma parte dessa chave.

Podemos afirmar que uma estrutura está na 2FN, se ela estiver na 1FN e não possuir campos que sejam funcionalmente dependentes de parte da chave. Exemplo:

Estrutura na Primeira Forma Normal (1FN)

Tabela de Notas Fiscais = {NUMNF, SERIE, DATA EMISSÃO, CODIGO DO CLIENTE, NOME DO CLIENTE, ENDEREÇO DO CLIENTE, CNPJ DO CLIENTE E TOTAL GERAL DA NOTA}

Tabela de Item de Nota Fiscal = {NUMNF, CODIGO DO PRODUTO, DESCRIÇÃO DO PRODUTO, QUANTIDADE VENDIDA, PRECO UNITARIO DE VENDA E TOTAL DA VENDA DESTE PRODUTO}

Observe que existe uma chave primária composta na tabela ITEM DE NOTA FISCAL.

Item de Nota Fiscal

NumNF	Código do Produto	Descrição do Produto	Quantidade de Produto	Valor Unitário	Valor Total
456123	45	Limpadora a Vácuo	1	600,00	600,00
859032	25	Computador	1	1.800,00	1.800,00
859631	32	Refrigerador Brastemp	2	800,00	1.600,00
745689	27	Televisão LG	3	950,00	2.850,00
485689	45	Limpadora. a Vácuo	2	600,00	1.200,00

As perguntas que colocamos agora definem onde existe e onde não existe dependência funcional total da chave primária:

O valor de Descrição do produto depende de NumNF, CODIGO DO PRODUTO?

Se considerarmos que em cada linha da tabela o produto pode ser descrito de forma diferente, fica ainda mais grave a anomalia nesta tabela.

Mas a resposta é não, pois em realidade a descrição do produto depende somente do código do produto. Ela varia pela mudança do valor do atributo CODIGO DO PRODUTO somente, assim como valor unitário varia de acordo com o valor de código do produto. Logo, estes dois atributos não possuem dependência funcional total da chave primária.

O atributo valor total varia conforme varia o valor do conjunto NumNF, CODIGO DO PRODUTO.

Logo, esse atributo tem dependência funcional total da chave primária, assim como o atributo quantidade.

Uma tabela está na segunda forma normal (2FN) se ela estiver na 1FN e todo atributo não chave é plenamente dependente da chave primária.

A resolução da aplicação da segunda forma normal é realizada através da exclusão dos atributos que não dependem totalmente da chave primária, da tabela original, e constituindo-se com estes uma nova tabela, que terá como chave primária o atributo participante da chave primária da tabela origem.

Neste exemplo a solução será composta de três tabelas agora, considerando a tabela de partida utilizada.

Estrutura na Segunda Forma Normal (2FN)

Tabela de Notas Fiscais (NUMNF, SERIE, DATA EMISSAO, CODIGO DO CLIENTE, NOME DO CLIENTE, ENDERECHO DO CLIENTE, CNPJ DO CLIENTE E TOTAL GERAL DA NOTA)

Tabela de Item de Nota Fiscal (NUMNF, CODIGO DO PRODUTO, QUANTIDADE VENDIDA E TOTAL DA VENDA DESTE PRODUTO)

Tabela de Produto (CODIGO DO PRODUTO, DESCRIÇÃO DO PRODUTO, PRECO UNITÁRIO DE VENDA)

Código do Produto	Descrição do Produto	Valor Unitário
45	Limpadora a Vácuo	600,00
25	Computador	1.800,00
32	Refrigerador Brastemp	800,00
27	Televisão LG	950,00

NumNF	Código do Produto	Quantidade de Produto	Valor Total
456123	45	1	600,00
859032	25	1	1.800,00
859631	32	2	1.600,00
745689	27	3	2.850,00
485689	45	2	1.200,00

Nota Fiscal

NumNF	Série	Data de Emissão	Código Cliente	Nome do Cliente	Endereço do Cliente	CNPJ	Total da Nota
456123	Única	20/02/2004	1	Luis Sampaio	R. Silva Sá 23/11	01.253.523/0001-85	600,00
859032	Única	20/02/2004	2	Carlos Pereira	R. Dias Melhores 334/122	32.253.501/0001-12	1.800,00
859631	Única	20/02/2004	3	José Alves	Av. Arapanés 4487/1915	005.333.510/0001-08	1.600,00
745689	Única	20/02/2004	4	Luis Paulo Souza	R. Botica do Ouvidor 44	11.111.111/0007-01	2.850,00
485689	Única	20/02/2004	2	Carlos Pereira	R. Dias Melhores 334/122	32.253.501/0001-12	1.200,00

Como resultado desta etapa, houve um desdobramento da tabela de item de nota fiscal (a tabela de notas fiscais ainda não foi alterada por não ter chave composta) em duas estruturas, a saber:

- ▶ **Primeira estrutura (tabela de item de nota fiscal):** contém os elementos originais, sendo excluídos os dados que são dependentes apenas do campo código do produto.
- ▶ **Segunda estrutura (tabela de produto):** contém os elementos que são identificados apenas pelo código do produto, ou seja, independentemente da nota fiscal, a descrição e o preço unitário de venda serão constantes.

Quando um atributo ou conjunto de atributos A de uma tabela depende de outro atributo B, que não pertence à chave primária, mas é dependente funcional deste, dizemos que A é dependente transitivo de B.

As dependências funcionais podem existir para atributos que não são chaves em uma tabela. Este fato denominamos de dependência funcional transitiva.

Terceira Forma Normal (3FN)

A terceira forma normal determina que não devem existir atributos com dependência funcional transitiva em uma tabela, pois podem provocar da mesma forma anomalias de inclusão, manutenção e deleção.

A aplicação da terceira forma normal consiste em retirar das estruturas os campos que são funcionalmente dependentes de outros campos que não são chaves.

Podemos afirmar que uma estrutura está na terceira forma normal se ela estiver na segunda forma normal e não possuir campos dependentes de outros campos não chaves. Exemplo:

Estruturas na Segunda Forma Normal (2FN)

Tabela de Notas Fiscais (NUMNF, SERIE, DATA EMISSAO, CODIGO DO CLIENTE, NOME DO CLIENTE, ENDEREÇO DO CLIENTE, CNPJ DO CLIENTE E TOTAL GERAL DA NOTA)

Tabela de Item de Nota Fiscal (NUMNF, CODIGO DO PRODUTO, QUANTIDADE VENDIDA E TOTAL DA VENDA DESTE PRODUTO)

Tabela de Produto (CODIGO DO PRODUTO, DESCRIÇÃO DO PRODUTO, PRECO UNITARIO DE VENDA)

Nota Fiscal							
NumNF	Série	Data de Emissão	Código Cliente	Nome do Cliente	Endereço do Cliente	CNPJ	Total Nota
456123	Única	20/02/2004	1	Luis Sampaio	R. Silva Sá 23/11	01.253.523/0001-85	600,00
859032	Única	20/02/2004	2	Carlos Pereira	R. Dias Melhores 334/122	32.253.501/0001-12	1.800,00
859631	Única	20/02/2004	3	José Alves	Av. Arapanés 4487/1915	005.333.510/0001-08	1.600,00
745689	Única	20/02/2004	4	Luis Paulo Souza	R. Botica do Ouvidor 44	11.111.111/0007-01	2.850,00
485689	Única	20/02/2004	2	Carlos Pereira	R. Dias Melhores 334/122	32.253.501/0001-12	1.200,00

Observa-se que na tabela de notas fiscais os dados NOME DO CLIENTE, ENDEREÇO DO CLIENTE, CNPJ DO CLIENTE dependem funcionalmente da coluna CODIGO DO CLIENTE, a qual não faz parte da chave primária desta tabela, logo têm dependência funcional transitiva.

Estas colunas são então retiradas e vão constituir uma nova tabela levando como chave primária o atributo do qual possuíam dependência transitiva.

Estrutura de Dados na Terceira Forma Normal (3FN)

Tabela de Notas Fiscais (NumNF, SERIE, DATA EMISSAO, CODIGO DO CLIENTE E TOTAL GERAL DA NOTA)

Existe agora uma nova estrutura, ou nova tabela:

Tabela de Clientes (CODIGO DO CLIENTE, NOME DO CLIENTE, ENDEREÇO DO CLIENTE E CNPJ DO CLIENTE)

Cliente

Código do Cliente	Nome do Cliente	Endereço do Cliente	CNPJ
1	Luis Sampaio	R. Silva Sá 23/11	001.253.523/001-85
2	Carlos Pereira	R. Dias Melhores 334/122	032.253.501/001-12
3	Jose Alves	Av. Arapanés 4487/1915	005.333.510/001-08
4	Luis Paulo Souza	R. Botica do Ouvidor 44	011.111.111/007-01
2	Carlos Pereira	R. Dias Melhores 334/122	032.253.501/001-12

Como resultado desta etapa, houve um desdobramento do arquivo de notas fiscais, por ser o único que tinha campos que não eram dependentes da chave principal (Num. NF), uma vez que agora independentes de número da nota fiscal, o nome, endereço e CNPJ do cliente são inalterados.

Esse procedimento permite evitar inconsistência nos dados das tabelas e economizar espaço por eliminar o armazenamento frequente desses dados e sua repetição.

A cada nota fiscal emitida para um cliente não mais haverá o armazenamento desses dados, não havendo divergência entre eles.

NotaFiscal

NumNF	Série	Data de Emissão	Código do Cliente	Total Nota
456123	Única	20/2/2004	1	600,00
859032	Única	20/2/2004	2	1.800,00
859631	Única	20/2/2004	3	1.600,00
745689	Única	20/2/2004	4	2.850,00
485689	Única	20/2/2004	2	1.200,00

As estruturas de dados foram alteradas pelos seguintes motivos:

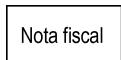
- ▶ **Primeira estrutura (tabela de notas fiscais):** contém os elementos originais, sendo excluídos os dados que eram dependentes apenas do campo código do cliente (informações referentes ao cliente).
- ▶ **Segunda estrutura (tabela de clientes):** contém os elementos que são identificados apenas pelo código do cliente, ou seja, são independentes da nota fiscal, o nome, endereço e CNPJ dos clientes serão constantes.

Após a normalização, as estruturas dos dados estão projetadas para eliminar as inconsistências e redundâncias dos dados, eliminando desta forma qualquer problema de atualização e operacionalização do sistema.

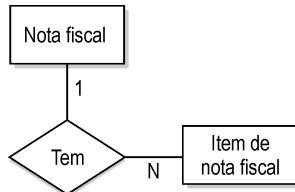
A versão final dos dados pode sofrer alguma alteração, para atender às necessidades específicas do sistema, a critério do analista de desenvolvimento durante o projeto físico do sistema.

Se analisarmos em termos de modelo de dados Entidade-Relacionamento, a evolução do seu processo de normalização é da sequência de figuras que apresentamos:

Antes da primeira forma normal:

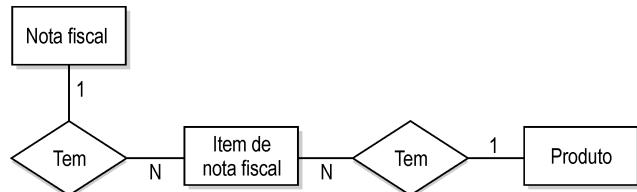


Após a aplicação da primeira forma normal:



Temos duas entidades e um relacionamento.

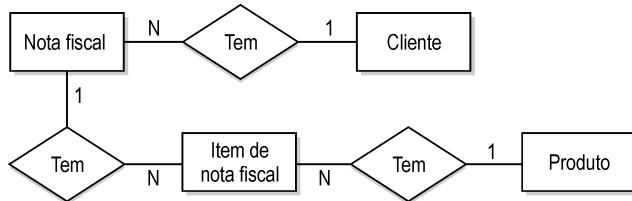
Após a aplicação da segunda forma normal:



Existem agora três entidades e dois relacionamentos.

Finalmente, após a aplicação da terceira forma normal temos quatro entidades e três relacionamentos.

A aplicação do processo de normalização sobre documentos, formulários e relatórios existentes em um ambiente em análise pode transformar-se em um instrumento de descoberta de objetos e conceitos muito útil ao processo de modelagem Entidade-Relacionamento.



Forma Normal de Boyce/Codd (FNBC)

As definições da 2FN e 3FN, desenvolvidas por Codd, não cobriam certos casos. Essa 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.
- ▶ As chaves candidatas sejam concatenadas (mais de um atributo).
- ▶ As chaves concatenadas compartilhem 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. Note que esta definição é em termos de chaves candidatas e não sobre chaves primárias.

Vamos imaginar a entidade **filho** com os seguintes atributos:



Por hipótese, vamos assumir que um professor possa estar associado a mais de uma escola e uma sala.

Com esta suposição tanto a chave (candidata) concatenada NOME-DA-ESCOLA + SALA-DA-ESCOLA como NOME-DA-ESCOLA + NOME-DO-PROFESSOR podem ser determinantes.

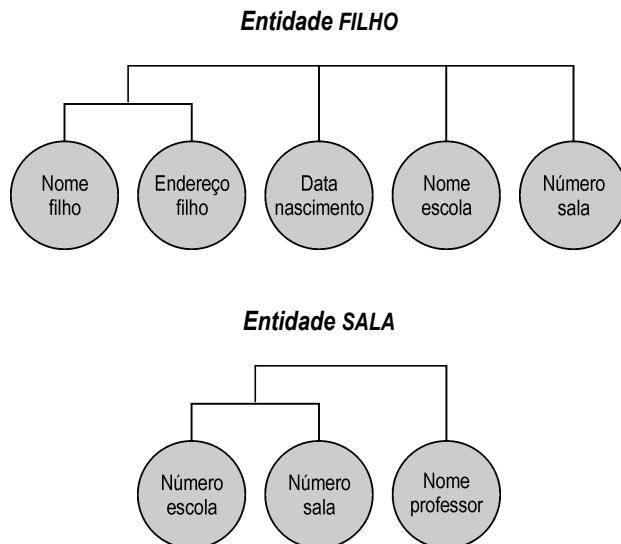
Logo, esta entidade atende às três condições relacionadas anteriormente:

- As chaves candidatas para a entidade FILHO são: NOME-DO-FILHO + ENDERECO-DO-FILHO, NOME-DO-FILHO + NUMERO-DA-SALA e NOME-DO-FILHO + NOME-DO-PROFESSOR;
- As três chaves apresentam mais de um atributo (concatenados).
- 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 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 aplicar FNBC, a entidade FILHO deve ser dividida em duas entidades, uma que contém todos os atributos que descrevem o FILHO e a segunda que contém os atributos que designam um professor em uma escola em particular e número de uma sala.



Os casos para aplicação dessa forma normal a FNBC são mais raros de encontrar na prática, pois derivam de erros de modelagem realizados quando da estruturação dos atributos de uma entidade.

Vamos buscar mais um exemplo de forma que fique clara a utilização da FNBC e seja visto algo um pouco mais próximo de uma realidade prática e uma definição mais ampla para a FNBC:

Uma tabela R está na FNBC com respeito a conjunto F de dependências funcionais da forma $A \rightarrow B$, em que $A \subseteq R$ e $B \subseteq R$ se ao menos se realizam:

- ▶ $A \rightarrow B$ é uma dependência funcional trivial, isto é, $B \subseteq A$.
- ▶ A é uma superchave para a tabela R.

Sejam as tabelas semelhantes às que já utilizamos neste livro para uma aplicação bancária.

Entidade Cliente

Cliente

NomCliente	RuaCliente	CidadeCliente
Luis Sampaio	Rua A	Rio de Janeiro
Carlos Pereira	Rua B	Niterói
Jose Alves	Rua C	Rio de Janeiro
Luis Paulo Souza	Rua B	Niterói

- ▶ $\text{Cliente} = \{\text{NOMCLIENTE}, \text{RUACLIENTE}, \text{RUACLIENTE}\}$

Podemos afirmar que a entidade cliente está na FNBC, pois uma chave candidata para esta entidade (tabela) é NomCliente.

Observando, temos:

- ▶ $\text{NOMCLIENTE} \rightarrow \text{RUACLIENTE}, \text{CIDADECLIENTE}$

Isso denota que não existem dependências multivaloradas nesta entidade nem existem múltiplas chaves candidatas.

Entidade Agência

Agência

NomAgencia	Fundos	CidadeAgencia
Rio Branco	1.210.000,00	Rio de Janeiro
Icarai	1.500.000,00	Niterói
Leblon	2.500.000,00	Rio de Janeiro
Ipanema	150.000,00	Rio de Janeiro
Estacio	38.000,00	Rio de Janeiro

- ▶ $\text{Agencia} = \{\text{NOMAGENCIA}, \text{FUNDOS}, \text{CIDADEAGENCIA}\}$
- ▶ $\text{NOMAGENCIA} \rightarrow \text{FUNDOS}, \text{CIDADEAGENCIA}$

Entidade Empréstimos

Empréstimo

NomAgencia	NomCliente	NumEmprest	Valor
Rio Branco	Luis Sampaio	902230	1500,00
Rio Branco	Carlos Pereira	902230	1500,00
Rio Branco	Luis Paulo Souza	902240	1200,00
Ipanema	Jose Alves	902289	3000,00
Ipanema	Luis Paulo Souza	902255	850,00
Icarai	Carlos Pereira	902290	700,00
Icarai	Jose Alves	902212	400,00

- ▶ $\text{Emprestimo} = \{\text{NOMAGENCIA}, \text{NOMCLIENTE}, \text{NUMEMPREST}, \text{VALOR}\}$

Esta estrutura de dados, entretanto, não satisfaz a FNBC, pois se considerarmos NUMEMPREST somente como sua chave candidata, pode haver um par de linhas representando o mesmo empréstimo, desde que fossem duas as pessoas participantes dele, como na tabela apresentada.

NomAgencia	NomCliente	NumEmprest	Valor
Rio Branco	Luis Sampaio	902230	1500,00
Rio Branco	Carlos Pereira	902230	1500,00

Isso provocaria que o valor do empréstimo aparecesse repetidas vezes na tabela.

Logo,

- ▶ NUMEMPREST → valor não é total para a tabela, pois não podemos considerar o somatório das ocorrências de número de empréstimo.

Poderíamos considerar como chaves candidatas os seguintes conjuntos de atributos:

- ▶ NOMAGENCIA+ NUMEMPREST
- ▶ NOMCLIENTE+ NUMEMPREST

Da mesma forma, se analisarmos as dependências funcionais, veremos que não é válido o que segue:

- ▶ $(\text{NOMAGENCIA} + \text{NUMEMPREST}) \rightarrow \text{VALOR}$
- ▶ $\text{NOMCLIENTE} + \text{NUMEMPREST} \rightarrow \text{VALOR}$

Como existe dependência multivalorada neste caso, temos de aplicar a FNBC.

Decompomos a entidade que não está na FNBC em duas entidades sem perder a capacidade de junção delas.

- ▶ Empréstimo = {NOMAGENCIA, NUMEMPREST, VALOR}
- ▶ Devedor = {NOMCLIENTE, NUMEMPREST}

Empréstimo

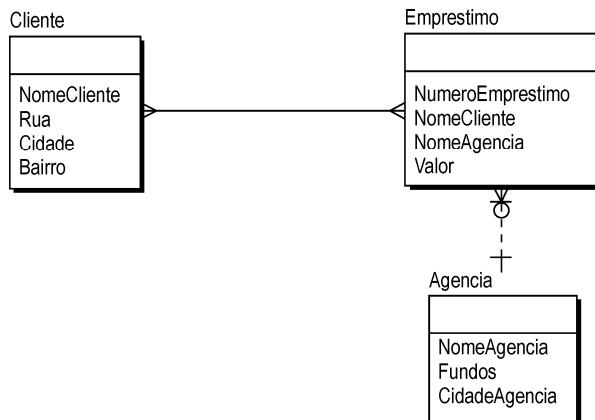
NomAgencia	NumEmprest	Valor
Rio Branco	902230	1500,00
Rio Branco	902240	1200,00
Ipanema	902289	3000,00
Ipanema	902255	850,00
Icarai	902299	700,00
Icarai	902212	400,00

Devedor

NomCliente	NumEmprest
Luis Sampaio	902230
Carlos Pereira	902230
Luis Paulo Souza	902240
Jose Alves	902289
Luis Paulo Souza	902255
Carlos Pereira	902299
Jose Alves	902212

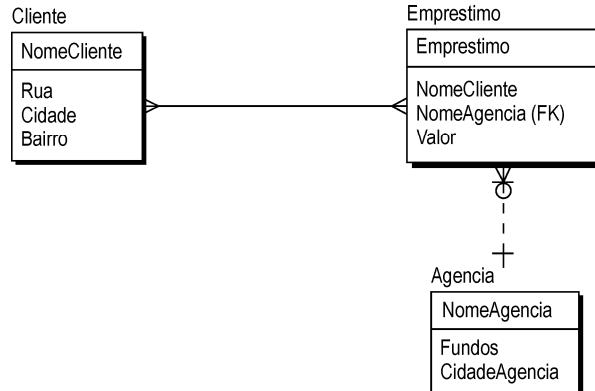
Olhando o resultado desta aplicação em um modelo de dados, vamos ver o modelo inicial e o modelo final após a aplicação da FNBC.

Modelo Inicial



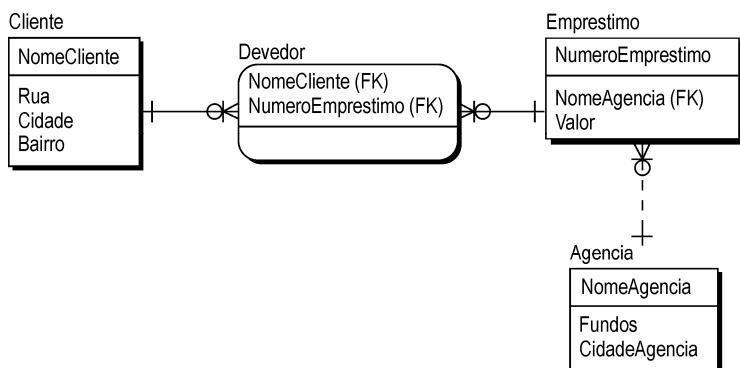
Identificando as chaves candidatas em Cliente e Agência, temos a figura seguinte.

Este é um modelo de dados que não satisfaz ainda a FNBC.



Aplicando a FNBC, teremos então um novo modelo refinado:

Modelo Final



Observe que o processo de modelagem natural de dados poderia começar este modelo muito próximo da realidade final com a FNBC. Tudo se origina na sua capacidade de abstração e de descrição das propriedades de um objeto.

Quando modelamos com foco lógico no modelo relacional, dificilmente temos situações como esta para resolver, pois é inherentemente a esse processo de modelagem a finalização encontrada, porém se o analista interpretar separadamente os objetos do ambiente, pode realmente criar a situação inicial para a aplicação da FNBC.

Quarta Forma Normal (4FN)

Na grande maioria dos casos, as entidades normalizadas até a 3FN são fáceis de entender, atualizar e 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. Essa nova dependência recebe o nome de multivalorada que existe somente se a entidade contiver no mínimo três atributos.

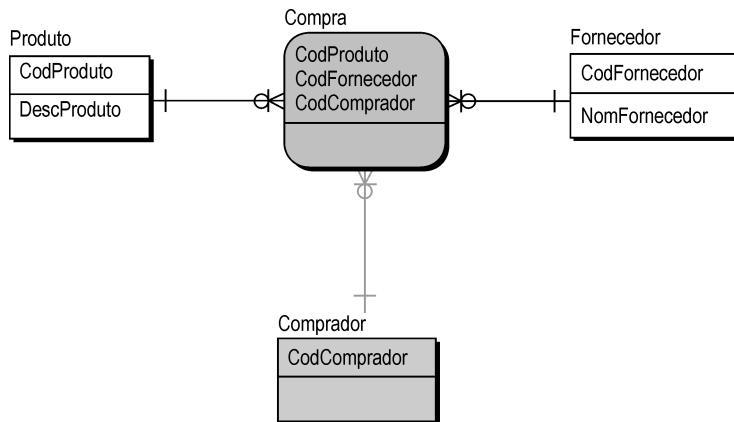
Uma entidade que esteja na 3FN também está na 4FN, se ela não contiver mais do que um fato multivalorado a respeito da entidade descrita. Essa dependência não é o mesmo que uma associação M:N entre atributos, geralmente descrita desta forma em algumas literaturas, mas ocorre quando consideramos a existência de relacionamentos, por exemplo, ternários.

Vamos imaginar o conteúdo desta tabela compra:

CodFornecedor	CodProduto	CodComprador
101	BA3	01
102	CJ10	05
110	88A	25
530	BA3	01
101	BA3	25

Como podemos observar, esta entidade tenta conter dois fatos multivalorados: os diversos produtos comprados e as diversas categorias. Com isso apresenta uma dependência multivalorada entre **CodFornecedor** e **CodProduto** e entre **CodFornecedor** e **CodComputador**.

Esta tabela está representada no modelo de dados em seguida:



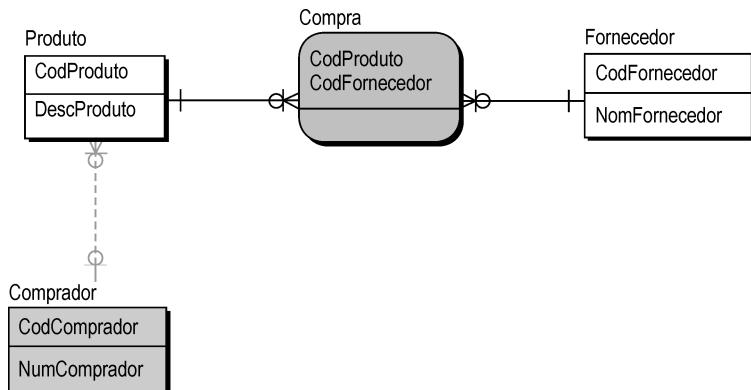
Embora esteja na 3FN, pois não existem dependências transitivas na tabela, ao conter mais de um fato multivalor, sua atualização torna-se muito difícil. Também existe a possibilidade de problemas relativos ao espaço físico de armazenamento virem a ocorrer, causados pela ocupação desnecessária de área de memória (primária ou secundária), podendo acarretar situações críticas em termos de necessidade de mais espaço para outras aplicações.

Para passarmos a entidade anterior para a 4FN, é necessária a realização de uma divisão da entidade original, em duas outras, ambas herdando a chave **CodFornecedor** e concatenada, em cada nova entidade, com os atributos **CodProduto** e **CodComputador**.

CodFornecedor	CodProduto
101	BA3
102	CJ10
110	88A
530	BA3
101	BA3

CodProduto	CodComprador
BA3	01
CJ10	05
88A	01
BA3	25

Observando o modelo de dados, podemos entender melhor:



É claro que incrementamos o modelo com mais atributos para ilustração, já que eles não atrapalham nada e permitem uma visualização de contexto maior.

A aplicação da 4FN foi realizada neste caso para corrigir um erro de modelagem de dados, pois no primeiro modelo utilizamos um relacionamento ternário de forma desnecessária, associando o comprador somente ao processo de compra, e não aos produtos que ele compra.

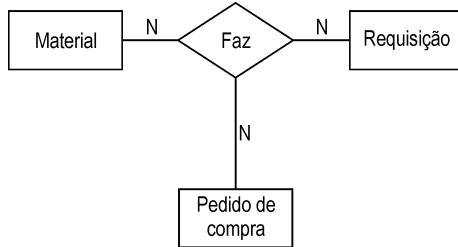
A conectividade apresentada é a mesma para os dois casos, um comprador compra muitos produtos, e um produto é comprado somente por um comprador. Entretanto, a anomalia iria permitir que esta regra de negócio fosse burlada pela simples estruturação com erro de 4FN no modelo inicial.

Um modelo de dados conceitual e lógico pode estar aparentemente correto até que se validem as possibilidades de dados no interior de suas tabelas relacionais.

Vamos usar como exemplo uma empresa que 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.

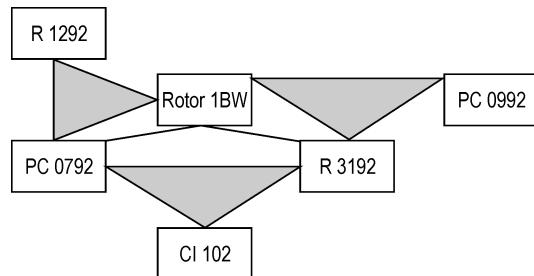
A modelagem deste exemplo mostra quais materiais de requisições geraram quais pedidos. Na figura é apresentado esse relacionamento ternário.



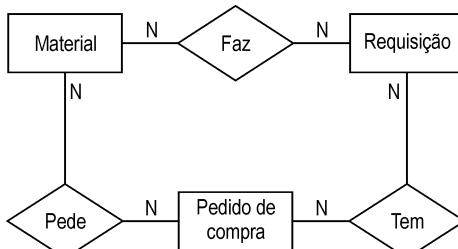
A tabela seguinte representa o relacionamento ternário faz que poderia conter os seguintes dados:

Material	Pedido de Compra	Requisição
ROTOR 1BW	PC 0792	R1292
ROTOR 1BW	PC0992	R3192
CI 102	PC0792	R3192
ROTOR 1BW	PC0792	R3192

Utilizando uma soma de visualização da dependência de junção, apresentada por James Bradley, obtemos o gráfico de dependência de junção, mostrado na figura seguinte:



Uma pergunta surge sobre este problema: é possível substituir o relacionamento ternário por relacionamentos binários, como os apresentados a seguir?



Como resposta podemos dizer que geralmente não é possível criar essa decomposição sem perda de informação, armazenada no relacionamento ternário.

Realizando uma projeção na tabela inicial, chegamos aos relacionamentos mostrados pelo modelo:

Relacionamento Pede

Material	Pedido de Compra
ROTOR 1BW	PC 0792
ROTOR 1BW	PC0992
CI 102	PC0792

Relacionamento Tem

Pedido de Compra	Requisição
PC 0792	R1292
PC0992	R3192
PC0792	R3192

Relacionamento Faz

Material	Requisição
ROTOR 1BW	R1292
ROTOR 1BW	R3192
CI 102	R3192

Se realizarmos agora um processo de junção destas três entidades, teremos:

Inicialmente, vamos juntar a tabela do relacionamento pede com a tabela do relacionamento tem por meio do campo pedido de compra. Obtemos então a tabela 4.

Tabela 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

Podemos observar que o registro apontado com sombreamento 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 tabela do relacionamento faz, através dos campos material e requisição. Após esta última operação de junção, obtemos a tabela 5, mostrada a seguir.

Tabela 5

Material	Pedido de Compra	Requisição
ROTOR 1BW	PC 0792	R1292
ROTOR 1BW	PC0992	R3192
CI 102	PC0792	R3192
ROTOR 1BW	PC0792	R3192

Como se pode notar, ao juntar as três tabelas, fruto da decomposição da tabela original, as suas informações foram preservadas. Isso significa que o relacionamento Faz inicial não está na 5FN, sendo necessário decompô-lo em relacionamentos binários, os quais estarão na 5FN.

Quinta Forma Normal (5FN)

A definição da 5FN diz que uma tabela de 4FN estará em 5FN quando seu conteúdo não puder ser reconstruído (existir perda de informação) a partir das diversas tabelas menores que não possuam a mesma chave primária. Essa 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.

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 entidade que conterá esses atributos, e que terá como chave primária o(s) atributo(s) do(s) qual(qualis) se tenha dependência parcial;
- ▶ Serão criadas tantas entidades quantos 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 esse atributo e cuja chave primária é o atributo que originou a dependência;
- ▶ Eliminar os atributos obtidos por cálculos realizados a partir de outros atributos.

ENTIDADES NA 3FN

Aplicação da FNBC

- ▶ Só aplicável em entidades que possuam chaves primárias e chaves candidatas concatenadas;
- ▶ Verificar se alguma chave candidata concatenada é um determinante, e em caso afirmativo, criar uma entidade com os que dependam funcionalmente desse determinante e cuja chave primária é o próprio determinante.

ENTIDADES NA FNBC

Aplicação da 4FN

- ▶ Para se normalizar em 4FN, a entidade precisa estar (obrigatoriamente) na 3FN;
- ▶ Verificar se a entidade possui atributos que não sejam participantes da chave primária e que sejam multivvalorados e independentes em relação a um mesmo valor da chave primária;
- ▶ Retirar esses atributos não chaves e multivvalorados, criando entidades 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 desse 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 esses relacionamentos. Sem essas redundâncias não haveria relacionamento entre entidades.

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 (*bottom-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-down*, no qual um modelo de dados é criado a partir da visualização da realidade, a normalização serve para realizar um aprimoramento desse 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 vamos observar o problema e não dar uma solução a ele. Neste caso, vamos projetar 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 consequência, vamos 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 ideias 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, pela 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.

Desnormalização dos Dados

Normalização é essencial para que bases de dados relacionais preservem a consistência da informação com respeito ao estado atual do sistema e torna o relacionamento entre entidades de dados (registros) implícito. Porém, requer navegação através de tabelas, que podem ser várias, para a composição de uma informação.

Alguns Motivos para a Desnormalização

Em contraste com os bancos de dados relacionais, um Data Warehouse diferencia-se, neste contexto, por manter dados em formato não normalizado, beneficiando-se da agilidade garantida em consultas. Além disso, a desnormalização faz com que o usuário realize consultas de forma transparente, isto é, ele não precisa saber o nome de tabelas de índices, ou tabelas intermediárias, resultantes da normalização. Desta forma, um Data Warehouse estabelece relacionamentos entre entidades de dados e possibilita consultas de forma eficiente.

A desnormalização é apenas tolerável devido a imperativos (rígidos) de desempenho como no caso de Data Warehouse, entre outros, ou se o sistema não consegue atingir um patamar mínimo de desempenho sem o processo de desnormalização.

Sempre devemos questionar se, após a desnormalização, o desempenho melhorará significativamente.

O processo de desnormalização vai retirar confiabilidade ou consistência da informação?

Importante documentar todos os atos de desnormalização e respectivas justificativas.

Outra justificativa para a desnormalização é a existência de várias consultas de desempenho crítico e que necessitam de informação de mais que duas tabelas (elevado número de junções) ou número elevado de chaves estrangeiras em uma tabela.

Também podemos usar a desnormalização se existir um número elevado de cálculos que devem ser efetuados sobre uma ou várias colunas antes da resposta a uma consulta.

UM ESTUDO DE CASO

Para que possamos desenvolver as técnicas que apresentamos até este momento no livro, vamos estudar um caso relativamente simples, partindo de uma análise de requisitos de sistemas. Construiremos uma lista de eventos, particionaremos os eventos para entendimento e detalhamento maior, e logo em seguida com maior abstração vamos desenhar um modelo de dados Entidade-Relacionamento.

É óbvio que é um caso fictício com algumas afirmações que são totalmente aderentes à administração hospitalar, mas é um exercício de análise de sistemas. Posteriormente você pode expandir os modelos resultantes para uma realidade mais próxima da sua.

O Problema - Administração de Cirurgias

Para este problema foi fornecido um conjunto de informações obtidas em uma reunião para definir o que o sistema deve controlar e realizar na visão dos usuários administradores.

Em um hospital existem diversas salas em seu centro cirúrgico. As salas de cirurgia possuem recursos para grupos de especialidades médicas, sendo apropriadas para cirurgias de uma especialidade.

Existem salas que se prestam somente a cirurgias de uma única especialidade, pois estão dotadas de equipamentos apropriados a essa especialidade médica.

Os médicos que realizam cirurgias no hospital em seus pacientes agendam as cirurgias conforme a disponibilidade das salas, informando datas, hora inicial e hora final prevista, junto à operadora do sistema.

Os horários para controle das salas são modulados de 30 em 30 minutos, de forma a manter-se uma quebra equilibrada da distribuição de horários. Quando uma cirurgia é encerrada, é informado ao controle do Centro Cirúrgico para a preparação e higienização para a próxima ocupação.

Uma sala de cirurgia não pode ser utilizada simultaneamente por mais de uma cirurgia, pois somente existe um leito cirúrgico em cada uma.

Um médico não pode ter cirurgias coincidentes em data e horário, mesmo que em salas diferentes.

Uma cirurgia somente deve ser realizada em uma sala adequada para a sua especialidade. Admitem-se urgências, porém devem ser destacadas para utilizarem salas que não são específicas da especialidade.

Urgência é considerada uma qualificação da cirurgia, mas não tem restrição de especialidade, podendo utilizar qualquer uma das salas.

Toda cirurgia tem somente um médico responsável, que devemos registrar para o controle do sistema.

Importante: Não constam deste escopo os assistentes, instrumentadores nem anestesistas.

Os medicamentos, materiais e remédios consumidos pela cirurgia devem ser computados para cobrança posterior.

O paciente é identificado pelo hospital, inclusive com o seu leito de internação e datas. Não existe no sistema nenhum controle pré-cirúrgico.

O sistema deve controlar cirurgias marcadas, assim como as já realizadas.

Para o Centro Cirúrgico especialidade é um dado do tipo: Cardiorrespiratória, Nefrologia, Ginecologia e Obstetrícia, Transplantes, Gastroenterologia, Oftalmologia, Traumatologia, Cirurgia Plástica, Oncologia etc.

Os médicos são todos registrados como habilitados no hospital, suas especialidades também controladas, podendo realizar cirurgias em somente uma especialidade.

São considerados materiais gastos em uma cirurgia as agulhas de sutura, algodões, sangue, gaze. Como medicamentos temos produtos como anestésicos em geral, antissépticos, soro etc., e remédios diversos com aplicação intravenosa.

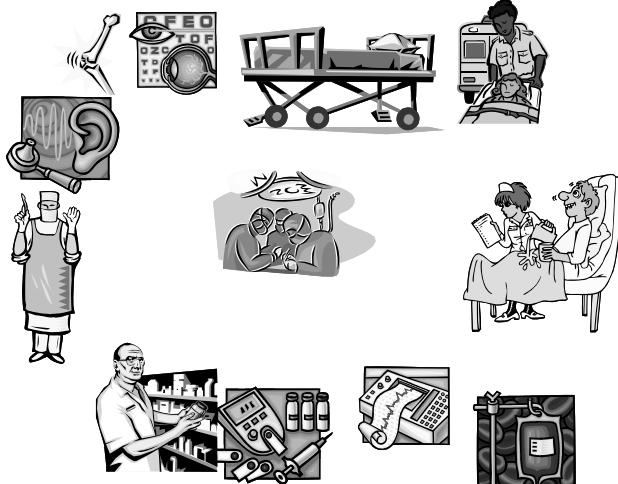


Modelagem

Vamos então começar a modelar este caso.

O que temos de coisa neste ambiente de negócios?

Utilizamos esta ilustração para despertar em você um olhar atento e estimular a sua capacidade de utilizar-se de abstração para identificar as coisas que estão neste ambiente.



Analisando, podemos concluir alguns objetos que temos neste ambiente de sistema.

A primeira coisa que aparece pode até causar dúvidas e permite duas interpretações. Uma como um fato e outra como uma entidade. A primeira interpretação apresenta-se como uma entidade.

A cirurgia.

Seguindo nossa abstração para a descoberta de entidades, pela ordem relativa temos dois objetos mais: médico e paciente.



Cirurgia



Paciente



Médico

É normal nos exercícios resolvidos em aula o aluno esquecer o paciente, mas nós já o movimentamos em nossos eventos e seria pouco provável esquecê-lo.

Temos então um conjunto de entidades até agora em número de três: médico, paciente e cirurgia.

É interessante discutir a segunda interpretação do objeto cirurgia com o leitor.

Cirurgia é uma entidade ou é um fato?

Tudo depende de sua interpretação do mundo real.

Se analisarmos que um médico realiza uma cirurgia, parece mais com uma entidade, pois médico relaciona-se com ela. Se analisarmos a realidade como médico opera paciente, este mesmo objeto passa a ser um relacionamento entre médico e paciente, um fato equivalente à entidade cirurgia.

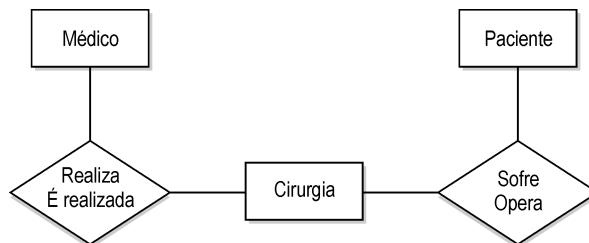
Na realidade temos duas soluções possíveis de modelo de dados conceitual para esta relação, ou o fato de acontecer uma cirurgia, ou do que é uma cirurgia.

No primeiro caso temos uma entidade cirurgia relacionada com a entidade Médico e com a entidade Paciente, pois os dois participam de cirurgia.

Resta-nos definir a conectividade entre estas entidades. Vamos lá!

Um médico pode realizar muitas cirurgias.

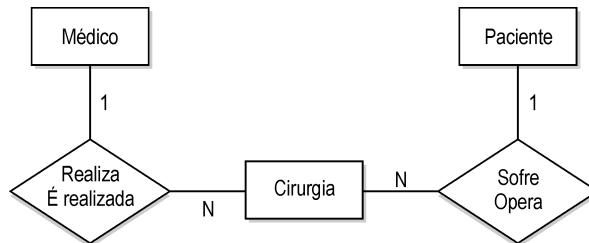
Uma cirurgia, segundo as premissas iniciais, tem somente um médico.



Uma cirurgia também tem somente um paciente, pois não são realizadas cirurgias coletivas nesse hospital. Já imaginou como seria?

Um paciente pode vir a sofrer muitas cirurgias. O médico pode esquecer uma tesoura na barriga do sujeito e ter de operar de novo para tirar.

Logo, podemos ter o Diagrama Entidade-Relacionamento (DER) com as conectividades corretas desta primeira interpretação.



Vamos agora exercitar e confirmar se a segunda interpretação está correta ou não.

Temos um relacionamento entre médico e paciente que é um fato, a própria cirurgia, o médico opera paciente.

Esse relacionamento deve ser entendido como a representação de um fato que pode ser denominado de cirurgia em si.



O que muda neste modelo em relação ao anterior?

A conectividade agora é diferente da anterior.

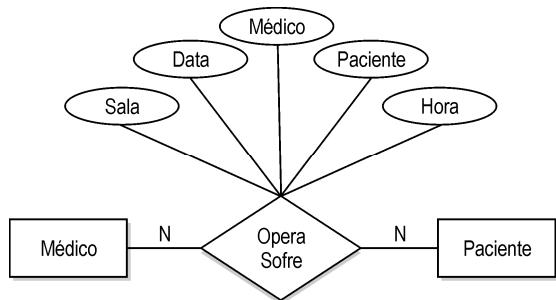
Um médico participa de opera muitos pacientes.

Um paciente sofre muitas cirurgias.

Logo, temos um relacionamento de muitos-para-muitos.



Vamos apresentar este relacionamento com atributos, pois nele estarão as informações de hora, data, especialidade da cirurgia etc.



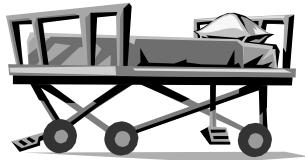
Veja que os atributos são os mesmos que estariam na entidade cirurgia da solução anterior, com três entidades.

E nos atributos aparece a localização da cirurgia, a sala.

Voltando à abstração em busca de entidades, pergunta-se:

Sala não é um objeto, uma coisa desse ambiente que possui dados, e é objeto do sistema?

A resposta evidente é positiva. Sala é uma entidade, pois pode estar localizada em uma ala, em um andar, tem uma especialidade que pode atender etc.



Vamos acrescentar ao modelo a entidade sala.

Mas já analisando, sala se relaciona com quem?

Sala se relaciona com médico?

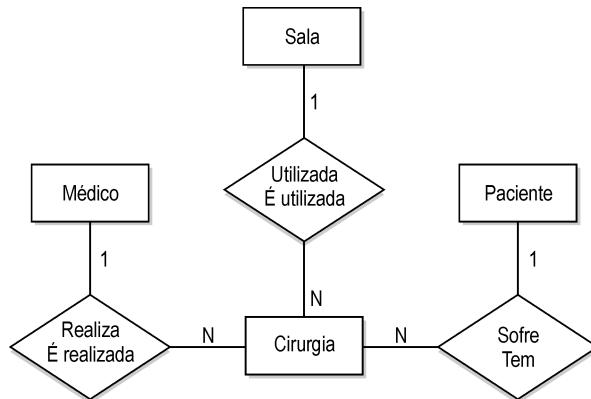
Somente através da cirurgia ou do relacionamento opera.

Sala se relaciona com paciente?

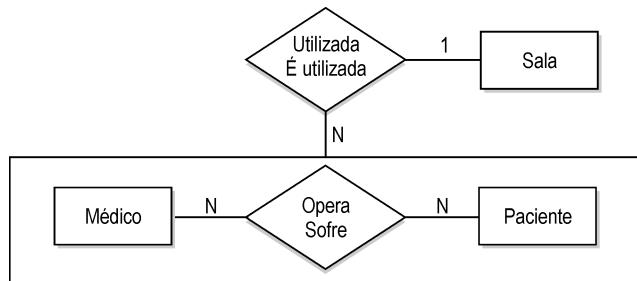
Não, somente por meio da cirurgia também.

Mas afinal, sala se relaciona com quem?

Sala se relaciona com cirurgia em nossa primeira solução.



Ou com a agregação do relacionamento opera da seguinte solução:



São duas soluções para retratar uma realidade, e as duas estão corretas.

Vamos seguir para verificar se existem mais entidades nesse ambiente de sistema.

Existe uma entidade a mais que é mencionada em várias afirmativas no levantamento de ambiente:

"As salas de cirurgia possuem recursos para grupos de especialidades médicas, sendo apropriadas para cirurgias de uma especialidade."

"Existem salas que se prestam somente a cirurgias de uma única especialidade, pois estão dotadas de equipamentos apropriados a essa especialidade médica."

"Os médicos são todos registrados como habilitados no hospital, sendo suas especialidades também controladas, podendo realizar cirurgias em somente uma especialidade."



Isso nos leva à existência da entidade especialidade.

Mas especialidade é efetivamente uma entidade ou é um atributo de sala e de médico?

No mundo real especialidade é um dado qualificativo de sala e de médico, pois complementa a descrição destes dois objetos.

Entretanto, como estamos pensando e raciocinando em termos de banco de dados relacional, podemos realizar uma análise de redundâncias do modelo de dados.

O que seria a análise?

Se especialidade é um atributo comum a duas entidades, podemos otimizar essa informação tabulando-a na forma de um objeto externo a estas duas entidades. Desta forma certamente teríamos informação unificada tanto para a qualificação de salas quanto para a qualificação de médicos.

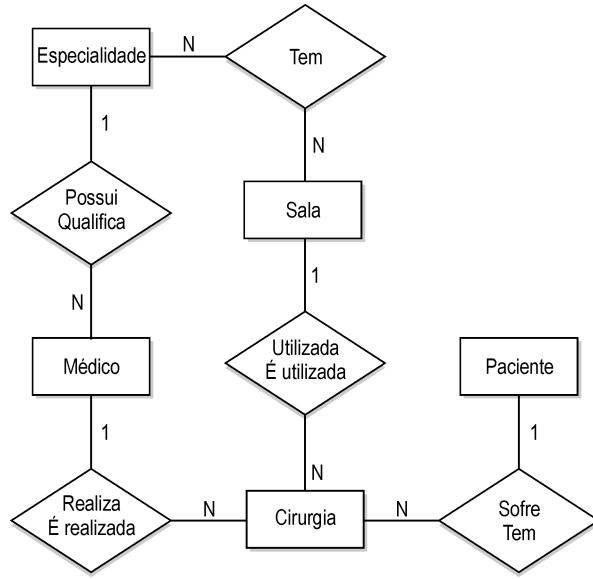
Como fazemos isso?

Criamos a entidade especialidade, que se relaciona com a entidade sala e com a entidade médico.

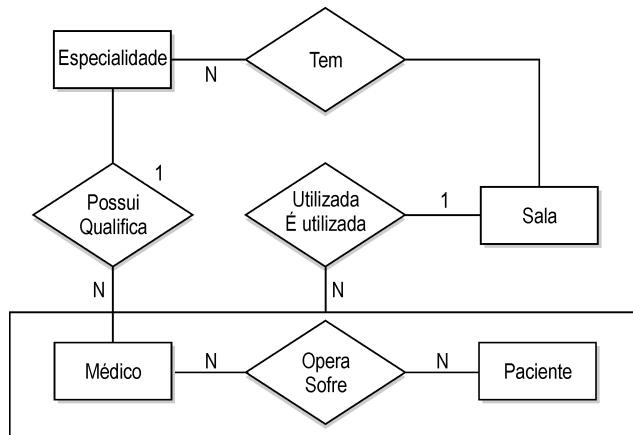
Com isso obtemos duas conectividades distintas em cada relacionamento:

- ▶ Um médico possui uma especialidade
- ▶ Uma especialidade qualifica muitos médicos
- ▶ Uma sala tem muitas especialidades
- ▶ Uma especialidade tem muitas salas

Com esta solução estamos com mais um relacionamento muitos-para-muitos no modelo.



No segundo formato de solução teríamos o mesmo formato de relacionamento, somente com o detalhe de observar que a ligação de especialidade com médico é indicada no diagrama ER com uma linha que conecta com a entidade médico e não com a caixa da agregação.



Seguindo nosso raciocínio abstrato, o que mais teríamos de significativo neste modelo de dados, com base no levantamento de informações inicial?

Temos mais duas afirmativas para analisar:

"Os medicamentos, materiais e remédios consumidos pela cirurgia devem ser computados para cobrança posterior."

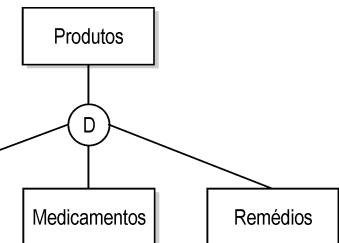
"São considerados materiais gastos em uma cirurgia as agulhas de sutura, algodão, sangue, gaze. Como medicamentos temos produtos como anestésicos em geral, antissépticos, soro etc., e remédios diversos com aplicação intravenosa."

Cuidado para não criar de imediato três entidades, pois temos um grupo de objetos que são consumidos por uma cirurgia.

Voltando aos conceitos de abstração, vamos lembrar que existe o conceito de classificação de objetos.

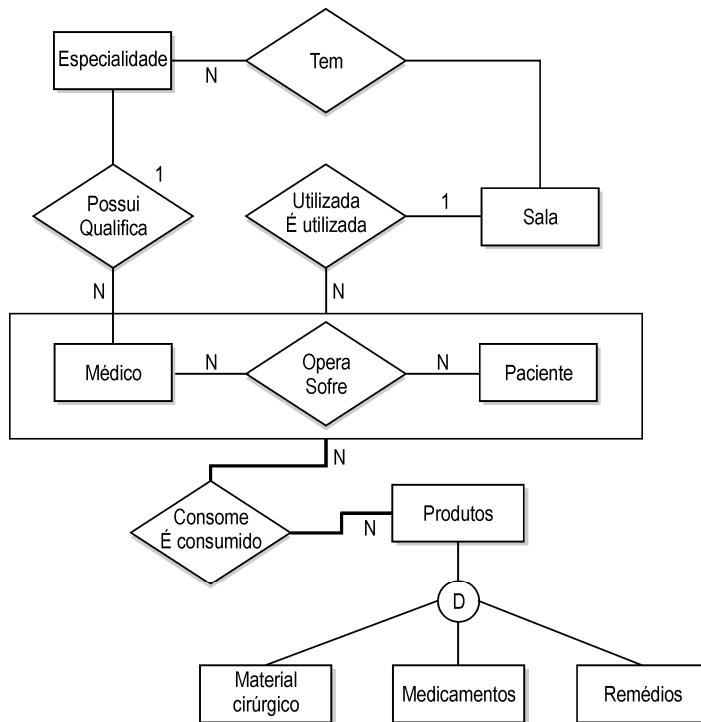
Há na realidade um conjunto de coisas que são consumidas na cirurgia, uma classe de objeto, que é composta de medicamentos, materiais e remédios por assim dizer.

Logo, podemos utilizar o conceito de generalização de entidades neste modelo para realizar o diagrama ER.



Este bloco de modelo deve estar relacionado com que entidade?

Bem, se os produtos são consumidos, utilizados na cirurgia, devem estar relacionados obviamente com cirurgia.



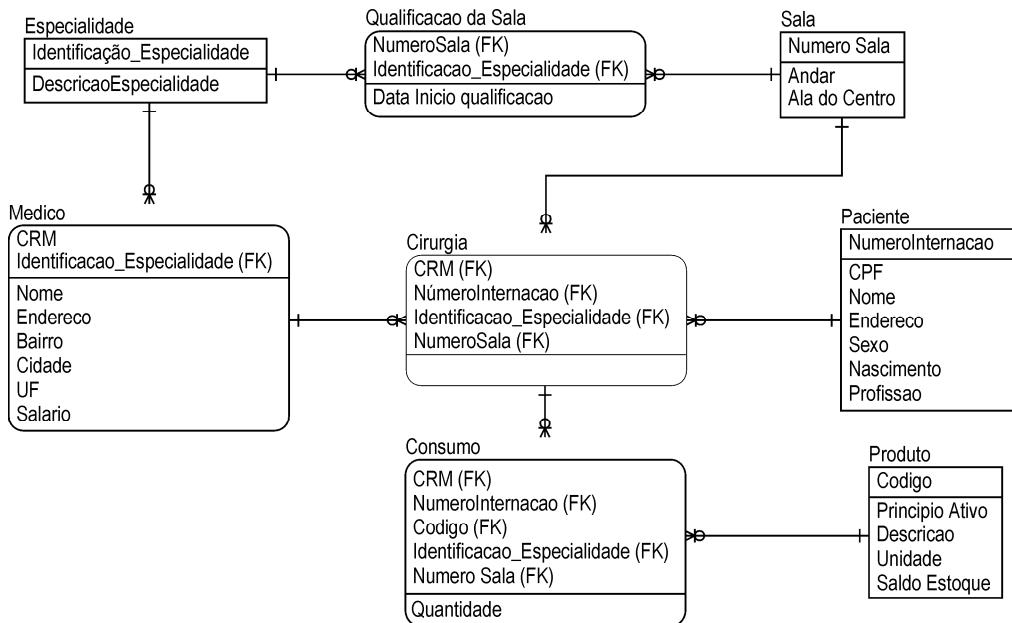
Um produto é consumido em muitas cirurgias e uma cirurgia consome muitos produtos.

Logo, há mais um relacionamento muitos-para-muitos para encerrar o nosso modelo.

Com base nas informações obtidas em nosso levantamento de dados, conseguimos determinar em um primeiro momento um conjunto de eventos e um modelo conceitual de dados para o sistema.

Vamos analisar os atributos desse modelo de dados até agora construído, utilizando uma ferramenta CASE que permita destacar as chaves primárias e estrangeiras do modelo, o ERwin.

Vamos elencar os possíveis atributos das nossas entidades e relacionamentos, assim como os obrigatórios para a efetivação dos relacionamentos.



Neste modelo apresentado, propositalmente deixamos de colocar informações relativas à data e hora da cirurgia.

Queremos provocar a discussão de onde devem ser colocados esses atributos.

Vamos exercitar então.

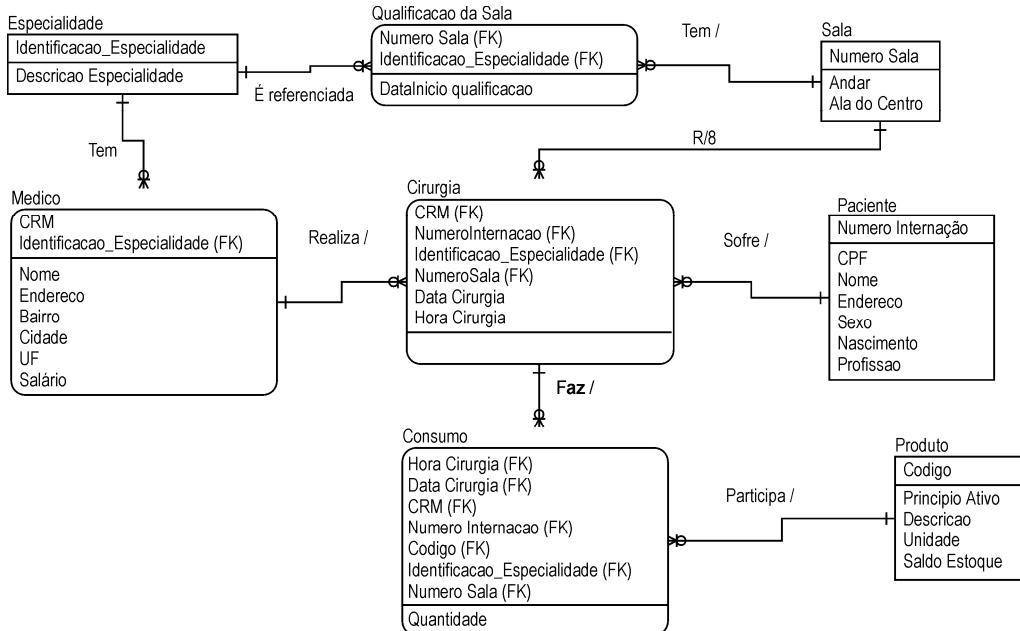
Se colocarmos somente a data na chave primária da entidade cirurgia, efetuamos uma restrição no modelo: um médico somente pode operar um paciente uma vez em um dia.

Caso o paciente tenha de voltar para uma nova cirurgia com o mesmo médico na mesma data, isso será impossível de ser registrado, pela restrição de identidade em uma tabela relacional. Não existe mais de uma ocorrência na tabela com a mesma chave primária.

Logo, temos de colocar os dois dados na chave primária para que se evite essa restrição no mundo real.

Assim, a chave primária da entidade cirurgia terá mais dois dados:

- ▶ Data da cirurgia
- ▶ Hora inicial da cirurgia



A informação de hora final estimada não necessita ficar na chave primária, até por ser um dado estimado, assim como não é relevante na identificação da coisa cirurgia.

Observe que a entidade cirurgia ficou composta somente de chave primária. Fica a cargo de sua criatividade completar essa entidade com outros atributos que, certamente, existem no mundo real.

Basta irmos até os usuários e questioná-los agora sobre que dados mais existem e são controlados em cirurgia.

Encerramos aqui este estudo de caso, no qual já conseguimos aplicar os princípios que estudamos em modelo de dados ER.

HIERARQUIAS

Tratamento de Hierarquias de Dados

As estruturas hierárquicas são aparentemente mais complexas de serem resolvidas, com Diagramas Entidade-Relacionamento e em ambientes relacionais, ou seja, bancos de dados relacionais.

Tudo é uma questão de entendimento e raciocínio por partes.

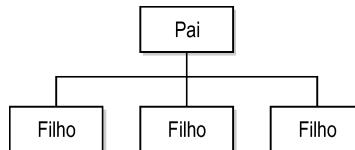
Então, seguindo a filosofia de Jack, o estripador, vamos por partes.

No nosso exemplo até que é bem simples:

Cabeça é composta de olho, ocelo, armadura bucal, antena e área intersutural.

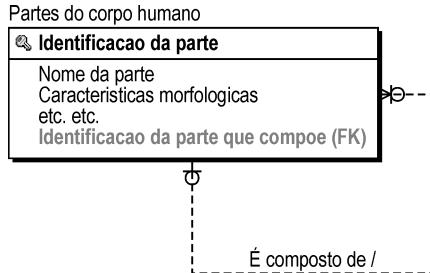
Essa estrutura de informações possui somente um nível de decomposição.

É, então, uma estrutura hierárquica de um nível como o diagrama apresentado a seguir:



O importante é conseguir visualizar uma tabela de dados e seu conteúdo para entender o modelo de dados que será apresentado como solução.

Vejamos que o conjunto de dados ou o objeto que se deseja modelar são, digamos, partes do corpo humano.



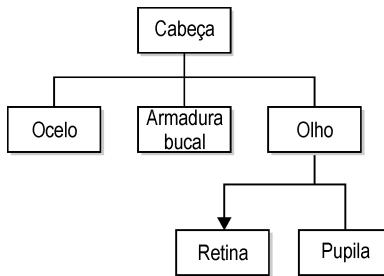
"Partes do corpo humano são compostas por outras partes do corpo humano."

Logo, temos um autorrelacionamento ou relacionamento reflexivo em uma mesma entidade, que denominamos de partes do corpo humano.

O modelo de dados fica simples de entender se analisarmos a tabela com que implementamos esse modelo de dados ER.

Identificação da Parte do Corpo	Descrição	Características	Parte que Compõe	etc. etc.
1	Cabeça	xxxx	nulo	kjgjgfj
2	Ocelo	yyyy	1	oiuytoru
3	Armadura Bucal	zzzz	1	b cbvu
4	Área intersutural	dddd	1	kmnflkj
5	Olho	fffff	1	mn fke
6	Pupila	vvvv	5	ldoirr

Observe que foi acrescentado um nível à estrutura hierárquica:



Cabeça também é composta por olho, que é composto por pupila, retina etc.

A navegação nesta tabela buscando os registros inter-relacionados fornece a árvore de composição de qualquer objeto modelado desta forma.

Identificação da Parte do Corpo	Descrição		Parte que Compõe		Identificação da Parte do Corpo	Descrição		Parte que Compõe	
1	Cabeça		nulo		2	Ocelo		1	
1	Cabeça		nulo		3	Armadura Bucal		1	
1	Cabeça		nulo		4	Área intersutural		1	
1	Cabeça		nulo		5	Olho		1	
5	Olho		1		6	Pupila		5	

Neste caso observamos muito bem a importância do conceito de chave estrangeira, pois com ela conseguimos realizar a comparação de valores e identificar quem se relaciona com quem.

O resultado de um SELECT, para encontrar a árvore de composição, fornece a tabela apresentada anteriormente.

Mesmo que existam vários níveis de hierarquia, sempre utilizaremos esse tipo de autorrelacionamento.

Somente vamos mudar o modelo quando tivermos composições em que uma determinada parte entra na composição de uma outra parte. Então teremos autorrelacionamentos de muitos-para-muitos.

No exemplo tratamos a composição como um relacionamento de um-para-muitos, pois estamos discutindo a implementação de uma hierarquia em um modelo de dados ER.

MODELO FÍSICO

O Modelo Físico em Si

Nesta etapa, o analista passa as visões do modelo conceitual para o modelo lógico relacional, no qual os dados são vistos como estruturas de dados voltadas para as características da abordagem escolhida, objetivando a implementação do banco de dados.

O modelo físico provê um contexto para definição e registro no catálogo do banco de dados dos elementos de dados que formam um database, por assim dizer, e fornece aos analistas de uma aplicação a possibilidade de escolha dos caminhos e acesso às estruturas de dados.

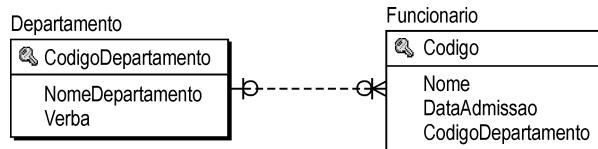
Na criação de um modelo físico de dados, realizada pela conversão do modelo conceitual, possuímos uma lista de conceitos dos componentes de cada modelo e sua equivalência, que pode orientar na relação dos termos utilizados que vamos apresentar neste capítulo, além de considerações gerais sobre o processo.

A transformação de um modelo conceitual ou lógico em modelo físico exige que sejam realizadas as definições dos atributos de uma tabela (colunas) que serão índices, como será o tipo desse atributo (numérico, caractere, data, outros), assim como a exigência de sua existência ou não, sua identificação como chave primária ou estrangeira na tabela em si.

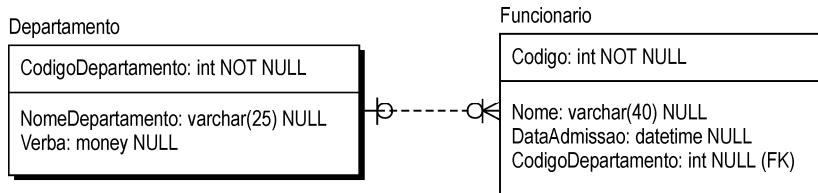
Na realidade, vamos adequar o tipo de dado ao tipo de dado permitido, implementado pelo Sistema Gerenciador de Banco de Dados Relacional com que vamos trabalhar.

As principais ferramentas CASE do mercado possuem interfaces de conversão do modelo lógico no modelo físico, que disponibilizam o tipo de dado correto e permitido para cada SGBD usualmente utilizado, além de permitirem assinalamento das outras características de um modelo físico necessário.

Representação Lógica de um Modelo de Dados



Representação Física de um Modelo de Dados



Propriedades de uma Coluna

Quando convertemos um modelo lógico em modelo físico, cada um dos atributos existentes nas entidades do modelo deve possuir agora um conjunto de propriedades relativas a uma coluna de uma tabela.

Vamos ver quais são as regras e o que devemos cuidar para a criação do modelo físico de dados.

As propriedades que devem ser seguidas neste processo de conversão de modelo lógico em modelo físico dizem respeito, em sua grande parte, a informações de implementação sobre cada coluna da tabela a ser criada.

O nome da coluna será restrinido pelas convenções e regras de metodologia utilizada ou da administração de dados da empresa, bem como pelos requisitos semânticos do SGBD.

O tipo de dado será limitado àqueles suportados pelo SGBD. Vale ressaltar que cada SGBD possui características próprias que determinam os tipos de dados aceitos por ele. Entretanto, vamos apresentar os principais tipos de dados e o seu significado para que o leitor familiarize-se com eles.

São tipos de dados que cada campo de uma tabela deve possuir.

Toda coluna deve ter preestabelecido o tipo de dado que pode conter. Por exemplo, se um campo de uma tabela for do tipo numérico, ele não pode receber dados que forem letras.

A seguir apresentamos uma tabela com os tipos de dados mais usuais.

Tipos de Dados Mais Comuns (Padrão ANSI)

<i>Smallint</i>	Armazena valores numéricos, em dois bytes binários, entre o intervalo -32768 e +32767.
<i>Integer</i>	Armazena valores numéricos, em quatro bytes binários, entre o intervalo -2147483648 e +2147483647.
<i>Float</i>	Armazena valores numéricos com ponto flutuante, em oito bytes binários, entre o intervalo -1.79769313486232E308 e -4.94065645841247E-324 para valores negativos, 4.94065645841247E-324 e 1.79769313486232E308 para valores positivos.
<i>Decimal(n,m)</i>	Armazena valores numéricos com no máximo 15 dígitos. Nessa opção deve ser definida a quantidade de dígitos inteiros (n) e de casas decimais (m) existentes no campo.
<i>Varchar (n)</i>	Definir um campo alfanumérico de até n caracteres, em que n deve ser menor ou igual a 254 caracteres.
<i>Char (n)</i>	Definir um campo alfanumérico de n caracteres, em que n deve ser menor ou igual a 254 caracteres.
<i>Long Varchar</i>	Definir um campo alfanumérico de comprimento maior que 254 caracteres.
<i>Date</i>	Definir um campo que vai armazenar datas.
<i>Time</i>	Definir um campo que vai armazenar horas.

Em seguida temos de definir se a existência de valor de dados é obrigatória para a coluna ou não.

A Opção de Nulo

Por padrão, as chaves primárias são definidas como *not null*, ou seja, não admitem a inexistência de valor em qualquer uma das colunas que a compõem.

Uma Regra de Validação

Se existir a regra de validação, ela deve conter uma lista de valores válidos para a coluna que você define. As regras de validação podem ser intervalos de valores ou um código em SQL.

Uma regra de validação é uma expressão que estabelece o intervalo de valores aceitáveis que podem ser armazenados em uma coluna.

Observe que a utilização e a implementação no modelo físico de tabelas de domínio para colunas de uma tabela são uma alternativa às regras de validação em alguns casos.

Se a lista de valores válidos for volátil ou longa, pode ser preferível usar, em sua substituição, uma tabela de domínio, visto que uma tabela de domínio será mais fácil de ser atualizada no ambiente de produção.

As tabelas de domínio podem e devem ser criadas somente no modelo físico, pois são objetos apenas físicos, uma vez que elas não têm base nem existem no modelo conceitual ou no modelo lógico.

Se a lista de valores for estável, o uso da regra de validação pode ser a melhor opção. Isso significa uma tabela a menos no banco de dados para cada regra usada.

Exemplo de regra de negócios

Nota_Fiscal
NumNota: int NOT NULL
ClienteNota: varchar(35) NULL
DataNota: datetime NULL
NumItensNota: int NULL

Regra de negócios: o número de itens por nota fiscal está sempre entre 5 e 15.

NumItensNota >= 5 and NumItensNota <= 15

Outro exemplo de regra de validação

Regra de validação StatusNotaFiscal:

- ▶ F = 'Faturada'
- ▶ C = 'Cancelado'

Por exemplo, se a empresa não permitisse cancelar uma nota com data de três dias antes do dia da própria nota.

Neste caso, teríamos a regra de validação:

NOT StatusNotaFiscal="C"(date < (DataNota+3) AND StatusNotaFiscal = "F")

Valor Padrão

Este é o valor colocado na coluna durante uma inserção de registro na tabela na ausência de qualquer outro valor para aquela coluna. Conhecido como valor default do atributo.

Visões de Dados

Uma visão do banco de dados (Database View) é uma apresentação personalizada dos dados armazenados em uma ou mais tabelas. As visões também são conhecidas como tabelas virtuais ou consultas armazenadas.

Uma visão pode:

- ▶ Incluir um subconjunto de colunas de uma tabela do banco.
- ▶ Incluir colunas de múltiplas tabelas do banco.
- ▶ Ser baseada em outras visões em vez das tabelas do banco.
- ▶ Ser consultada, atualizada, inserida e apagada. Todas estas ações afetam os dados armazenados nas tabelas do banco, com algumas restrições, como veremos no capítulo sobre SQL.
- ▶ Pode ser frequentemente alterada para se adequar às necessidades de alteração sem exigir uma mudança para a tabela do banco subjacente.

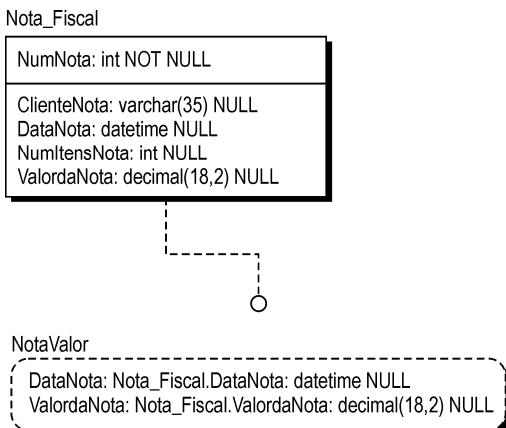
É importante lembrar-se de que as visões exibem dados das tabelas do banco, mas não armazem dados.

Visto o conceito, podemos ter no modelo físico de dados a inserção de visões de dados no sentido de criarmos limitações ou restrições ao acesso amplo da aplicação a todos os dados de uma tabela, ou na análise de volumes de dados criar visões de subtipos para acesso distinto de usuários.

As visões podem ser usadas para fornecer um nível adicional de segurança de tabelas, ocultar a complexidade de determinados dados, simplificar os comandos para usuários, fornecer diferentes apresentações de dados ou armazenar consultas complexas, evitando que se consumam recursos de processamento elevados desnecessariamente.

As visões de dados somente existem no modelo físico, e as suas colunas podem ser: colunas de tabela, colunas de outras visões ou expressões em SQL.

Exemplo de Visão



Índices do Banco de Dados

Um índice é uma estrutura associada a uma tabela que torna a pesquisa mais rápida.

Deve-se levar em conta o emprego de índices em qualquer coluna frequentemente utilizada na pesquisa de uma tabela, pois, com essa opção, a duração das pesquisas será consideravelmente diminuída.

Os índices de livro são uma boa analogia; por exemplo, os divisores tabulados representam um índice agrupado e os outros índices estão dispostos no final do livro.

Entretanto, pode haver uma diminuição de rendimento com a utilização de índices. À medida que mais índices são criados, o banco de dados realiza mais lentamente atualizações. Por exemplo, cada vez que uma inserção é realizada em uma coluna indexada, cada índice precisa ser atualizado.

Tecnicamente não há limite ao número de índices, mas quanto maior o número deles mais difícil fica a sua manutenção.

Outro aspecto a considerar para inserção de índices em uma tabela é a sua utilização nas consultas mais frequentes, pois conforme a disposição e execução dessa consulta, principalmente se estivermos utilizando múltiplas junções, esses índices não serão totalmente utilizados quando da sua execução.

Vamos comentar mais sobre este aspecto na abordagem do SQL e sua decomposição algébrica de comandos.

A escolha de indexação de colunas depende também do negócio para o qual o BD foi modelado, sendo preciso considerar:

- ▶ As atualizações ou consultas são muito críticas?
- ▶ Qual é o volume de dados previsto para a tabela?
- ▶ Quão volátil será essa tabela?

Para tabelas de um Data Warehouse, por exemplo, quase tudo é indexado porque os requisitos do negócio são consultas rápidas e as tabelas possuem baixa volatilidade.

Chaves Substitutas

É comum que a chave primária de uma tabela seja substituta, ou seja, é criada uma coluna que contém um número ou código de identificação que é um identificador único, mas que não tem um significado intrínseco no que se refere ao objeto sendo modelado.

Uma chave substituta é artificial e será usada como uma substituta para uma chave natural, derivada dos relacionamentos da tabela.

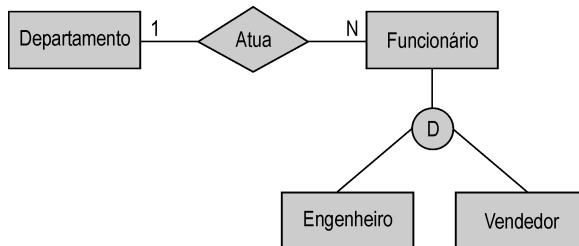
A inserção de chaves substitutas no modelo físico é uma realidade, mas deve ser feita de forma criteriosa, pois implica na existência de processos de consistência das colunas que originalmente compunham a chave primária da tabela, e que foram definidas no modelo lógico.

As Generalizações

O tratamento que deve ser dado à conversão de generalizações no modelo físico envolve o seu mapeamento e clara definição de tabelas para os subtipos, quando eles efetivamente possuírem conjuntos distintos de atributos.

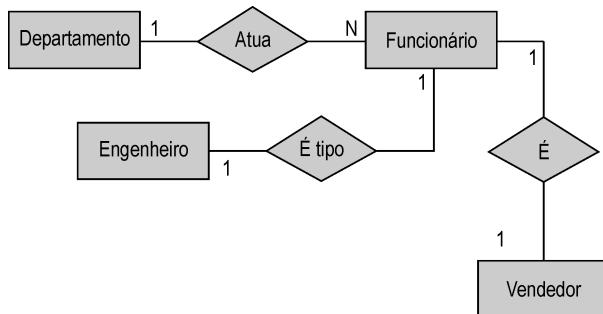
Exemplo

Dada a entidade funcionário, existe uma variação de atributos para ela, pois há um ponto básico para os funcionários que são engenheiros: eles têm informações adicionais, que são dados adicionais no subtipo 'engenheiro', e para o subtipo 'vendedor' existem outros dados que são diferentes dos dados adicionais de engenheiro.



Como não é possível ter colunas com tamanho e tipo variável, temos de criar tabelas para os casos em que as informações variam.

A tabela funcionário só pode ter um e somente um conjunto de colunas.



As informações dos engenheiros serão completadas pela tabela engenheiro.

Se não tivesse atributo diferente para os outros, todos os que não fossem engenheiros ou vendedores só teriam seus dados na tabela funcionário.

Não é possível um empregado cuja função é engenheiro atuar como técnico, porque os elementos não podem se sobrepor.

Os subtipos tornam-se tabelas carregando o identificador do conjunto a que pertencem.

Tabelas do Exemplo

Departamento

CodigoDept	Nome Departamento
1	Engenharia
2	Vendas
3	Administração

Funcionário

Matricula	Nome	Funcao	CodigoDept
101	Luis Sampaio	Eng	1
104	Carlos Pereira	Vend	2
134	Jose Alves	Vend	2
121	Luis Paulo Souza	Sec	3
123	Pedro Sergio Doto	Vend	1

Engenheiro

Matricula	Ajuda de Custo	Especialidade
101	1.500,00	Obras Civis

Vendedor

Matricula	Hora Extra	Despesa Extra	Placa carro
104	5,00	255,00	LVW 5289
134	3,50	300,00	LXT 5289
123	2,50	150,00	LOP 5289

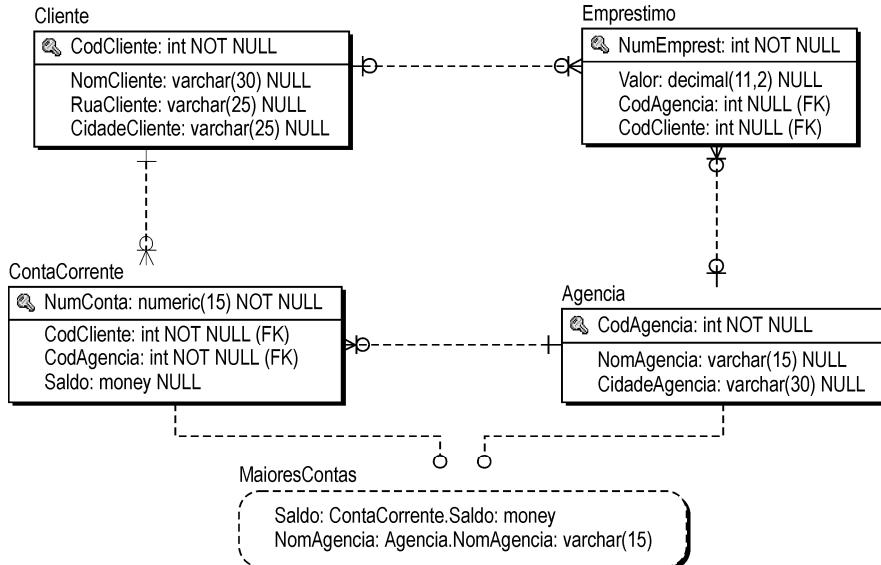
Secretaria

Matricula	Lingua Estrang	Curso
121	Inglês	Computação

Colocamos secretária neste exemplo para ampliar a sua visão da implementação dos subtipos no modelo físico.

A entidade funcionário vira uma tabela (regra padrão) e os subtipos são transformados em outras tabelas, carregando a chave primária matrícula.

Efetivado o modelo físico de dados, estamos aptos a criar o banco de dados projetado pela geração de um script de comandos SQL para o SGBD.



```

CREATE TABLE Cliente (
    CodCliente          int NOT NULL,
    NomCliente          varchar(30) NULL,
    RuaCliente          varchar(25) NULL,
    CidadeCliente       varchar(25) NULL,
    PRIMARY KEY (CodCliente ASC) )
CREATE TABLE Agencia (
    CodAgencia          int NOT NULL,
    NomAgencia          varchar(15) NULL,
    CidadeAgencia       varchar(30) NULL,
    PRIMARY KEY (CodAgencia ASC) )
CREATE TABLE ContaCorrente (
    NumConta            numeric(15) NOT NULL,
    CodCliente          int NOT NULL,
    CodAgencia          int NOT NULL,
    Saldo               money NULL,
    PRIMARY KEY (NumConta ASC),
    FOREIGN KEY (CodAgencia)
        REFERENCES Agencia (CodAgencia),
    FOREIGN KEY (CodCliente)
        REFERENCES Cliente (CodCliente) )
CREATE TABLE Emprestimo (
    NumEmprest          int NOT NULL,
    Valor                decimal(11,2) NULL,
    CodAgencia          int NULL,
    CodCliente          int NULL,
    PRIMARY KEY (NumEmprest ASC),
    FOREIGN KEY (CodCliente)
        REFERENCES Cliente (CodCliente),
    FOREIGN KEY (CodAgencia)
        REFERENCES Agencia (CodAgencia) )
CREATE VIEW MaioresContas AS
    SELECT ContaCorrente.Saldo, Agencia.NomAgencia
    FROM ContaCorrente, Agencia
    WHERE Saldo > 100000

```

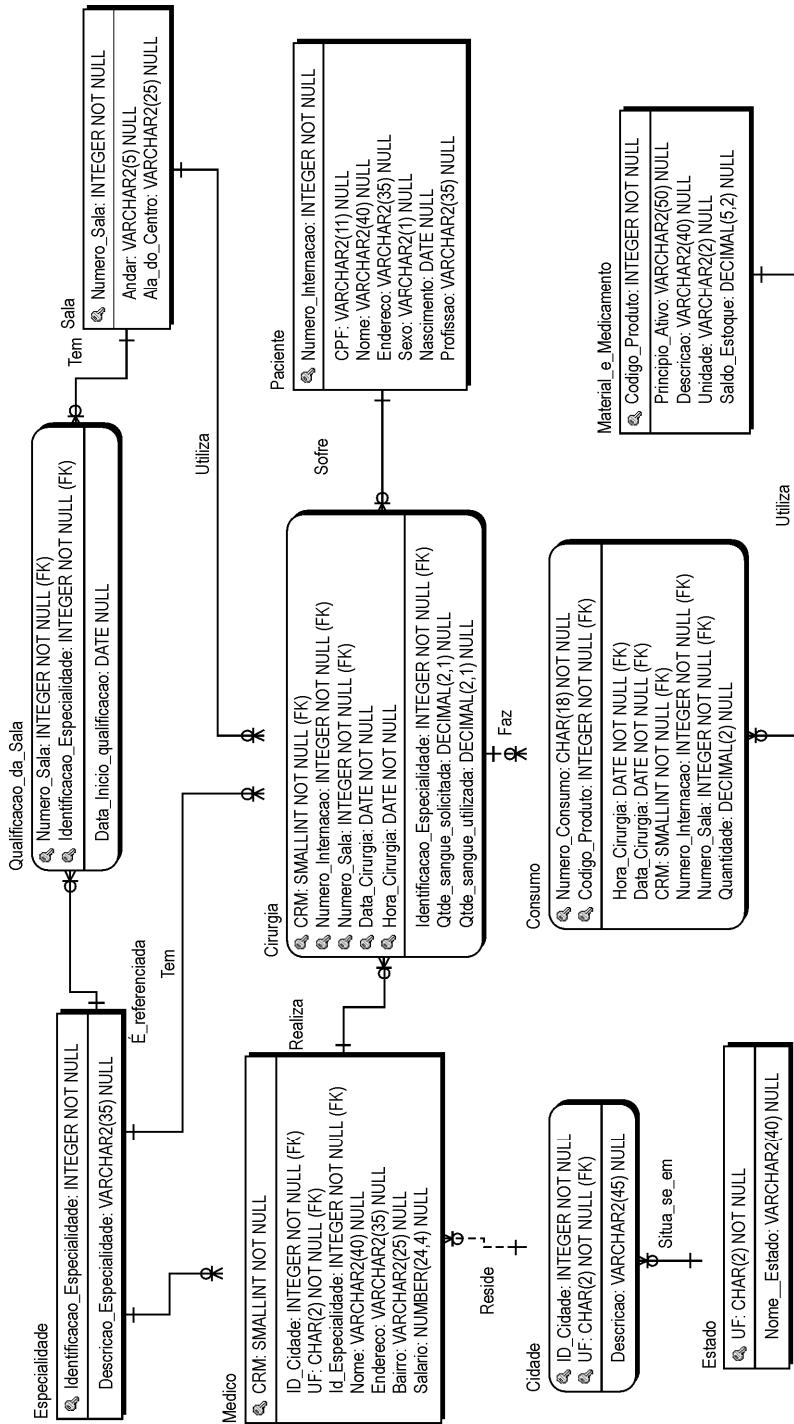
Relação entre Modelo Lógico e Modelo Físico

Esta tabela apresenta as correspondências entre o modelo lógico e o modelo físico.

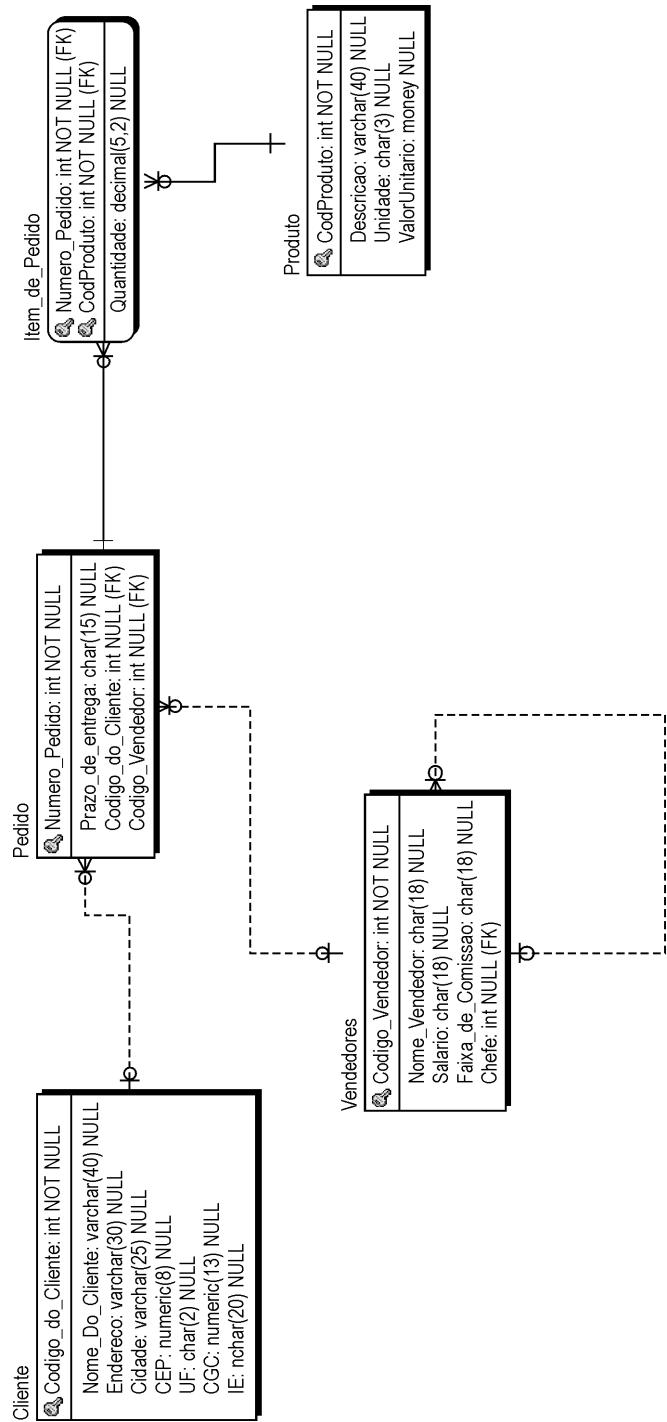
Modelo Lógico	Modelo Físico
Entidade	Tabela
Entidade dependente	A chave estrangeira faz parte da chave primária
Entidade independente	A chave estrangeira não faz parte de sua chave primária
Atributo	Coluna
Tipo de dado (texto, numérico, alfanumérico etc.)	Restrito aos tipos de dados físicos implementados no SGBD a ser utilizado
Chave primária	Coluna índice único de chave primária
Chave estrangeira	Coluna índice não único de chave estrangeira
Atributo de busca não chave	Coluna índice não único
Regra de validação	Restrição de integridade
Regra de negócio	Domínio, ou regra de validação, ou stored procedure ou trigger

Observe em seguida alguns modelos físicos (inclusive o do nosso estudo de caso).

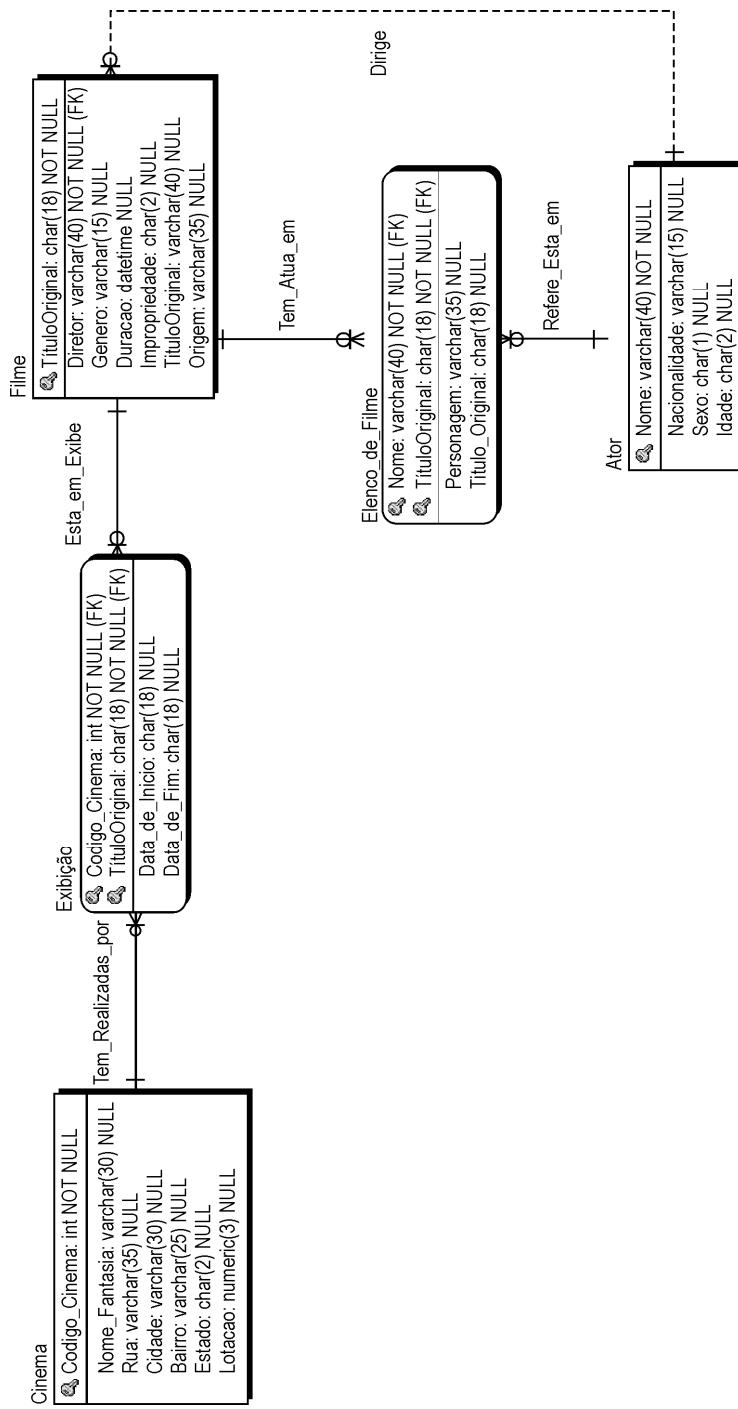
Modelo Físico para Administração de Cirurgias



Modelo Físico para Sistema de Vendas (ver capítulo de SQL)



Modelo Físico de Cinemas e Filmes



MAPEAMENTO DE OBJETOS - ER

A adoção das metodologias de desenvolvimento orientadas a objetos como um padrão de mercado levou a uma mudança radical na estruturação e organização de informação.

Contudo, a utilização de bancos de dados relacionais ainda é uma prática comum e será mantida por grande período de tempo. Graças à necessidade de trabalhar com essas bases de dados relacionais para o armazenamento persistente de dados, é comum adaptar os modelos de objetos na tentativa de compatibilizá-los com o modelo Entidade-Relacionamento, mais direcionado ao modelo relacional.

Para piorar ainda mais este quadro, é notório o esforço aplicado no processo de persistência manual dos objetos no banco de dados, o que força os desenvolvedores de aplicações a terem de dominar a linguagem SQL e utilizá-la para realizar acessos ao banco de dados.

Estas duas questões principais levam a uma redução considerável na qualidade do produto final, construção de uma modelagem orientada a objetos inconsistente e a um desperdício considerável de tempo na implementação manual da persistência.

Apesar disso, não é possível ignorar a confiabilidade dos Sistemas de Gerenciamento de Bancos de Dados (SGBDs) relacionais, após anos de desenvolvimento e ajustes de performance que fazem dos bancos de dados relacionais a opção mais eficiente, se comparados à maioria dos SGBDs orientados a objetos disponibilizados no mercado nos dias de hoje.

Para permitir um processo de mapeamento entre sistemas baseados em objetos e bases de dados relacionais, foram propostas diversas ideias que convergiram para o conceito de **Camada de Persistência**.

Conceitualmente, Camada de Persistência de Objetos é uma biblioteca que permite a realização do processo de persistência (isto é, o armazenamento e manutenção do estado de objetos em algum meio não volátil, como um banco de dados) de forma transparente.

Graças à independência entre a camada de persistência e o repositório utilizado, também é possível gerenciar a persistência de um modelo de objetos em diversos tipos de repositório, teoricamente com pouco ou nenhum esforço extra.

A utilização deste conceito permite ao analista trabalhar como se estivesse em um sistema completamente orientado a objetos - utilizando métodos para incluir, alterar e remover objetos e uma linguagem de consulta para SGBDs orientados a objetos, comumente a linguagem OQL, para realizar consultas que retornam coleções de objetos instanciados. Mas destaque-se somente em um ambiente de SGBDs orientados a objetos e não no ambiente tradicional de SGBDs relacionais.

As vantagens decorrentes do uso de uma Camada de Persistência no desenvolvimento de aplicações orientadas a objetos são evidentes: a sua utilização isola os acessos realizados diretamente ao banco de dados da aplicação, bem como centraliza os processos de construção de consultas e operações de manipulação de dados em uma camada de objetos inacessível ao programador.

Esse encapsulamento de responsabilidades garante maior confiabilidade às aplicações e permite que, em alguns casos, o próprio SGBD ou a estrutura de suas tabelas possam ser modificados, sem trazer impacto à aplicação nem forçar a revisão e recompilação de códigos.

Deixar o programador atuar sem acesso a uma camada de objetos pode provocar erros de desconhecimento da estrutura de banco de dados sobre a qual ele opera suas instruções, levando a erros conceituais de projeto de consultas e regras de negócio.

Hoje existem diversos utilitários que permitem uma independência de código SQL nas aplicações orientadas a objetos e escritas em JAVA, por exemplo. Esses aplicativos realizam o acesso e manipulação do banco de dados por meio da inserção somente das condições de validação e ou seleção de dados, não envolvendo o programador em altos conhecimentos de SQL, por exemplo.

De qualquer maneira existe a necessidade de não simplesmente traduzir diretamente em um script SQL de criação de banco de dados o modelo de classes de um projeto OO, pois não nos permite nenhuma interação do mundo real de negócios com a persistência que vamos utilizar, que é um banco de dados relacional. Isso provoca a existência de mapeamento de objetos em um modelo de dados Entidade-Relacionamento.

Mapeamento de Objetos para Tabelas (ER)

Para permitir a correta persistência de objetos em um banco de dados relacional, algo deve ser feito no tocante à forma como os dados serão armazenados.

No decorrer deste capítulo, detalhamos algumas sugestões de como é feito o mapeamento de cada um dos elementos de um objeto: seus atributos, relacionamentos e classes descendentes (herança).

Ao transpor um objeto para uma tabela relacional, os seus atributos são mapeados em colunas dessa tabela.

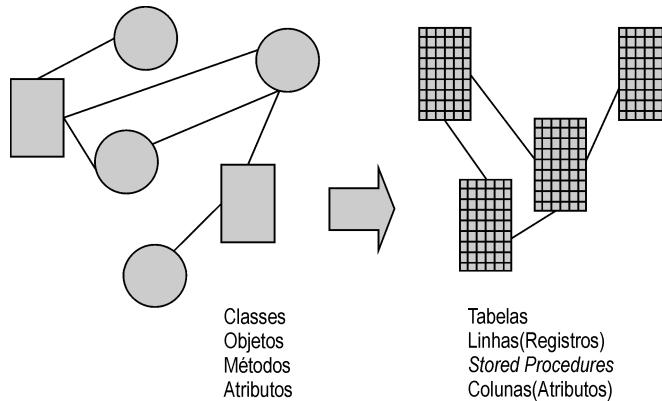
Entretanto, esse processo de mapeamento deve levar em consideração fatores como os tipos dos dados e o tamanho máximo dos campos (no caso de números e **strings**).

Também é importante destacar que, em muitas vezes, os atributos de um objeto não devem ser obrigatoriamente uma coluna em uma tabela.

Como exemplo, podemos citar atributos que são o valor total de alguma coisa: pois esse dado poderia ser armazenado no objeto para fins de consulta, mas mantê-lo no banco de dados simplesmente talvez não seja interessante, por tratar-se de um valor que pode ser obtido por processamento de uma consulta.

Além disso, existem casos em que um atributo pode ser mapeado para diversas colunas, pois é definido no modelo de forma genérica sem preocupação com composição e atributos como no modelo relacional (exemplos incluem endereços completos, nome dividido em 'primeiro nome' e 'sobrenome' no banco de dados) ou vários atributos que podem ser mapeados para uma mesma coluna (prefixo e número de telefone, por exemplo).

Em uma visão minimalista o mapeamento de objetos em modelo relacional (ER) é o da figura apresentada em seguida:



A transposição do modelo de classes para o modelo relacional tem como objetivo final a criação da base de dados do sistema, coerente com a modelagem da fase de análise.

As regras descritas não devem ser interpretadas como "leis" rígidas de transposição, mas uma indicação suscetível de adaptação em função da análise do problema em questão, para que possam traduzir rigorosamente a realidade.

Regra 1

Todas as tabelas (ou relações) devem ter uma chave primária, um conjunto mínimo de atributos que permitam identificar univocamente cada linha da tabela. No caso de não existirem atributos que satisfaçam esta condição, é preciso criar um identificador único designado por id.

Aqui provavelmente começa um dos problemas da transposição de classes em tabelas, pois existem classes que contêm coleções de objetos e classes que têm objetos dentro de objetos.

Por exemplo, classe departamento que tem incluso um objeto diretor e um conjunto de objetos seção.

```
Class Departamento {  
    String nome;  
    Empregado Diretor  
    Seção seções[];  
}
```

```
class Emprego {  
    string nome;  
    Tarefa tarefas [] ,  
    ...
```

Criar na tabela correspondente à classe original, por exemplo (departamento), uma coluna com o nome do atributo/objeto incluso (diretor), desempenhando o papel de chave estrangeira para a tabela correspondente à outra classe (empregado), pois muitas vezes não é representada esta associação no modelo de classes.

Classes com Coleções de Objetos

Se a relação entre a classe contentora e a classe contida for um-para-muitos, como no exemplo anterior, departamento e seção, inserir na tabela correspondente à classe contida o identificador da classe contentora (chave principal da respectiva tabela). Inserir na entidade seção a chave primária de departamento como chave estrangeira do relacionamento.

```
Class empregado {  
    String nome;  
    Tarefa tarefas[];  
    .....  
}
```

Se a relação entre a classe contentora (empregado) e a classe contida (tarefa) for muitos-para-muitos, criar uma terceira tabela que contém duas colunas.

A primeira coluna contém o identificador da classe contentora (empregado), e a segunda coluna contém o identificador da classe contida (tarefa). Com isso criamos uma entidade associativa entre as duas entidades relacionadas de muitos-para-muitos.

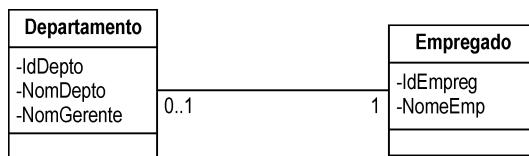
Regra 2

As tabelas resultam exclusivamente das classes do modelo e das associações de muitos-para-muitos.

Regra 3 - Transposição de Associações Um-para-Um

Nesse tipo de associação uma das tabelas deve herdar como um atributo normal (chave estrangeira) a chave primária da outra tabela.

A determinação da tabela que herdará a chave estrangeira fica a critério do analista e da interpretação que faz da realidade, devendo optar pelo que fizer mais sentido, lembrando que a associação um-para-um somente pode ser garantida totalmente se a chave estrangeira for no modelo físico definida como única da mesma forma que a chave primária, transformando-se assim em uma chave candidata.



Em princípio, o esquema da tabela que herdará a chave estrangeira corresponde à classe que tiver um menor número de ocorrências potenciais.

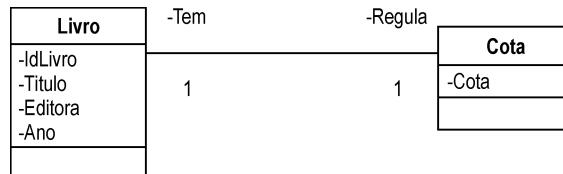
- ▶ **Departamento**=**{IDDEPTO, NOMDEPTO, NOMGERENTE}**
- ▶ **Empregado**=**{IDEMPREG, NOMEEMP}**

Este caso é melhor solucionado com:

- ▶ **Departamento**=**{IDDEPTO, NOMDEPTO, IDEMPREGERENTE}**
- ▶ **Empregado**=**{IDEMPREG, NOMEEMP}**

Outra solução é a viabilização de que esta relação seja transformada em um simples atributo descritivo do lado da classe maior quando uma das classes possuir somente um atributo além de seu identificador.

- ▶ **Livro**=**{IDLIVRO, TITULO, EDITORA, ANO, COTA}**
- ▶ **Cota**=**{COTA}**

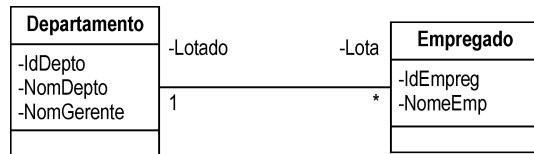


Se estas duas classes tiverem sido modeladas assim, podem se transformar em uma única tabela:

- ▶ **Livro**=**{IDLIVRO, TITULO, EDITORA, ANO, COTA}**

Regra 4 - Transposição de Associações Um-para-Muitos

Numa associação de um-para-muitos a tabela cujos registros são suscetíveis de serem endereçados diversas vezes é a que herda a referência da tabela cuja correspondência é unitária.

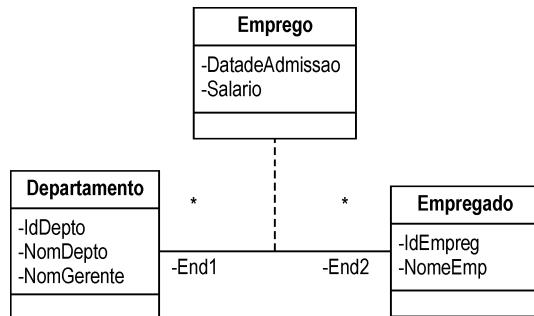


Logo, temos então:

- ▶ **Departamento**=**{IDDEPTO, NOMDEPTO, IDEMPREGERENTE }**
- ▶ **Empregado**=**{IDEMPREG, NOMEEMP, IDDEPTO}**

Regra 5 - Transposição de Associações Um-para-Muitos com Classe de Associação

Neste caso aplica-se a regra da associação de um-para-muitos (Regra 4) e os atributos da classe associativa são herdados como atributos normais pela tabela que herda a chave estrangeira.



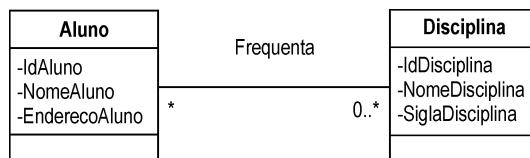
Resultado:

- ▶ **Departamento**=**{IDDEPTO, NOMDEPTO, IDEMPREGERENTE }**
- ▶ **Empregado**=**{IDEMPREG, NOMEEMP, IDDEPTO, DATADEADMISSAO, SALARIO}**

Regra 6 - Transposição de Associações Muitos-para-Muitos

A associação de muitos-para-muitos caracteriza-se por fazer a correspondência múltipla entre os elementos das classes associadas, sendo o número de correspondências o máximo permitido pela combinação de dois elementos com ocorrências variáveis (número de registos).

Assim, a regra de transposição dessa associação dá origem a uma tabela representativa da associação em que a chave primária é composta pelas chaves primárias das tabelas associadas à famosa e já conhecida entidade ou tabela associativa.

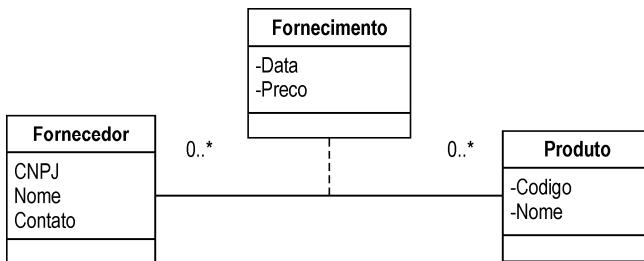


Logo, temos as seguintes tabelas:

- ▶ **Aluno**={IDALUNO, NOMEALUNO, ENDERECOALUNO}
- ▶ **Disciplina**={IDDISCIPLINA, NOMEDISCIPLINA, SIGLADISCIPLINA}
- ▶ **Frequenta**={IDDISCIPLINA, IDALUNO}

Regra 7 - Transposição de Associações Muitos-para-Muitos com Classe de Associação

Neste caso aplica-se a regra da associação de muitos-para-muitos (Regra 6) e os atributos da classe associativa permanecem na tabela que é gerada para mapear a associação.



- ▶ **Fornecedor**={CNPJ, NOME, CONTATO}
- ▶ **Produto**={CODIGO, NOME}
- ▶ **Fornecimento**={CNPJ, CODIGO, DATA, PRECO}

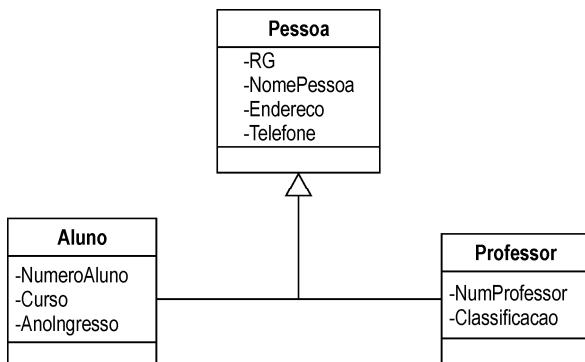
Regra 8 - Transposição de Generalizações

Nas generalizações duas situações distintas podem ocorrer:

- As classes **filhas** (subclasses) têm entidade própria independentemente da classe **pai** (superclasse).

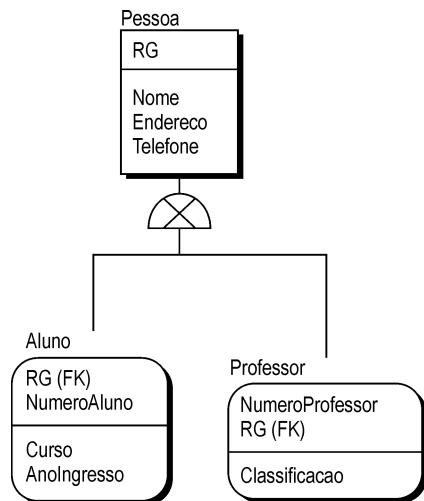
Neste caso, a chave primária das tabelas que implementam as classes *filhas* é obtida por meio dos atributos da própria tabela.

É preciso criar um atributo chave primária para a tabela que implementa a classe **pai**, e essa tabela deve ter uma propriedade discriminante (um atributo) que indica a qual das **filhas** o registro diz respeito.

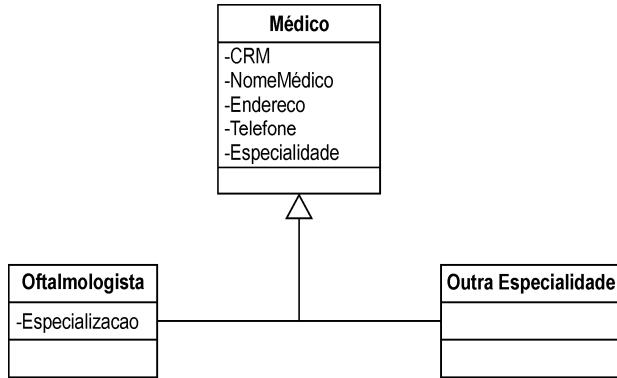


Todos os atributos que constituem a chave primária da tabela que implementa a superclasse terão de constar nas tabelas que implementam as subclasses como atributos normais (como chaves estrangeiras).

Os subtipos tornam-se tabelas carregando o identificador do conjunto ao qual pertencem na formação de sua chave primária.



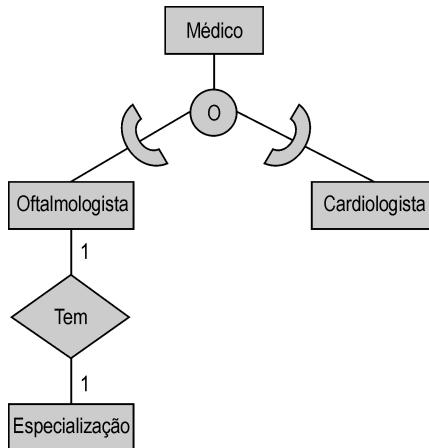
- ▶ **Pessoa**={RG, NOME, ENDERECHO, TELEFONE}
- ▶ **Aluno**={RG, NUMEROALUNO, CURSO, ANOINGRESSO}
- ▶ **Professor**={RG, NUMEROPROFESSOR, CLASSIFICACAO}



2. As classes **filhas** só têm identidade enquanto associadas à classe **pai**.

Neste caso, a chave primária da tabela que implementa a classe **pai** é obtida por meio dos atributos da própria tabela.

Importante entender a equivalência existente com um modelo lógico de dados neste caso, pois temos uma generalização disjunta, em que existem subtipos determinantes com atributos específicos de cada subtipo. No caso o subtipo oftalmologista possui um atributo específico que é inserido em um nova entidade que fica associada ao subtipo.



Também é necessário criar uma propriedade discriminante para indicar a qual das subclasses o registro diz respeito. Neste caso a propriedade determinante pode ser implementada como uma regra de negócio. Mas quando o valor de especialidade, por exemplo, for igual a oftalmologista, é a condição para a existência do relacionamento.

As tabelas correspondentes às classes **filhas** herdarão a chave primária da tabela que implementa a classe **pai**, caracterizando relacionamentos um-para-um.

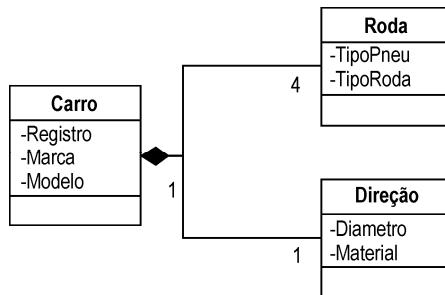
- ▶ Médico={CRM, NOMEMEDICO, ENDEREÇO, TELEFONE, ESPECIALIDADE}
- ▶ Especialização={CRM, ESPECIALIZACAO}

Regra 9 - Transposição de Agregações

Existem dois tipos de agregação:

1. As agregações simples

No caso, a agregação tem como objetivo indicar a existência de uma relação em que os elementos da classe agregada fazem parte de um todo, representado pela classe agregadora.

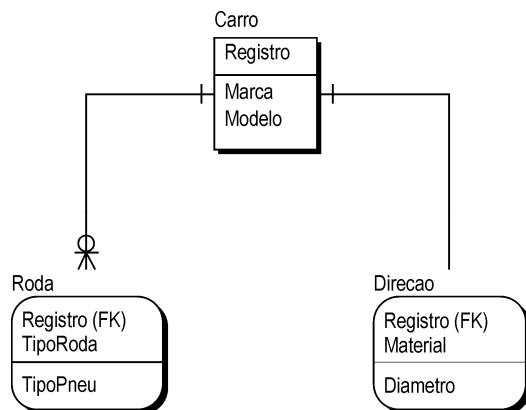


A transposição para o modelo relacional obedece às regras de transposição das associações com a mesma multiplicidade (Regras 3, 4 e 6).

O que temos é um formato de composição, ou seja, relacionamentos distintos em entidades dependentes, que só têm sentido quando da existência da entidade da qual dependem. São na realidade entidades fracas.

- ▶ **Carro**={REGISTRO, MARCA, MODELO}
- ▶ **Roda**={REGISTRO, TIPORODA, TIOPNEU}
- ▶ **Direção**={REGISTRO, DIAMETRO, MATERIAL}

O que corresponde ao modelo seguinte:



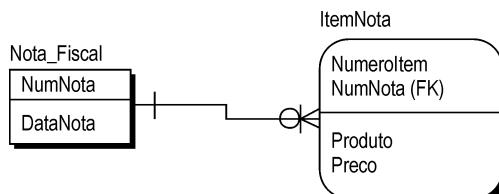
2. As composições

A composição é também uma agregação, mas com um sentido de dependência mais forte. Neste caso, os elementos agregados fazem parte da composição que os une. No caso de a composição ser eliminada, também os seus componentes desaparecem.

Na transposição para o relacional, a chave primária das tabelas que representam as classes componentes é composta pela respectiva chave primária, associada à chave primária da tabela referente à classe composição.



- ▶ **NotaFiscal=**{NUMNOTA, DATANOTA}
- ▶ **ItemNota=**{NUMNOTA, NUMEROITEM, PRODUTO, PRECO}



Considerações Finais

Acreditamos que com o material disposto neste capítulo auxiliamos sensivelmente o trabalho de conversão, transposição de classes de dados para um modelo Entidade-Relacionamento.

Devemos considerar que mesmo com a existência nas ferramentas de modelagem UML de processo para conversão automática com geração de scripts de bancos de dados a partir do modelo de classes, ela não supre a aderência aos conceitos de um modelo ER.

Com isso criam-se situações que não correspondem ao que seria normalmente modelado utilizando os conceitos e extensões do modelo Entidade-Relacionamento.

Por este motivo é recomendável realizar sempre manualmente essa transposição.

ÁLGEBRA RELACIONAL

Álgebra Relacional e Operações Relacionais

A álgebra relacional é uma linguagem de consulta procedural. Ela consiste em um conjunto de operações que tomam uma ou duas tabelas como entrada e produzem uma nova tabela como resultado.

Essas operações baseiam-se na teoria dos conjuntos (as tabelas correspondem a conjuntos).

Existem cinco operações fundamentais na álgebra relacional que são selecionar, projetar, renomear, (unárias) - produto cartesiano, união e diferença de conjuntos (binárias).

Além das operações fundamentais, existem outras, como interseção de conjuntos, ligação natural, dentre outras, que são definidas em termos das operações fundamentais.

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

O entendimento e o domínio das operações de álgebra relacional são fundamentais para a compreensão da linguagem SQL e um auxiliar excelente no momento de construção de consultas, pois dota o analista de conhecimento sobre a operação interna que é realizada pelo Sistema Gerenciador de Banco de Dados, tanto no tocante à recuperação dos dados como na análise da performance que a execução de um comando SQL vai possuir conforme a construção utilizada na consulta.

Todo o comando SQL pode ser desmembrado em operações de álgebra relacional, desta forma a visualização da sequência de execução das operações de álgebra relacional possibilita ao analista compreender a utilização de índices em determinadas consultas de SQL mais complexas, ou que envolvam várias tabelas.

Os Operadores de Álgebra Relacional

A álgebra relacional é uma linguagem formal composta por um conjunto de operadores que permite expressar qualquer operação de manipulação de dados em um banco de dados relacional. Esses operadores são classificados de várias formas conforme a funcionalidade.

As classificações são as seguintes:

Quanto à sua origem:

- ▶ **Fundamentais:** cinco operadores são ditos primitivos ou fundamentais, ou seja, através deles qualquer expressão de consulta de dados é possível. São eles: seleção, projeção, produto cartesiano, união e diferença.
- ▶ **Derivados:** derivam dos operadores fundamentais. São definidos para facilitar a especificação de certos procedimentos. São eles: intersecção, junção (normal e natural) e divisão.
- ▶ **Especiais:** operadores que não se enquadram nos itens anteriores. São eles: renomeação e alteração.

Quanto ao número de relações operandas:

- ▶ **Unários:** operam em uma única tabela. São eles: seleção, projeção, renomeação e alteração.
- ▶ **Binários:** operam em duas tabelas. São eles: união, intersecção, diferença, produto cartesiano, junção e divisão.

Quanto à origem da área da matemática:

- ▶ **Teoria dos conjuntos:** operadores usuais da teoria de conjuntos da matemática. São eles: união, intersecção, diferença e produto cartesiano.
- ▶ **Especiais:** operadores adicionais, definidos pela álgebra relacional para manipulação de dados. São eles: seleção, projeção, junção, divisão, renomeação e alteração.

Além desses operadores, é definido também o operador de atribuição que permite atribuir o resultado de uma expressão da álgebra a uma tabela.

Para que possamos entender mais facilmente as operações de álgebra relacional, vamos utilizar algumas tabelas para a explanação e apresentação de exemplos.

Estas são as tabelas que apresentamos em seguida, mas não se preocupe que vamos apresentá-las novamente sempre que for necessário ao bom entendimento deste assunto.

Estamos interessados em obter informações armazenadas nesse banco de dados (as três tabelas) e para isso devemos formular expressões em álgebra relacional combinando operações básicas, as quais vamos apresentar uma a uma de forma simplificada e também gráfica, para obtermos entendimento de seu significado prático e notação clássica.

Tabela Cargo

CdCargo	NumCargo	VlrSalario
C1	Aux Vendas	350,00
C3	Vendedor	800,00
C7	Diretor	2500,00
C2	Vigia	400,00
C5	Gerente	1000,00
C4	Aux Cobrança	250,00

Tabela Departamento

CdDept	NumDept	RamalTel
D1	Assist.Técnica	2246
D2	Estoque	2589
D3	Administração	2772
D4	Segurança	1810
D5	Vendas	2599
D6	Cobrança	2688

Tabela Funcionário

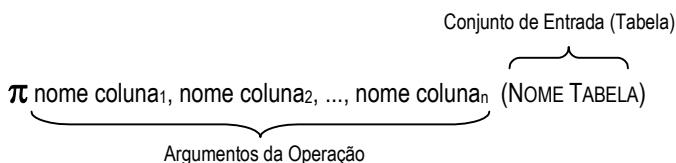
NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/8/2003	M	C3	D5
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
195	Marta Silveira	5/1/2002	F	C1	D5
139	Ana Luiza Magalhães	12/1/2003	F	C4	D6
123	Pedro Sergio Doto	29/6/2003	M	C7	D3
148	Larissa Silva	1/6/2002	F	C4	D6
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/2/2000	M	C2	D4

Vamos imaginar então que desejamos obter o nome completo de todos os funcionários que existem em nosso banco de dados.

Para isso vamos utilizar uma operação denominada projeção, aplicada sobre a tabela funcionário que contém esta informação.

Operação de Projeção

Na literatura que trata de bancos de dados indica-se por π (a letra grega pi) e produz um conjunto (tabela) em que há um elemento para cada elemento (linha) do conjunto (tabela) de entrada, e a estrutura dos membros do conjunto (tabela) resultante é definida nos argumentos da operação.



Pode ser entendida como uma operação que filtra as colunas de uma tabela. Por operar em apenas um conjunto de entrada, a projeção é classificada como uma operação unária.

Apesar de que vamos apresentar operações conjuntas, convém salientar que o operador de projeção sempre atua em **uma tabela somente**, seja ela do nosso banco de dados ou uma tabela resultante de outra operação relacional executada, motivo pelo qual a denominação de operação unária.

Então, em nosso exemplo temos:



Este é o formato do operador, projeção da coluna NomeFunc da tabela funcionário.

Vamos agora olhar a operação e o resultado na tabela para melhor entendê-la.

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/8/2003	M	C3	D5
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
195	Marta Silveira	5/1/2002	F	C1	D5
139	Ana Luiza Magalhães	12/1/2003	F	C4	D6
123	Pedro Sergio Doto	29/6/2003	M	C7	D3
148	Larissa Silva	1/6/2002	F	C4	D6
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/2/2000	M	C2	D4



Observe que a tabela resultante obedece à regra de ter somente as colunas constantes no comando de projeção.

Vamos ver mais um exemplo da mesma tabela. Digamos que queremos saber além do nome, o sexo de todos os funcionários.

O formato do comando de projeção então seria:

$$\pi_{\text{NOMEFUNC, SEXO}} (\text{FUNCIONÁRIO})$$

E o resultado aplicado na mesma tabela funcionário apresenta como tabela resultante a seguinte:

NomeFunc	Sexo
Luis Sampaio	M
Carlos Pereira	M
Jose Alves	M
Luis Paulo Souza	M
Marta Silveira	F
Ana Luiza Magalhães	F
Pedro Sergio Doto	M
Larissa Silva	F
Roberto Fernandes	M
Sergio Nogueira	M

Grave bem, projeção é a operação que recupera colunas determinadas (especificadas) de uma tabela em um banco de dados relacional.

$$\pi_{\text{NOME COLUNA1}, \text{NOME COLUNA2}, \dots, \text{NOME COLUNAN}} (\text{NOME TABELA})$$

Praticamente devemos considerar que toda operação sobre tabelas de um banco de dados, quando queremos obter algum resultado visível, ou seja, queremos ver algum dado resultante, necessita utilizar a operação de projeção.

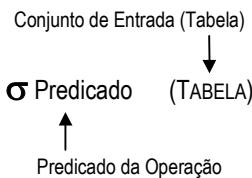
A operação de projeção é a efetivação de resultado de qualquer tipo de consulta. Devemos lembrar sempre que as colunas utilizadas como argumento da operação devem obrigatoriamente existir na tabela de entrada da operação.

Entretanto, existem casos em que queremos descartar algumas ocorrências do conjunto de entrada, ou seja, da tabela de entrada da operação. Por exemplo, se desejamos saber os funcionários que são do sexo masculino somente.

Neste caso usamos a operação de seleção, que alguns autores e professores destacam como sendo de restrição.

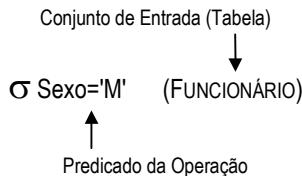
Operação de Seleção

Indicada por σ (a letra grega *sigma*), é uma operação que, para um conjunto inicial fornecido como entrada, produz um subconjunto estruturalmente idêntico, mas apenas com os elementos do conjunto entrada que atendem a uma determinada condição (também chamada de *predicado*).



A seleção pode ser entendida como a operação que filtra, seleciona as linhas de uma tabela, e é também uma operação de projeção, uma operação unária, operando em um único conjunto de dados.

Para melhor visualizar a estrutura da operação, vamos apresentar como ela é sintaticamente no caso de desejarmos obter os funcionários do sexo masculino.

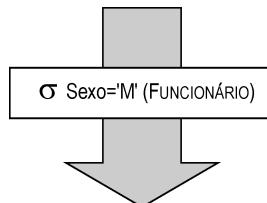


A execução dessa operação produz uma tabela resultante, que nada mais é do que o conjunto de ocorrências de funcionários que atendem à condição de a coluna *sexo* possuir o valor '*M*', ou seja, os funcionários do sexo masculino.

Um detalhe importante é que esse subconjunto gerado possui todos os atributos do conjunto de entrada (tabela funcionários).

Senão, vejamos:

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/8/2003	M	C3	D5
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
195	Marta Silveira	5/1/2002	F	C1	D5
139	Ana Luiza Magalhães	12/1/2003	F	C4	D6
123	Pedro Sergio Doto	29/6/2003	M	C7	D3
148	Larissa Silva	1/6/2002	F	C4	D6
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/2/2000	M	C2	D4



NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/8/2003	M	C3	D5
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
123	Pedro Sergio Doto	29/6/2003	M	C7	D3
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/2/2000	M	C2	D4

Analisando o resultado, efetivamente comprovamos que retornaram todas as colunas da tabela original, funcionários, das linhas desta tabela que satisfizeram a condição, predicado, $\text{Sexo} = 'M'$.

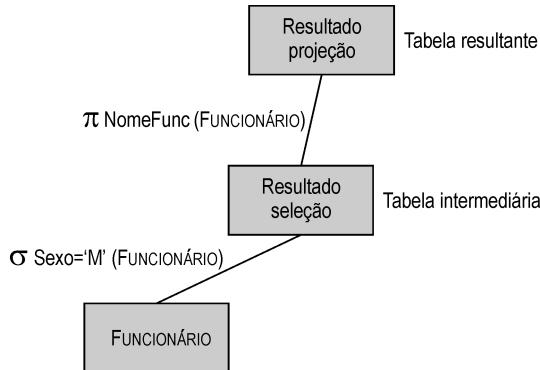
Agora, considerando que desejamos saber somente o nome e a data de admissão dos funcionários de sexo masculino, o que fazer?

Neste caso começamos a utilizar operações combinadas, pois queremos realizar a seleção dos dados, porém projetar somente duas colunas do resultado dessa seleção.

Em primeiro lugar devemos entender as precedências de operações quando as utilizamos combinadas.

Se você quiser projetar, ver colunas determinadas, específicas de uma tabela, para uma determinada condição, primeiro deve selecionar as linhas que atendem a esta condição, para então desse resultado realizar a projeção das colunas desejadas.

Vejamos então graficamente:



Vamos ver agora o que acontece com as tabelas, que resulta essa operação combinada e qual a sua sintaxe:

$\pi_{\text{NOMEFUNC}, \text{DTADMISSÃO}} (\sigma_{\text{SEXO}='M'} (\text{FUNCIONÁRIO}))$

Comparando o gráfico com a sintaxe do comando para a operação, podemos observar que a escrevemos de cima para baixo, ou seja, *top-down*.

Este esquema de precedência deve sempre ser considerado, e aconselhamos por prática utilizar um projeto gráfico para a execução de operações relacionais, pois simplifica o entendimento e proporciona maior clareza de entendimento das operações em si.

Vamos ilustrar novamente para fixar agora as tabelas do início até o fim da operação:

Tabela Inicial - Funcionários

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/8/2003	M	C3	D5
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
195	Marta Silveira	5/1/2002	F	C1	D5
139	Ana Luiza Magalhães	12/1/2003	F	C4	D6
123	Pedro Sergio Doto	29/6/2003	M	C7	D3
148	Larissa Silva	1/6/2002	F	C4	D6
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/2/2000	M	C2	D4

Execução da operação de seleção - $\sigma_{\text{SEXO}='M'} (\text{FUNCIONÁRIO})$ resulta a tabela temporária, intermediária, apresentada em seguida:

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/8/2003	M	C3	D5
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
123	Pedro Sergio Doto	29/6/2003	M	C7	D3
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/2/2000	M	C2	D4

Execução da operação de projeção da tabela intermediária:

$\pi_{\text{NOMEFUNC}, \text{DTADMISSÃO}} (\sigma_{\text{SEXO}='M'} (\text{FUNCIONÁRIO}))$

em que $\sigma_{\text{SEXO}='M'}$ (Funcionário) representa a tabela intermediária, e temos como resultado final da operação a tabela seguinte:

NomeFunc	DtAdmissão
Luis Sampaio	10/8/2003
Carlos Pereira	2/3/2004
Jose Alves	23/5/2002
Luis Paulo Souza	10/12/2001
Pedro Sergio Doto	29/6/2003
Roberto Fernandes	15/10/2003
Sergio Nogueira	10/2/2000

Já aconteceu de perguntarem em aulas algo como:

Mas e a coluna sexo, por que não apareceu, se é o argumento de seleção?

Para muitos pode parecer uma pergunta sem sentido, mas é comum alunos se embalar com a conjunção de comandos nas operações relacionais.

A razão que deve ficar clara é que na tabela resultado **somente as colunas projetadas**, que fazem parte do argumento do comando de projeção, que é o último a ser executado, estarão no resultado final.

Por estes motivos sempre são utilizados esquemas gráficos tanto para explicar como para projetar operações e queries.

A álgebra relacional é considerada uma **linguagem procedural**, uma vez que requer sempre a existência de definição quanto à ordem em que as operações serão realizadas.

Linguagens em que apenas definimos o resultado que é desejado, sem fazer menção nenhuma à forma e sequência de como isso deve ser feito, são chamadas de **linguagens não procedurais**.

Suponha agora que precisamos obter o nome completo, a data de admissão, o nome do cargo e o salário de cada funcionário cadastrado.

Observe que para essa consulta temos um diferencial que podemos dizer que é um fato novo na utilização de operações relacionais até o momento, que é a referência a colunas de mais de uma tabela, uma vez que o nome e a data de admissão fazem parte da tabela funcionário, enquanto o nome do cargo e salário existem apenas na tabela cargos.

Isso seria problemático, pois as duas operações que conhecemos até o momento são unárias, e temos necessidade de combinar os dados de mais de uma tabela neste caso.

Como resolver este impasse?

Para situações como esta existe uma operação chamada produto cartesiano que pode ser então utilizada conjuntamente com outras operações para obtermos o resultado desejado.

Produto Cartesiano

A notação adotada em matemática ($\text{conjunto1} \times \text{conjunto2}$) para representar o produto cartesiano entre dois conjuntos (uma operação binária) indica tipicamente a natureza dessa operação: o resultado de um produto cartesiano de duas tabelas é uma terceira tabela com as combinações possíveis entre os elementos das tabelas originais.

Essa tabela resultante possui um número de colunas que é igual à soma do número de colunas das duas tabelas iniciais, e um número de linhas igual ao produto do número de linhas das duas tabelas.

Se fizermos o produto cartesiano de uma tabela X que possua quatro colunas e dez linhas com uma tabela Y em que existem três colunas e seis linhas, a tabela resultante terá:

$$4 \text{ colunas} + 3 \text{ colunas} = 7 \text{ colunas}$$

e

$$10 \text{ linhas} * 7 \text{ linhas} = 70 \text{ linhas.}$$

Porque cada linha da tabela resultante corresponderá à concatenação de uma linha da primeira tabela com uma linha da segunda.

Então, a primeira linha da tabela X será concatenada com sete linhas da tabela Y, a segunda linha da tabela X será concatenada com as mesmas sete linhas da tabela Y e assim por diante, chegando a um total de 70 linhas na tabela resultado.

O produto cartesiano não é muito usado como um fim em si mesmo, ou seja, dificilmente estaremos interessados em saber todas as combinações possíveis entre as linhas de duas tabelas, pois a utilidade desse tipo de conhecimento é muito discutível, apesar de existirem casos de sua utilização prática.

Entretanto, esta é a forma primitiva de que dispomos para unir informações de duas tabelas heterogêneas para posterior processamento.

Se desejarmos obter apenas algumas das combinações resultantes de um produto cartesiano, será necessário executar uma operação de seleção do seu resultado, de forma a descartar as combinações que não são de interesse entre as linhas das tabelas originais.

Essa operação mostra todos os atributos das relações envolvidas.

A forma geral é:

Tabela A × Tabela B

Antes que perguntem, a resposta é não.

Não existe nenhum símbolo grego associado a esta operação relacional. O símbolo é exatamente o operador \times entre o nome das tabelas participantes.

É importante ressaltar neste ponto a questão colunas do resultado de um produto cartesiano, pois apresentamos a regra numérica para o número de colunas, porém não apresentamos de forma bem clara ainda a composição dessas colunas.

Se considerarmos a estrutura de duas de nossas tabelas, por exemplo, a tabela funcionários e a tabela cargo:

A tabela funcionários possui os seguintes atributos:

- ▶ NumReg
- ▶ Sexo
- ▶ NomeFunc
- ▶ CdCargo
- ▶ DtAdmissao
- ▶ CdDept

A tabela cargo possui os seguintes atributos:

- ▶ CdCargo
- ▶ NumCargo
- ▶ VlrSalario

Seguindo o raciocínio, temos seis colunas na primeira tabela e três colunas na segunda tabela, logo o resultado final terá nove colunas.

Como elas vão aparecer? Em que ordem? Quais serão as colunas resultantes?

A tabela resultante terá as seguintes colunas:

- ▶ NumReg
- ▶ NomeFunc
- ▶ DtAdmissao
- ▶ Sexo

- ▶ CdCargo
- ▶ CdDept
- ▶ CdCargo
- ▶ NumCargo
- ▶ VlrSalario

Exatamente nesta ordem, sendo esta a estrutura final de atributos da tabela resultado.

Agora vamos ver como fica o resultado, visualizando as tabelas de entrada da operação e a tabela de saída com dados do exemplo:

Funcionários × Cargo

Tabela Funcionários

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/8/2003	M	C3	D5
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
195	Marta Silveira	5/1/2002	F	C1	D5
139	Ana Luiza Magalhães	12/1/2003	F	C4	D6
123	Pedro Sergio Doto	29/6/2003	M	C7	D3
148	Larissa Silva	1/6/2002	F	C4	D6
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/2/2000	M	C2	D4

Tabela Cargos

CdCargo	NumCargo	VlrSalario
C1	Aux Vendas	350,00
C3	Vendedor	800,00
C7	Diretor	2500,00
C2	Vigia	400,00
C5	Gerente	1000,00
C4	Aux Cobrança	250,00

A tabela resultado será:

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDeptº	CdCargo	NumCargo	VlrSalario
101	Luis Sampaio	10/8/2003	M	C3	D5	C1	Aux Vendas	350,00
101	Luis Sampaio	10/8/2003	M	C3	D5	C3	Vendedor	800,00
101	Luis Sampaio	10/8/2003	M	C3	D5	C7	Diretor	2500,00
101	Luis Sampaio	10/8/2003	M	C3	D5	C2	Vigia	400,00
101	Luis Sampaio	10/8/2003	M	C3	D5	C5	Gerente	1000,00
101	Luis Sampaio	10/8/2003	M	C3	D5	C4	Aux Cobrança	250,00
104	Carlos Pereira	2/3/2004	M	C4	D6	C1	Aux Vendas	350,00
104	Carlos Pereira	2/3/2004	M	C4	D6	C3	Vendedor	800,00
104	Carlos Pereira	2/3/2004	M	C4	D6	C7	Diretor	2500,00
104	Carlos Pereira	2/3/2004	M	C4	D6	C2	Vigia	400,00
104	Carlos Pereira	2/3/2004	M	C4	D6	C5	Gerente	1000,00
104	Carlos Pereira	2/3/2004	M	C4	D6	C4	Aux Cobrança	250,00
.....
.....
22	Sergio Nogueira	10/2/2000	M	C2	D4	C1	Aux Vendas	350,00
22	Sergio Nogueira	10/2/2000	M	C2	D4	C3	Vendedor	800,00
22	Sergio Nogueira	10/2/2000	M	C2	D4	C7	Diretor	2500,00
22	Sergio Nogueira	10/2/2000	M	C2	D4	C2	Vigia	400,00
22	Sergio Nogueira	10/2/2000	M	C2	D4	C5	Gerente	1000,00
22	Sergio Nogueira	10/2/2000	M	C2	D4	C4	Aux Cobrança	250,00

Evidentemente não apresentamos todas as linhas da tabela resultado, pois não caberia nesta página, mas a demonstração de que para cada linha da tabela funcionários combinamos todas as linhas existentes na tabela cargo.

Acreditamos que seja possível o leitor compreender o resultado final desta operação.

Agora vamos estudar uma aplicação de operações relacionais em que possamos utilizar o produto cartesiano combinado com outras operações, e a forma como utilizá-las.

Supondo que desejamos saber o nome e a data de admissão dos funcionários que têm cargo de auxiliar de cobrança.

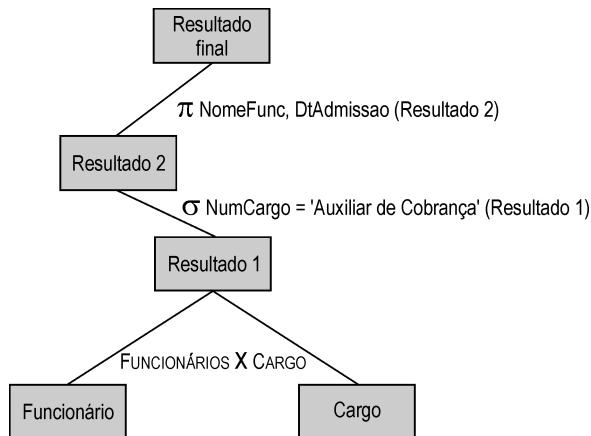
A primeira operação a ser realizada é um produto cartesiano entre a tabela funcionários e a tabela cargos, o que está evidente pelo que estudamos até este ponto.

A segunda operação é selecionar do resultado do produto cartesiano as linhas que satisfazem a condição de Nome do Cargo se igual a "auxiliar de cobrança" e **alguma coisa a mais** que vamos deixar como dúvida até este ponto:

NumCargo='Auxiliar de Cobrança'

A terceira operação é a projeção das colunas nome do funcionário e data de admissão do resultado desta seleção.

Então, graficamente:



Se estruturarmos o comando em uma única sentença, temos:

$\pi_{NOMEFUNC, DTADMISSÃO} (\sigma_{NUMCARGO='AUXILIAR DE COBRANÇA'} (FUNCIONÁRIOS \times CARGO))$

Cabe o questionamento: está correta esta aplicação de produto cartesiano combinada?

Para que você entenda, vamos colocar a tabela resultante para que possamos estudar o erro que acontece pela falta de um argumento no contexto todo.

NumReg	NomeFunc	DtAdmissao	Sexo	CdCargo	CdDept	CdCargo	NumCargo	VlrSalario
101	Luiz Sampaio	10/08/2003	M	C3	D5	C4	Aux Cobrança	250,00
104	Carlos Pereira	02/03/2004	M	C4	D6	C4	Aux Cobrança	250,00
134	Jose Alves	25/05/2002	M	C5	D1	C4	Aux Cobrança	250,00
121	Luis Paulo Souza	10/12/2001	M	C3	D5	C4	Aux Cobrança	250,00
195	Marta Silveira	05/01/2002	F	C1	D5	C4	Aux Cobrança	250,00
139	Ana Luiza Magalhães	12/01/2003	F	C4	D6	C4	Aux Cobrança	250,00
123	Pedro Sergio Doto	29/06/2003	M	C7	D3	C4	Aux Cobrança	250,00
148	Larissa Silva	01/06/2002	F	C4	D6	C4	Aux Cobrança	250,00
115	Roberto Fernando	15/10/2003	M	C3	D5	C4	Aux Cobrança	250,00
22	Sergio Nogueira	10/02/2000	M	C2	D4	C4	Aux Cobrança	250,00

"Você cometeu uma operação ilegal e o programa será fechado. Reinicie o computador."

Felizmente este é um livro e não aquele programa conhecido.

Como o produto cartesiano junta todas as linhas da primeira tabela com todas as linhas da segunda tabela, o resultado foi que todos os funcionários aparecem no resultado da primeira operação, conduzindo-nos a um resultado final errado, pois teremos uma tabela final com o conteúdo apresentado em seguida:

NomeFunc	DtAdmissão
Luis Sampaio	10/08/2003
Carlos Pereira	02/03/2004
Jose Alves	23/05/2002
Luis Paulo Souza	10/12/2001
Marta Silveira	05/01/2002
Ana Luiza Magalhães	12/01/2003
Pedro Sergio Doto	29/06/2003
Larissa Silva	01/06/2002
Roberto Fernandes	15/10/2003
Sergio Nogueira	10/02/2000

Talvez você ache estranho ensinar por meio de erros, mas a prática levou a esta metodologia neste caso, pois faz com que o estudante fixe as regras com maior amplitude.

Vejamos o que faltou:

Faltou informar a condição de que queríamos somente os funcionários com o cargo de auxiliar de cobrança.

No comando que realizamos, especificamos somente que o nome do cargo deveria ser auxiliar de cobrança, não especificando de nenhuma forma como associar que este era o cargo ao funcionário que efetivamente o possui.

Olhando as tabelas, vemos que somente os funcionários cujo atributo CdCargo for igual a 'C4' é que possuíram este cargo de auxiliar de cobrança.

Para descobrir o cargo verdadeiro de um funcionário, observando as tabelas e as chaves estrangeiras, conclui-se que é necessário que o atributo da tabela funcionário CdCargo seja igual ao atributo CdCargo da tabela resultante do produto cartesiano.

Funcionário.CdCargo = Cargo.CdCargo

Aqui entram as referências de cada tabela na identificação do atributo, pois não podemos ser ambíguos ao nos referirmos às colunas atributos de tabelas.

Desta forma, o critério de seleção correto é uma condição composta de dois critérios:

O campo NumCargo deve ser igual a 'Auxiliar de Cobrança' e Funcionário.CdCargo = Cargo.CdCargo.

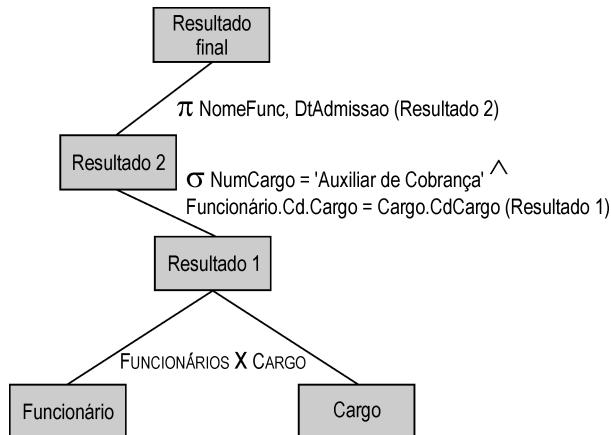
Como se representa então uma dupla condição, uma condição com um conectivo lógico 'E'?

O símbolo utilizado na álgebra relacional é o circunflexo ' \wedge ', tradicionalmente conhecido por representar este conectivo lógico.

Vamos então à montagem de nosso comando, agora de forma correta.

```
 $\pi$  NOMEFUNC,DTADMISSÃO ( $\sigma$  NUMCARGO='AUXILIAR DE COBRANÇA'  $\wedge$  FUNCIONÁRIO.CDCARGO = CARGO.CDCARGO  
(FUNCIONÁRIOS  $\times$  CARGO))
```

Para que não restem dúvidas, observe a representação gráfica do esquema da consulta agora corrigida.



Vamos para um outro exemplo simples para aumentarmos a fixação do conteúdo dessa operação relacional.

Sejam as tabelas de uma cantina, como num exemplo do professor Francisco Rapchan, em sua apostila de álgebra relacional, o qual achei muito interessante. Uma cantina, seus clientes e suas contas de fiado (pagamento futuro, conta pendurada). Temos neste caso duas tabelas, a tabela de clientes e a de fiados, as famosas continhas de botequim.

Observação: No Rio de Janeiro é muito comum ter conta em um botequim e este exemplo adapta-se perfeitamente.

Vamos usar tabelas pequenas de poucas ocorrências para facilitar a visualização.

Tabela Clientes

CodCliente	NomCliente
1	Luis Sampaio
2	Carlos Pereira
3	Jose Alves
4	Luis Paulo Souza

Tabela Contas Penduradas (Fiado)

CodCliente	Valor
1	50,00
2	43,00
3	28,00
4	15,00

Se desejarmos uma consulta com o nome dos clientes e o valor que estão devendo, devemos realizar um conjunto de operações combinadas, utilizando-nos de produto cartesiano como base das operações.

$$\sigma_{\text{CLIENTE.CODCLIENTE} = \text{CONTA.CODCLIENTE}} (\text{CLIENTES} \times \text{CONTA})$$

Com esta operação obtemos os clientes e suas contas.

CodCliente	NomCliente	CodCliente	Valor
1	Luis Sampaio	1	50,00
2	Carlos Pereira	2	43,00
3	Jose Alves	3	28,00
4	Luis Paulo Souza	4	15,00

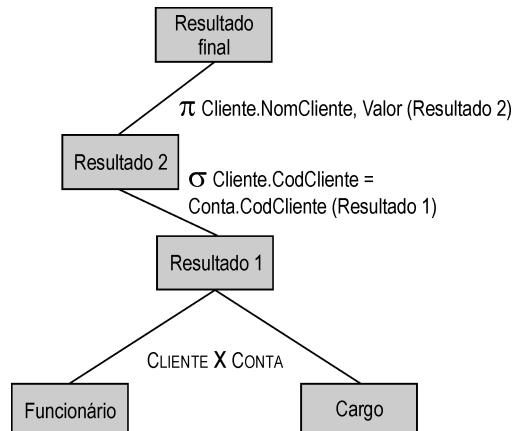
Mas como queremos no resultado somente o nome do cliente e o valor que ele está devendo, devemos completar este comando com a projeção das colunas que desejamos no resultado final:

$$\pi_{\text{NOMCLIENTE}, \text{VALOR}} (\sigma_{\text{CLIENTE.CODCLIENTE} = \text{CONTA.CODCLIENTE}} (\text{CLIENTES} \times \text{CONTA}))$$

E teremos, então, como resultado a tabela seguinte:

NomCliente	Valor
Luis Sampaio	50,00
Carlos Pereira	43,00
Jose Alves	28,00
Luis Paulo Souza	15,00

Voltando à nossa mania de visualizar tudo graficamente, observe o gráfico:



É importante salientar neste ponto que o **produto cartesiano não possui cardinalidade**, ou seja, **não importa** a ordem em que as tabelas são dispostas no comando, o resultado prático será sempre o mesmo.

Entretanto, a ordem de disposição das colunas no seu resultado muda conforme a ordem em que as tabelas forem escritas na operação.

Logo, a sequência das colunas para Cliente × Conta é diferente da sequência para Conta × Cliente:

Cliente × Conta

CodCliente	NomCliente	CodCliente	Valor
1	Luis Sampaio	1	50,00
1	Luis Sampaio	2	43,00
1	Luis Sampaio	3	28,00
1	Luis Sampaio	4	15,00
2	Carlos Pereira	1	50,00
2	Carlos Pereira	2	43,00
2	Carlos Pereira	3	28,00
2	Carlos Pereira	4	15,00
3	Jose Alves	1	50,00
3	Jose Alves	2	43,00
3	Jose Alves	3	28,00
3	Jose Alves	4	15,00
4	Luiz Paulo Souza	1	50,00
4	Luiz Paulo Souza	2	43,00
4	Luiz Paulo Souza	3	28,00
4	Luiz Paulo Souza	4	15,00

Conta × Cliente

CodCliente	Valor	CodCliente	NomCliente
1	50,00	1	Luis Sampaio
1	50,00	2	Carlos Pereira
1	50,00	3	Jose Alves
1	50,00	4	Luiz Paulo Souza
2	43,00	1	Luis Sampaio
2	43,00	2	Carlos Pereira
2	43,00	3	Jose Alves
2	43,00	4	Luiz Paulo Souza
3	28,00	1	Luis Sampaio
3	28,00	2	Carlos Pereira
3	28,00	3	Jose Alves
3	28,00	4	Luiz Paulo Souza
4	15,00	1	Luis Sampaio
4	15,00	2	Carlos Pereira
4	15,00	3	Jose Alves
4	15,00	4	Luiz Paulo Souza

Agora existem alguns casos em que necessitamos realizar produto cartesiano.

Com a mesma tabela, ou seja, a tabela vai aparecer duas vezes na mesma operação. Se utilizarmos o mesmo nome de tabela, vamos provocar uma ambiguidade, inviabilizando o comando, ou melhor, a operação relacional desejada.

Para este caso existe a operação de renomear, ou *rename*, que atribui à tabela uma segunda nomenclatura.

Operação Renomear

A operação de renomear uma tabela é utilizada sempre que ela aparece mais de uma vez em uma consulta. Como ao contrário dos bancos de dados o resultado de uma operação

em álgebra relacional não tem nome, também utilizamos a operação *rename* para nomear um resultado.

Essa operação é representada pela letra grega ρ (rho).

O formato geral é:

$$\rho <\text{novo nome}> (\text{TABELA})$$

Por exemplo:

$$\rho \text{ Cliente2} (\text{CLIENTE})$$

Este operador permite agora que se realizem operações como $\text{Cliente2} \times \text{Cliente}$.

Vamos criar um exemplo em que possamos visualizar e entender mais claramente a operação.

Seja a tabela cliente seguinte:

CodCliente	NomCliente	RuaCliente	CidadeCliente
1	Luis Sampaio	Rua A	Rio de Janeiro
2	Carlos Pereira	Rua B	Niterói
3	Jose Alves	Rua C	Rio de Janeiro
4	Luis Paulo Souza	Rua B	Niterói

Pode parecer uma consulta bem ridícula, mas serve aos nossos propósitos de explicar a utilização da operação renomear:

Quais são os clientes que moram na mesma rua e cidade que Carlos Pereira?

Pela simples leitura da tabela você vai responder à consulta, mas queremos realizá-la com instruções de álgebra relacional, então vamos lá.

Em primeiro lugar vamos definir como selecionar os dados relativos a Carlos Pereira.

Devemos selecionar a linha que contém os dados de Carlos Pereira e projetar as colunas rua e cidade.

O comando seguinte é o que efetiva a seleção do cliente específico:

$$\sigma \text{ NOMCLIENTE} = \text{"CARLOS PEREIRA"} (\text{CLIENTE})$$

Em seguida devemos projetar as colunas desejadas:

$$\pi \text{ RUA CLIENTE, CIDADE CLIENTE} ()$$

Concluindo o comando, temos:

$$\pi_{\text{RUACLIENTE}, \text{CIDADECLIENTE}} (\sigma_{\text{NOMCLIENTE} = "CARLOS PEREIRA"} (\text{CLIENTE}))$$

Não se esqueça de que devemos fechar dois conjuntos de parênteses.

Feito isso, devemos pensar que é contra essa operação, ou melhor, tomando como base de tudo essa operação, vamos descobrir quem tem a mesma rua e cidade que o nosso amigo Carlos Pereira.

Vamos realizar o produto cartesiano do resultado obtido com a tabela cliente, porém renomear para que não existam ambiguidades em nossa consulta.

$$\rho_{\text{Cliente2}} (\text{CLIENTE})$$

Vamos realizar a esquematização gráfica da ordem de execução dos comandos para que possamos entender melhor e mais claramente esta consulta, até certo ponto complexa em sintaxe:

O primeiro passo já demos que foi selecionar e projetar as colunas de rua e endereço da tabela cliente para o cliente = "Carlos Pereira", que no gráfico chamamos de **resultado1**.

RuaCliente	CidadeCliente
Rua B	Niterói

O segundo passo é realizarmos o produto cartesiano entre o resultado do primeiro passo e a tabela cliente agora renomeada como Cliente2, que no gráfico denominamos de **Resultado2**.

$$(\sigma_{\text{NOMCLIENTE} = "CARLOS PEREIRA"} (\text{CLIENTE})) \times (\rho_{\text{Cliente2}} (\text{CLIENTE})) \quad \text{Resultado2}$$

Como terceiro passo devemos selecionar as linhas do resultado2 em que a seguinte condição é satisfeita.

$$\sigma_{\text{CLIENTE2.RUACLIENTE} = \text{CLIENTE.RUACLIENTE} \wedge \text{CLIENTE2.CIDADECLIENTE} = \text{CLIENTE.CIDADECLIENTE}} (\text{Resultado2})$$

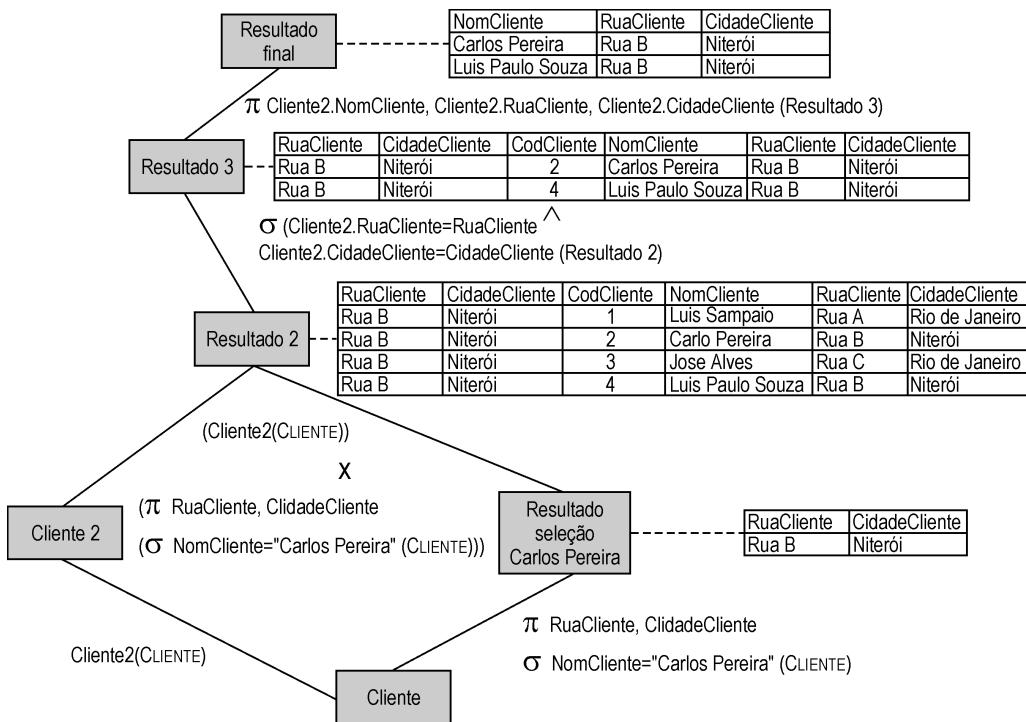
Resultado3

As linhas iguais às de Carlos Pereira serão selecionadas e teremos o **Resultado3**.

O quarto passo é a projeção das colunas do Resultado3 que nos interessam, sendo NomCliente, RuaCliente e CidadeCliente.

$$\pi_{\text{CLIENTE2.NOMCLIENTE}, \text{CLIENTE2.RUACLIENTE}, \text{CLIENTE2.CIDADECLIENTE}} (\text{Resultado3})$$

Observe o gráfico e depois leia o comando completo, prestando atenção nos parênteses que separam cada bloco.



$\pi_{\text{CLIENTE2.NOMCLIENTE}, \text{CLIENTE2.RUACLIENTE}, \text{CLIENTE2.CIDADECLIENTE}}$

$\sigma_{\text{CLIENTE2.RUACLIENTE} = \text{CLIENTE.RUACLIENTE} \wedge \text{CLIENTE2.CIDADECLIENTE} = \text{CLIENTE.CIDADECLIENTE}}$

$(\sigma_{\text{NOMCLIENTE} = "CARLOS PEREIRA"} (\text{CLIENTE})) \times (\rho_{\text{Cliente2} (\text{CLIENTE})})$

)

)

Destacamos os parênteses para que você possa identificá-los em seu fechamento.

Agora, se analisarmos corretamente esta aplicação das operações relacionais, pode surgir a questão:

Por que a resposta veio com o Carlos Pereira novamente?

Não há como ver os clientes sem ver a linha relativa a Carlos Pereira?

A resposta é sim. É possível ver somente os clientes que têm a mesma rua e cidade de Carlos Pereira sem ter de mostrar os dados de Carlos Pereira.

Basta acrescentar condições de seleção ao terceiro passo apresentado. Vamos inserir a condição de que o NomCliente seja diferente de 'Carlos Pereira'.

Qual a simbologia utilizada para expressar esta condição?

NomCliente <> 'Carlos Pereira'

Entretanto, devemos cuidar da montagem desta condição porque em alguns casos pode implicar a inserção de novo conjunto de parênteses na sintaxe.

Então vejamos: a primeira condição que é composta de dois elementos de seleção é acrescida de um terceiro elemento pela conjunção e (^).

(Cliente2.RuaCliente = Cliente.RuaCliente ^ Cliente2.CidadeCliente = Cliente.CidadeCliente) ^ Cliente2.NomCliente <>'Carlos Pereira'

Desta forma, temos de inserir parênteses para que este bloco efetivamente funcione da forma correta, só selecionando quando as três condições forem satisfeitas.

σ CLIENTE2.RUACLIENTE = CLIENTE.RUACLIENTE ^ CLIENTE2.CIDADECLIENTE= CLIENTE.CIDADECLIENTE ^ CLIENTE2.NOMCLIENTE <>'CARLOS PEREIRA' (Resultado2)
--

π CLIENTE2.NOMCLIENTE,CLIENTE2.RUACLIENTE,CLIENTE2.CIDADECLIENTE (
σ CLIENTE2.RUACLIENTE = CLIENTE.RUACLIENTE ^ CLIENTE2.CIDADECLIENTE= CLIENTE.CIDADECLIENTE ^ CLIENTE2.NOMCLIENTE <>'CARLOS PEREIRA' (
(σ NOMCLIENTE=" CARLOS PEREIRA" (CLIENTE)) \times (ρ Cliente2 (CLIENTE))
)
)

A tabela resultante no final das operações seria a apresentada a seguir:

NomCliente	RuaCliente	CidadeCliente
Luis Paulo Souza	Rua B	Niterói

Sequência de tabelas do resultado da operação:

Resultado1 (Seleção e Projeção)

RuaCliente	CidadeCliente
Rua B	Niterói

Resultado2 (Produto Cartesiano)

RuaCliente	CidadeCliente	CodCliente	NomCliente	RuaCliente	CidadeCliente
Rua B	Niterói	1	Luis Sampaio	Rua A	Rio de Janeiro
Rua B	Niterói	2	Carlos Pereira	Rua B	Niterói
Rua B	Niterói	3	Jose Alves	Rua C	Rio de Janeiro
Rua B	Niterói	4	Luis Paulo Souza	Rua B	Niterói

Resultado3 (Seleção com Duas Condições)

RuaCliente	CidadeCliente	CodCliente	NomCliente	RuaCliente	CidadeCliente
Rua B	Niterói	4	Luis Paulo Souza	Rua B	Niterói

Resultado4 (Projeção)

NomCliente	RuaCliente	CidadeCliente
Luis Paulo Souza	Rua B	Niterói

Esperamos que com este exemplo detalhado você tenha condições de compreender completamente o jogo de sequência de operações e montagem de consultas com as operações relacionais até o momento apresentadas.

Observe que com apenas três operações de álgebra relacional vistas até este ponto do livro foi possível extrair um grande conjunto de consultas de um banco de dados.

- ▶ O nome de todos os funcionários.
- ▶ O nome e o sexo de todos os funcionários.
- ▶ O nome e data de admissão dos funcionários de sexo masculino.
- ▶ O nome e a data de admissão de funcionários que são auxiliares de cobrança.
- ▶ O nome e a dívida dos clientes de um bar.
- ▶ O nome, a rua e a cidade de clientes que moram na mesma rua e cidade de um cliente específico.

Isso que só vimos três operações relacionais até o momento e ainda não exploramos todo o potencial delas.

Vamos adiante!

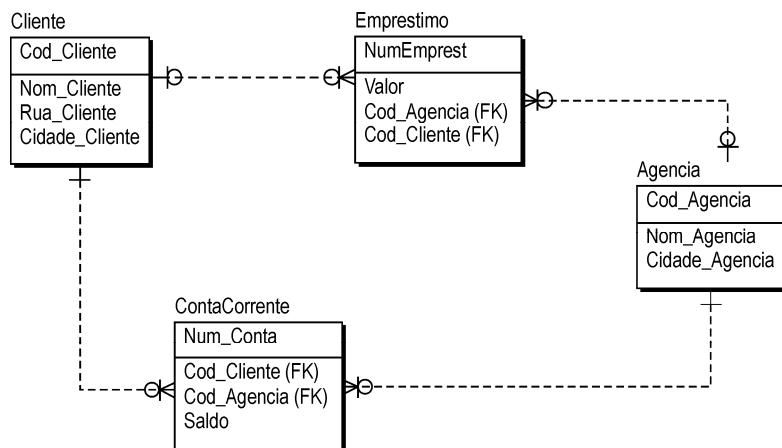
Agora vamos conhecer as operações binárias e que exigem o que é denominado de tabelas de união compatíveis. Somente podem ser utilizadas com duas tabelas que possuam a mesma estrutura de dados, ou seja, mesmo conjunto de colunas e dispostas na mesma ordem.

Para este estudo vamos utilizar as tabelas seguintes, apresentadas em sua estrutura, modelo de dados e exemplo de conteúdo para auxiliar no entendimento.

Estruturas de Dados

Tabela Cliente	CodCliente, NomCliente, RuaCliente, CidadeCliente
Tabela Agência	CodAgencia, NomAgencia, CidadeAgencia
Tabela Conta-Corrente	CodAgencia, NumConta, CodCliente, Saldo
Tabela Empréstimo	CodAgencia, CodCliente, NumEmprest, Valor

Modelo de Dados



Tabelas Exemplo

Cliente

CodCliente	NomCliente	RuaCliente	CidadeCliente
1	Luis Sampaio	Rua A	Rio de Janeiro
2	Carlos Pereira	Rua B	Niterói
3	Jose Alves	Rua C	Rio de Janeiro
4	Luis Paulo Souza	Rua B	Niterói

Agência

CodAgencia	NomAgencia	CidadeAgencia
1	Rio Branco	Rio de Janeiro
2	Icarai	Niterói
3	Leblon	Rio de Janeiro
4	Ipanema	Rio de Janeiro

Conta Corrente

CodAgencia	NumConta	CodCliente	Saldo
1	256589	1	1200,00
3	328941	1	845,00
4	749621	3	512,00
2	856200	2	2650,00
3	454501	4	800,00
2	468952	3	6920,00

Empréstimo

CodAgencia	CodCliente	NumEmprest	Valor
1	1	902230	500,00
3	1	902231	1500,00
4	2	902240	1200,00
2	3	902289	3000,00
3	1	902255	850,00
1	3	902299	700,00
4	3	902212	400,00

Estes empréstimos estão um pouco miseráveis, mas a realidade é que existem casos assim ainda hoje em dia.

Operação de União (Binária)

Essa operação requer como operandos tabelas de união compatíveis.

Produz como resultado uma tabela que contém todas as linhas da primeira tabela seguida de todas as linhas da segunda tabela.

A tabela resultante possui a mesma quantidade de colunas que as tabelas originais, e tem um número de linhas que é no máximo igual à soma das linhas das tabelas fornecidas como operandos, já que **as linhas que são comuns a ambas as tabelas aparecem uma única vez no resultado**.

Ou seja, as duas tabelas devem ter o mesmo número de atributos e os domínios dos atributos correspondentes (mesma coluna) devem ser idênticos.

A operação de união é representada, como na teoria dos conjuntos, pelo símbolo \cup .

A sintaxe geral é:

Tabela 1 \cup Tabela 2

Suponha que quiséssemos saber o nome de todas as pessoas que possuem CONTA OU EMPRÉSTIMO numa determinada agência. Observe que usamos um operador lógico OU.

Com os recursos que temos até agora não seria possível conseguirmos tal informação, pois as operações vistas até o momento não fornecem subsídios para executar tal consulta.

Para resolver esta situação, devemos fazer a união de todos que possuem conta nessa agência com todos que possuem empréstimos nessa mesma agência, por exemplo, a agência 3 (CodAgencia=3).

Como fazer isso?

O primeiro passo é determinar quem possui conta na agência especificada.

$$\pi_{\text{NOMCLIENTE}} (\sigma_{\text{CODAGENCIA} = '3'} \wedge \text{CONTACORRENTE.CODCLIENTE} = \text{CLIENTE.CODCLIENTE} \\ (\text{CLIENTE} \times \text{CONTA}))$$

Devemos agora saber quem tem empréstimos na agência 3 (CodAgencia=3):

$$\pi_{\text{NOMCLIENTE}} (\sigma_{\text{CODAGENCIA} = '3'} \wedge \text{CONTACORRENTE.CODCLIENTE} = \text{CLIENTE.CODCLIENTE} \\ (\text{EMPRESTIMO} \times \text{CLIENTE}))$$

Vamos olhar as duas operações e o seu resultado após cada uma delas:

Seleção em Cliente X Conta

CodCliente	NomCliente	RuaCliente	CidadeCliente	CodAgencia	NumConta	CodCliente	Saldo
1	Luis Sampaio	Rua A	Rio de Janeiro	3	328941	1	845,00
4	Luis Paulo Souza	Rua B	Niterói	3	454501	4	800,00

Com a projeção informada temos como resultado a tabela seguinte:

NomCliente
Luis Sampaio
Luis Paulo Souza

No segundo caso temos como resultado da seleção sobre o produto cartesiano de empréstimos e clientes:

CodCliente	NomCliente	RuaCliente	CidadeCliente	CodAgencia	CodCliente	NumEmprest	Valor
1	Luis Sampaio	Rua A	Rio de Janeiro	3	1	902231	1500,00
1	Luis Sampaio	Rua A	Rio de Janeiro	3	1	902255	850,00

Projetando, temos:

NomCliente
Luis Sampaio
Luis Sampaio

Coincidemente o nosso exemplo teve resultados idênticos.

Para responder à consulta desejada (o nome de todas as pessoas que possuem CONTA ou EMPRÉSTIMO na agência 3), realizamos então a união dos dois resultados:

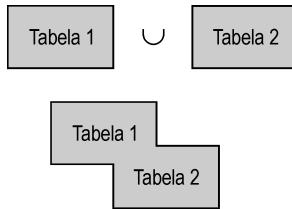
$$\pi_{\text{NomCliente}} (\sigma_{\text{CODAGENCIA} = '3'} \wedge \text{CONTACORRENTE.CODCLIENTE} = \text{CLIENTE.CODCLIENTE}$$
$$(\text{CLIENTE} \times \text{CONTA}))$$
$$\cup$$
$$\pi_{\text{NomCliente}} (\sigma_{\text{CODAGENCIA} = '3'} \wedge \text{EMPRESTIMO.CODCLIENTE} = \text{CLIENTE.CODCLIENTE}$$
$$(\text{EMPRESTIMO} \times \text{CLIENTE}))$$

E temos a tabela resultado final:

NomCliente
Luis Sampaio
Luis Paulo Souza

Pois Luis Sampaio **ou tem Conta ou tem Empréstimo** na agência 3 e Luis Paulo Souza igualmente **ou tem Conta ou tem Empréstimo** na agência 3.

Sem esquecer um certo grafismo tradicional, esta operação é ilustrada graficamente da seguinte forma:



Para uma operação **união Tabela 1 \cup Tabela 2** ser válida, é preciso que condições sejam cumpridas:

- ▶ As relações r e s precisam ter a mesma paridade, isto é, elas precisam ter o mesmo número de atributos.
- ▶ Os domínios do i-ésimo atributo de r e do i-ésimo atributo de s devem ser os mesmos.

- A disposição dos atributos nas duas tabelas deve ser a mesma, ou seja, não podemos ter os mesmos atributos nas duas tabelas e sequências diferentes. Observe o exemplo seguinte:

Cliente

CodCliente	NomCliente	RuaCliente	CidadeCliente
1	Luis Sampaio	Rua A	Rio de Janeiro
2	Carlos Pereira	Rua B	Niterói
3	Jose Alves	Rua C	Rio de Janeiro
4	Luis Paulo Souza	Rua B	Niterói

Cliente

CodCliente	RuaCliente	NomCliente	CidadeCliente
1	Rua A	Luis Sampaio	Rio de Janeiro
2	Rua B	Carlos Pereira	Niterói
3	Rua C	Jose Alves	Rio de Janeiro
4	Rua B	Luis Paulo Souza	Niterói

Estas duas tabelas não são de uniões compatíveis. Os atributos não estão na mesma ordem nas tabelas.

Operação de Diferença (Binária)

A operação de diferença de tabelas permite encontrar linhas que estão em uma tabela, mas não estão em outra.

A expressão Tabela 1 – Tabela 2 resulta uma tabela que contém todas as linhas que estão na tabela 1 e não estão na tabela 2.

O formato geral é:

Tabela 1 – Tabela 2

Utilizando o mesmo banco de dados usado para a operação de união, vejamos a consulta:

Encontrar todos os clientes que possuam uma conta, mas não tenham um empréstimo na agência 3.

$\pi_{\text{NOMCLIENTE}} (\sigma_{\text{CODAGENCIA} = '3'} \wedge \text{CONTACORRENTE.CODCLIENTE} = \text{CLIENTE.CODCLIENTE}$
 $(\text{CLIENTE} \times \text{CONTA}))$

$\pi_{\text{NOMCLIENTE}} (\sigma_{\text{CODAGENCIA} = '3'} \wedge \text{EMPRESTIMO.CODCLIENTE} = \text{CLIENTE.CODCLIENTE}$
 $(\text{EMPRESTIMO} \times \text{CLIENTE}))$

O primeiro bloco resulta a tabela:

NomCliente
Luis Sampaio
Luis Paulo Souza

Que são os clientes que possuem conta na agência 3.

O segundo bloco resulta a tabela:

NomCliente
Luis Sampaio
Luis Sampaio

Que são os clientes que possuem empréstimos na agência 3.

Um simples olhar nas duas tabelas e já sabemos a resposta de quem tem conta, mas não tem empréstimo na agência 3, e o resultado é a tabela seguinte:

NomCliente
Luis Paulo Souza

Que é o cliente que tem conta, mas não tem empréstimo nessa agência específica.

Muito importante salientar que a operação de diferença não é comutativa, ou seja, a ordem dos operandos afeta o seu resultado.

Então:

Tabela 1 – Tabela 2

não é a mesma coisa que

Tabela 2 – Tabela 1

Se invertêssemos a ordem dos dois blocos no exemplo anterior, teríamos como resultado uma tabela que não responderia corretamente à nossa consulta:

NomCliente

Pois retornaria uma tabela vazia como resultado.

Elimina-se o elemento comum na tabela 1 (Luis Sampaio) e sobra o Luis Paulo Souza:

Tabela 1	-	Tabela 2	-	Resultado								
<table border="1"> <thead> <tr><th>NomCliente</th></tr> </thead> <tbody> <tr><td>Luis Sampaio</td></tr> <tr><td>Luis Paulo Souza</td></tr> </tbody> </table>	NomCliente	Luis Sampaio	Luis Paulo Souza	-	<table border="1"> <thead> <tr><th>NomCliente</th></tr> </thead> <tbody> <tr><td>Luis Sampaio</td></tr> <tr><td>Luis Sampaio</td></tr> </tbody> </table>	NomCliente	Luis Sampaio	Luis Sampaio	-	<table border="1"> <thead> <tr><th>NomCliente</th></tr> </thead> <tbody> <tr><td>Luis Paulo Souza</td></tr> </tbody> </table>	NomCliente	Luis Paulo Souza
NomCliente												
Luis Sampaio												
Luis Paulo Souza												
NomCliente												
Luis Sampaio												
Luis Sampaio												
NomCliente												
Luis Paulo Souza												

Elimina-se o elemento comum na tabela 1 (Luis Sampaio) e não sobra nada.

Tabela 2	-	Tabela 1	-	Resultado							
<table border="1"> <thead> <tr><th>NomCliente</th></tr> </thead> <tbody> <tr><td>Luis Sampaio</td></tr> <tr><td>Luis Sampaio</td></tr> </tbody> </table>	NomCliente	Luis Sampaio	Luis Sampaio	-	<table border="1"> <thead> <tr><th>NomCliente</th></tr> </thead> <tbody> <tr><td>Luis Sampaio</td></tr> <tr><td>Luis Paulo Souza</td></tr> </tbody> </table>	NomCliente	Luis Sampaio	Luis Paulo Souza	-	<table border="1"> <thead> <tr><th>NomCliente</th></tr> </thead> </table>	NomCliente
NomCliente											
Luis Sampaio											
Luis Sampaio											
NomCliente											
Luis Sampaio											
Luis Paulo Souza											
NomCliente											

Vamos agora a um exemplo muito interessante, voltando ao nosso banco de dados do botequim, com os clientes e as contas penduradas.

Veja em seguida as tabelas para você lembrar.

CodCliente	NomCliente	CodCliente	Valor
1	Luis Sampaio	1	50,00
2	Carlos Pereira	2	43,00
3	Jose Alves	3	28,00
4	Luis Paulo Souza	4	15,00

Queremos agora saber qual a maior conta pendurada no botequim.

Vamos utilizar as operações já estudadas até este ponto do livro.

Temos de descobrir os menores valores de conta. Vamos usar a tabela conta em produto cartesiano com ela mesma, utilizando o operador de renomear.

$$\pi_{VALOR} (\sigma_{VALOR < FIADO.VALOR} (CONTA \times \rho_{FIADO}(CONTA)))$$

Com isso obtemos os menores valores de conta, senão vejamos a resolução com as tabelas:

Conta - Fiado

CodCliente	Valor	CodCliente	Valor
1	50,00	1	50,00
1	50,00	2	43,00
1	50,00	3	28,00
1	50,00	4	15,00
2	43,00	1	50,00
2	43,00	2	43,00
2	43,00	3	28,00
2	43,00	4	15,00
3	28,00	1	50,00
3	28,00	2	43,00
3	28,00	3	28,00
3	28,00	4	15,00
4	15,00	1	50,00
4	15,00	2	43,00
4	15,00	3	28,00
4	15,00	4	15,00

As linhas sombreadas representam as que satisfazem a condição de:

Valor < Fiado.valor

Logo, a tabela resultante será:

Menores Valores

Valor
43,00
28,00
28,00
15,00
15,00
15,00

Agora fazemos a diferença entre conta e este resultado:

$\pi \text{ VALOR} (\text{CONTA} - \pi \text{ VALOR} (\sigma \text{ VALOR} < \text{FIADO.VALOR} (\text{CONTA} \times p \text{ FIADO}(\text{CONTA})))$

A ilustração da operação facilita o entendimento.

CodCliente	Valor		Valor		Valor
1	50,00	-	43,00	=	50,00
2	43,00		28,00		
3	28,00		28,00		
4	15,00		15,00		
			15,00		
			15,00		

Para simplificar os comandos, vamos aprender outro operador que permite assinalar o resultado de uma operação a uma tabela.

Esta é outra forma de renomear uma tabela.

Realizamos através do símbolo \leftarrow , assim como podemos atribuir a outra tabela o resultado de uma consulta, ou de um bloco de uma consulta.

Neste último exemplo poderíamos realizá-lo com a seguinte sintaxe passo a passo.

1º passo

Menores $\leftarrow \pi_{\text{VALOR}} (\sigma_{\text{VALOR} < \text{FIADO.VALOR}} (\text{CONTA} \times p \text{ Fiado}(\text{CONTA})))$

2º passo

$\pi_{\text{VALOR}} (\text{CONTA} - \text{Menores})$
--

Operação de Intersecção de Tabelas

Esta é uma operação adicional que produz como resultado uma tabela que contém, sem repetições, todos os elementos que são comuns às duas tabelas fornecidas como operandos.

As tabelas devem ser neste caso igualmente união-compatíveis. É representado pelo símbolo \cap .

Sua forma de sintaxe básica é:

Tabela A \cap Tabela B

Interessante é que o mesmo resultado da operação de intersecção pode ser obtido realizando uma combinação de diferenças entre tabelas:

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

Ou ainda uma combinação de união e diferenças:

$$= (\text{Tabela A} \cup \text{Tabela B}) - (\text{Tabela A} - \text{Tabela B}) - (\text{Tabela B} - \text{Tabela A})$$

ou

$$= (\text{Tabela A} \cup \text{Tabela B}) - ((\text{Tabela A} - \text{Tabela B}) \cup (\text{Tabela B} - \text{Tabela A}))$$

Parece complicado, mas se você tiver o trabalho de exercitar com tabelas estas opções, vai concluir igualmente pela montagem dos comandos.

Vamos ver então uma consulta com a utilização de intersecção de tabelas. No caso das tabelas de contas-correntes, empréstimos, clientes e agência.

Agora vamos saber o nome de todos os clientes que têm empréstimo e conta-corrente na agência 3.

Antes queríamos uma consulta com o operador lógico OU, agora desejamos com o operador lógico E.

Novamente o primeiro passo é determinar quem possui conta em agência especificada, a agência 3.

$$\begin{aligned} \text{ComConta} &\leftarrow \pi_{\text{NOMCLIENTE}} (\sigma_{\text{CODAGENCIA} = '3'} \\ &\wedge \text{CONTACORRENTE.CODCLIENTE} = \text{CLIENTE.CODCLIENTE} (\text{CLIENTE} \times \text{CONTA})) \end{aligned}$$

Devemos agora saber quem tem empréstimos na agência 3 (CodAgencia=3).

$$\begin{aligned} \text{ComEmprestimo} &\leftarrow \pi_{\text{NOMCLIENTE}} (\sigma_{\text{CODAGENCIA} = '3'} \\ &\wedge \text{EMPRESTIMO.CODCLIENTE} = \text{CLIENTE.CODCLIENTE} (\text{EMPRÉSTIMO} \times \text{CLIENTE})) \end{aligned}$$

Para obtermos quem tem conta e empréstimo nessa agência, usamos a intersecção dos dois resultados com o operador para assinalar, renomear os resultados anteriores.

$$\pi_{\text{NOMCLIENTE}} (\text{ComConta} \cap \text{ComEmprestimo})$$

Vamos olhar as duas tabelas resultado e verificar as linhas comuns:

ComConta - Cliente × Conta

CodCliente	NomCliente	RuaCliente	CidadeCliente	CodAgencia	NumConta	CodCliente	Saldo
1	Luis Sampaio	Rua A	Rio de Janeiro	3	328941	1	845,00
4	Luis Paulo Souza	Rua B	Niterói	3	454501	4	800,00

Resultado1

NomCliente
Luis Sampaio
Luis Paulo Souza

ComEmprestimo - Cliente × Empréstimo

CodCliente	NomCliente	RuaCliente	CidadeCliente	CodAgencia	CodCliente	NumEmprest	Valor
1	Luis Sampaio	Rua A	Rio de Janeiro	3	1	902231	1500,00
1	Luis Sampaio	Rua A	Rio de Janeiro	3	1	902255	850,00

Resultado2

NomCliente
Luis Sampaio
Luis Sampaio

As linhas comuns entre as duas tabelas estão sombreadas, destacadas; logo, o resultado final destacando-se as linhas comuns será:

NomCliente
Luis Sampaio

Vamos estudar outra operação que efetiva a junção das tabelas baseada em atributos comuns às duas tabelas.

Operação de Junção

Esta é uma operação que realiza a combinação das linhas de uma tabela com as linhas correspondentes de outra tabela, sendo correspondente a uma operação de seleção através dos atributos do relacionamento entre essas duas mesmas tabelas, aplicada essa seleção no produto cartesiano dessas mesmas tabelas.

A operação de junção foi criada justamente porque esse tipo de combinação de tabelas é de uso muito comum, pois os modelos de dados Entidade-Relacionamento estabelecem relacionamentos através de colunas comuns entre as tabelas, normalmente as chaves estrangeiras de um relacionamento, facilitando com isso a escrita de expressões.

O formato geral da operação de junção é:

Tabela A  Tabela B

Em que **A.chave1 = B.chave2** é o predicado da seleção, ou seja, uma seleção do produto cartesiano das duas tabelas:

$$\sigma A.chave1 = B.chave2 (A \times B)$$

Na junção a tabela resultante tem todas (sem exceção) as colunas da primeira tabela e todas da segunda tabela, pois é derivada de um produto cartesiano.

Isso faz com que os valores dos campos utilizados como critério para a correspondência entre as linhas apareçam duplicados, já que um vem da primeira tabela e outro da sua existência na segunda tabela.

Existe uma tipologia da junção, chamada *junção natural*, que fornece mesmo resultado, mas sem essa repetição de valores: uma das colunas correspondentes aos atributos de relacionamento é descartada.

A junção natural utiliza como predicado de seleção as colunas comuns às duas tabelas, com um detalhe sem especificação de predicado, como na junção normal.

A junção natural, pelo fato de combinar sempre todos os atributos idênticos de duas relações, não deve ser empregada quando se deseja associar duas relações apenas por um ou alguns dos seus atributos idênticos.

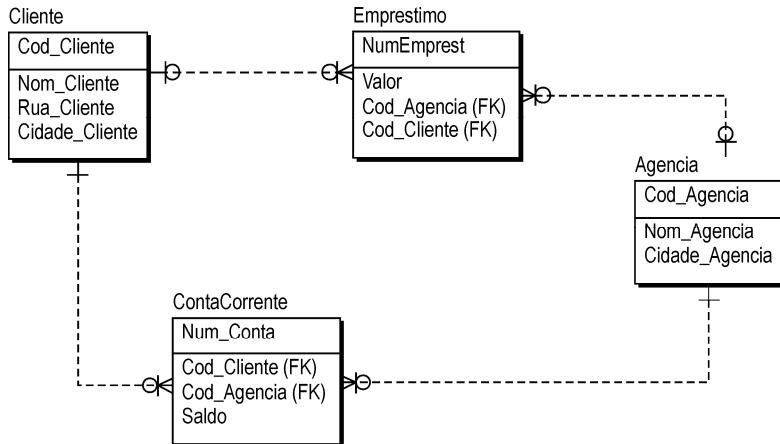
Logo, analisando um modelo de dados, se as duas tabelas às quais se deseja aplicar uma operação de junção possuírem mais de uma coluna em comum, os resultados da aplicação da junção natural são imprevisíveis, e certamente não vão responder adequadamente à nossa consulta.

A prática leva-nos sempre a optar por utilizar uma operação de junção com especificação de predicados, que a torna segura para atingirmos os resultados desejados de consulta.

Vamos visualizar as duas utilizações de junção em um dos nossos modelos de tabelas, o bancário.

Para que o leitor não tenha que voltar páginas, vamos reapresentar a estrutura de dados, modelo de dados e a simulação das tabelas.

Tabela Cliente	CodCliente, NomCliente, RuaCliente, CidadeCliente	Tabela Conta-Corrente	CodAgencia, NumConta, CodCliente, Saldo
Tabela Agência	CodAgencia, NomAgencia, CidadeAgencia	Tabela Empréstimo	CodAgencia, CodCliente, NumEmprest, Valor



Tabelas

Cliente

CodCliente	NomCliente	RuaCliente	CidadeCliente
1	Luis Sampaio	Rua A	Rio de Janeiro
2	Carlos Pereira	Rua B	Niterói
3	Jose Alves	Rua C	Rio de Janeiro
4	Luis Paulo Souza	Rua B	Niterói

Agência

CodAgencia	NomAgencia	CidadeAgencia
1	Rio Branco	Rio de Janeiro
2	Icarai	Niterói
3	Leblon	Rio de Janeiro
4	Ipanema	Rio de Janeiro

Conta Corrente

CodAgencia	NumConta	CodCliente	Saldo
1	256589	1	1200,00
3	328941	1	845,00
4	749621	3	512,00
2	856200	2	2650,00
3	454501	4	800,00
2	468952	3	6920,00

Empréstimo

CodAgencia	CodCliente	NumEmprest	Valor
1	1	902230	500,00
3	1	902231	1500,00
4	2	902240	1200,00
2	3	902289	3000,00
3	1	902255	850,00
1	3	902299	700,00
4	3	902212	400,00

Exemplo

- Desejamos o nome de todos os cliente que têm empréstimos e a cidade em que vivem.

O primeiro passo é realizarmos uma junção das duas tabelas.

CLIENTE \bowtie CLIENTE.CODCLIENTE=EMPRÉSTIMO.CODCLIENTE EMPRÉSTIMO

Ou com junção natural, uma vez que só possuem uma coluna em comum.

CLIENTE \bowtie EMPRÉSTIMO

Como resultado da junção teremos a tabela seguinte:

CodCliente	NomCliente	RuaCliente	CidadeCliente	CodAgencia	CodCliente	NumEmprest	Valor
1	Luis Sampaio	Rua A	Rio de Janeiro	1	1	902230	500,00
1	Luis Sampaio	Rua A	Rio de Janeiro	3	1	902255	850,00
1	Luis Sampaio	Rua A	Rio de Janeiro	3	1	902231	1500,00
2	Carlos Pereira	Rua B	Niterói	4	2	902240	1200,00
3	Jose Alves	Rua C	Rio de Janeiro	2	3	902289	3000,00
3	Jose Alves	Rua C	Rio de Janeiro	1	3	902299	700,00
3	Jose Alves	Rua C	Rio de Janeiro	4	3	902212	400,00

Deste resultado devemos projetar as colunas desejadas na consulta:

π NOMCLIENTE, CIDADECLIENTE (RESULTADO)

Realizando agora a consolidação das operações em uma única operação, temos:

π NOMCLIENTE, CIDADECLIENTE (CLIENTE \bowtie CLIENTE.CODCLIENTE=EMPRÉSTIMO.CODCLIENTE EMPRÉSTIMO)

Ou com junção natural:

π NOMCLIENTE, CIDADECLIENTE (CLIENTE \bowtie EMPRÉSTIMO)

Vamos a outro exemplo:

Encontre todos os clientes que têm empréstimos e moram em "Niterói".

Usando junção natural, temos:

π NOMCLIENTE, CIDADECLIENTE (σ CIDADECLIENTE = 'NITERÓI' (CLIENTE \bowtie EMPRÉSTIMO))

Colocamos a seleção da coluna CidadeCliente='Niterói' sobre a junção natural de Cliente e Empréstimo.

Tabela resultante:

CodCliente	NomCliente	RuaCliente	CidadeCliente	CodAgencia	NumEmprest	Valor
2	Carlos Pereira	Rua B	Niterói	4	902240	1200,00

Observe que este resultado, além de ser diferente do anterior pelo fato de existir uma seleção para a cidade de Niterói, possui somente uma coluna CodCliente, por ser uma junção natural, a qual elimina uma das colunas comuns no resultado.

Agora vamos complicar um pouquinho:

Encontre os nomes de todos os clientes que têm conta nas agências situadas em "Niterói".

Você vai se perguntar: opa, neste caso temos três tabelas envolvidas: cliente, conta e agência.

Lembrando a conceituação de junção:

"Esta é uma operação que realiza a combinação das linhas de uma tabela com as linhas correspondentes de outra tabela, sendo correspondente a uma operação de seleção através dos atributos do relacionamento entre essas duas mesmas tabelas, aplicada essa seleção no produto cartesiano dessas mesmas tabelas."

Como então trabalhar com três tabelas?

Sem problema, se considerarmos junção natural.

```
 $\pi_{\text{NOMCLIENTE}, \text{NUMCONTA}} (\sigma_{\text{CIDADEAGENCIA} = "NITERÓI"} (\text{CLIENTE} \bowtie \text{CONTA} \bowtie \text{AGÊNCIA}))$ 
```

Se quisermos abrir em junções com predicado, o comando somente ficará mais longo, nada mais.

```
 $\pi_{\text{NOMCLIENTE}, \text{NUMCONTA}} (\sigma_{\text{CIDADEAGENCIA} = "NITERÓI"} ((\text{CLIENTE} \bowtie \text{CLIENTE.CODCLIENTE} = \text{CONTA.CODCLIENTE} \bowtie \text{CONTA}) \bowtie \text{CONTA.CODAGENCIA} = \text{AGÊNCIA.CODAGENCIA}))$ 
```

Vamos analisar as operações sobre os resultados das tabelas para melhor entender o escopo das operações:

Cliente \bowtie Conta

CodCliente	NomCliente	RuaCliente	CidadeCliente	CodAgencia	NumConta	CodCliente	Saldo
1	Luis Sampaio	Rua A	Rio de Janeiro	1	256589	1	1200,00
1	Luis Sampaio	Rua A	Rio de Janeiro	3	328941	1	845,00
2	Carlos Pereira	Rua B	Niterói	2	856200	2	2650,00
3	Jose Alves	Rua C	Rio de Janeiro	4	749621	3	512,00
3	Jose Alves	Rua C	Rio de Janeiro	2	468952	3	6920,00
4	Luis Paulo Souza	Rua B	Niterói	3	454501	4	800,00

(Este resultado) |×| Agência

CodCliente	NomCliente	RuaCliente	CidadeCliente	CodAgencia	NumConta	Saldo	NomAgencia	CidadeAgencia	CidadeAgencia
1	Luis Sampaio	Rua A	Rio de Janeiro	1	256589	1200,00	Rio Branco	Rio de Janeiro	Rio de Janeiro
1	Luis Sampaio	Rua A	Rio de Janeiro	3	328941	845,00	Leblon	Rio de Janeiro	Rio de Janeiro
2	Carlos Pereira	Rua B	Niterói	2	856200	2650,00	Icarai	Niterói	Niterói
3	Jose Alves	Rua C	Rio de Janeiro	4	749621	512,00	Ipanema	Rio de Janeiro	Rio de Janeiro
3	Jose Alves	Rua C	Rio de Janeiro	2	468952	6920,00	Icarai	Niterói	Niterói
4	Luis Paulo	Rua C	Niterói	3	454501	800,00	Leblon	Rio de Janeiro	Rio de Janeiro

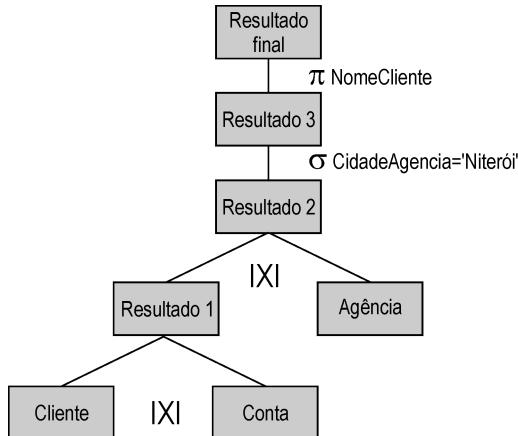
A seleção desta tabela retorna:

CodCliente	NomCliente	RuaCliente	CidadeCliente	CodAgencia	NumConta	Saldo	NomAgencia	CidadeAgencia	CidadeAgencia
2	Carlos Pereira	Rua B	Niterói	2	856200	2650,00	Icarai	Niterói	Niterói
3	Jose Alves	Rua C	Rio de Janeiro	2	468952	6920,00	Icarai	Niterói	Niterói

E a projeção do comando fornece o resultado final:

NomCliente
Carlos Pereira
Jose Alves

Para auxiliar a visualização da sequência de operações realizadas, apresentamos em seguida o diagrama da operação completa:



Vamos acompanhar a sequência e estudar a última das operações relacionais desse conjunto que estamos apresentando, que é a operação de divisão.

Operação de Divisão

A operação divisão, representada por \div , serve para consultas como as do tipo para todos. Vamos exemplificar esta situação.

Para que o exemplo possa ser viável dentro do esquema bancário que temos utilizado neste capítulo, vamos aumentar os dados da tabela conta-corrente para entendermos melhor a operação.

Observe a nova tabela de conta-corrente à qual acrescentamos a linha:

4	278156	1	10000,00
---	--------	---	----------

Colocando o CodCliente de valor 1 agora com uma conta na agência 4 também.

Conta-Corrente

CodAgencia	NumConta	CodCliente	Saldo
1	256589	1	1200,00
3	328941	1	845,00
4	749621	3	512,00
2	856200	2	2650,00
3	454501	4	800,00
2	468952	3	6920,00
4	278156	1	10000,00

Suponha que desejamos encontrar todos os clientes que têm uma conta em todas as agências localizadas no Rio de Janeiro. Podemos obter todas as agências do Rio de Janeiro pela expressão:

Agência

CodAgencia	NomAgencia	CidadeAgencia
1	Rio Branco	Rio de Janeiro
2	Icarai	Niterói
3	Leblon	Rio de Janeiro
4	Ipanema	Rio de Janeiro

Sabendo que pela tabela as agências do Rio de Janeiro são as com CodAgencia de valor 1, 3 e 4, com um rápido olhar na tabela de conta-corrente já observamos que o cliente CodCliente=1 tem conta em todas as agências da cidade do Rio de Janeiro.

Mas vamos obter esta resposta utilizando o operador de divisão.

Em primeiro lugar temos de obter uma tabela com todas as agências do Rio de Janeiro:

$\text{AgenciaRio} \leftarrow \pi_{\text{CODAGENCIA}} (\sigma_{\text{CIDADEAGENCIA}=\text{"RIO DE JANEIRO"}} (\text{AGÊNCIA}))$

Obtemos a tabela seguinte:

AgenciaRio

CodAgencia
1
3
4

Assinalamos o resultado para a tabela AgenciaRio, colocamos uma tabela temporária, uma variável por assim dizer.

Vamos encontrar todos os pares NomCliente, CodAgencia referentes a um cliente possuir uma conta em uma agência realizando a junção entre conta e cliente:

$\text{ClienteConta} \leftarrow \pi_{\text{NOMCLIENTE}, \text{CODAGENCIA}} (\text{CONTA} \bowtie \text{CLIENTE})$

ClienteConta

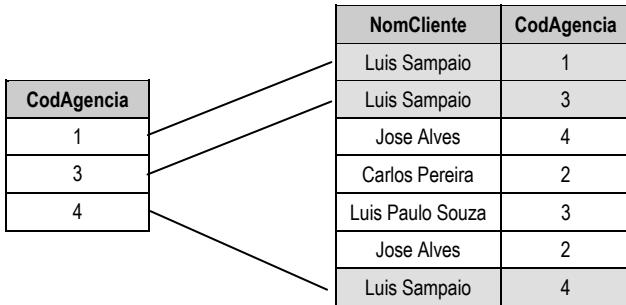
NomCliente	CodAgencia
Luis Sampaio	1
Luis Sampaio	3
Jose Alves	4
Carlos Pereira	2
Luis Paulo Souza	3
Jose Alves	2
Luis Sampaio	4

Agora precisamos encontrar clientes que apareçam em **ClienteConta** com cada nome de agência em **AgenciaRio**.

Escrevemos esta consulta no seguinte formato:

$\pi_{\text{NOMCLIENTE}, \text{CODAGENCIA}} (\text{CONTA} \bowtie \text{CLIENTE}) \div$
$\pi_{\text{CODAGENCIA}} (\sigma_{\text{CIDADEAGENCIA}=\text{"RIO DE JANEIRO"}} (\text{AGÊNCIA}))$

São procurados os conjuntos de linhas em **ClienteConta** cujos valores dos atributos comuns são iguais a todos os que aparecem em **AgenciaRio**.



O resultado final desta operação é a tabela seguinte:

NomCliente
Luis Sampaio

O operando divisor é a tabela **AgenciaRio** e o operando dividendo consiste na tabela **ClienteConta**. A tabela resultante da operação é o **Quociente** da operação.

Vistas todas as operações relacionais, vamos resolver alguns exercícios complexos de consulta para que o leitor aumente sua base de conhecimento sobre a utilização de operações de álgebra relacional e enfrente qualquer prova de conceitos com sua utilização.

Antes, porém, vamos apresentar uma tabela com as operações e seus símbolos e sintaxe para disponibilizar a você uma fonte única de consulta da simbologia utilizada.

Símbolo	Operação	Sintaxe
σ	Seleção/Restrição	σ Condição (Tabela)
π	Projeção	π Colunas (Tabela)
\cup	União	Tabela 1 \cup Tabela 2
\cap	Intersecção	Tabela 1 \cap Tabela2
$-$	Diferença	Tabela 1 $-$ Tabela 2
\times	Produto Cartesiano	Tabela 1 \times Tabela 2
\bowtie	Junção	Tabela 1 \bowtie Tabela 2
\div	Divisão	Tabela 1 \div Tabela 2
ρ	Renomeação	ρ Nome (Tabela)
\leftarrow	Atribuição	Nome \leftarrow Tabela ou Resultado Operação

Agora vamos a modelos e casos mais complexos para estudarmos soluções com álgebra relacional.

Exercícios Resolvidos e Operadores Adicionais

Considere o seguinte esquema de uma base de dados de uma transportadora aérea (em que os atributos chave primária se encontram sublinhados):

- ▶ Marca ({Marca, Lugares, Autonomia})
- ▶ Piloto ({NomeP, Endereco, Local, Idade})
- ▶ Aviao ({Matr, NomeA, Marca})
- ▶ Voo ({Num, Matr, Data, Hora, De, Para, NomeP})

A seguir vamos simular dados nas tabelas apresentadas:

Marca

Marca	Lugares	Autonomia
Boeing 747	180	10.000
Boeing 737-300	120	8000
Boeing 737-500	118	8000
Airbus 300	150	8500
Airbus 200	110	7500

Piloto

NomeP	Endereço	Local	Idade
Joao	E-1	Local-1	34
Pedro	E-2	Local-2	36
Inacio	E-3	Local-3	42
Jose	E-4	Local-4	28
Carlos	E-5	Local-5	31
Adao	E-6	Local-6	37
Karl	E-7	Local-7	48
Daniel	E-8	Local-8	45

Avião

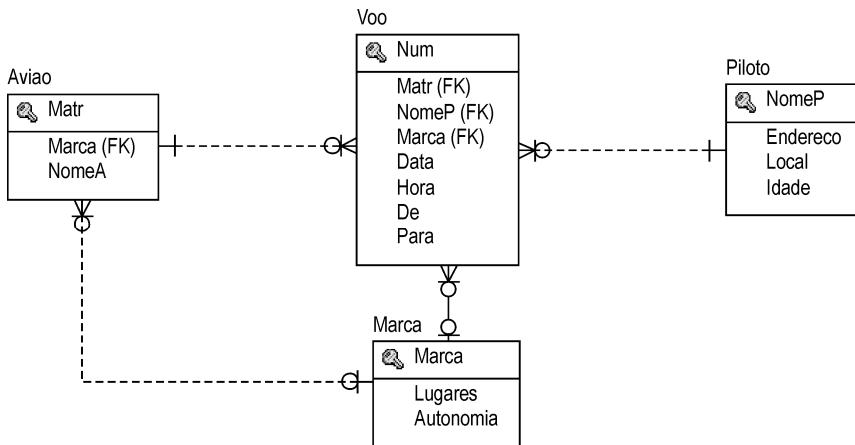
Matr	NomeA	Marca
12	PPT-XP	Boeing 747
15	PPT-CD	Boeing 747
20	PPT-AS	Boeing 737-300
5	PPT-TD	Boeing 737-500
3	PPT-FR	Boeing 737-300
40	PPT-XJ	Airbus 300
32	PPT-AM	Airbus 200

Voo

Num	Matr	Data	Hora	De	Para	NomeP
100	15	15/12/2002	08:00	SDU	CGH	Pedro
101	12	16/12/2002	09:00	SDU	CGH	Pedro
102	5	16/12/2002	10:05	SDU	BSB	Inacio
103	12	16/12/2002	11:00	CGH	SDU	Pedro
104	5	16/12/2002	17:42	BSB	SDU	Daniel
105	12	16/12/2002	13:00	SDU	CGH	Inacio

Num	Matr	Data	Hora	De	Para	NomeP
106	15	15/12/2002	10:00	CGH	SDU	Inacio
108	20	16/12/2002	09:12	SDU	POA	Pedro
321	40	18/12/2002	09:52	SDU	FOR	Inacio
322	32	19/12/2002	09:57	SDU	REC	Inacio
324	20	17/12/2002	11:03	POA	SDU	Pedro
456	15	15/12/2002	14:00	SDU	CGH	Daniel
457	15	15/12/2002	19:00	CGH	SDU	Daniel
4020	20	20/12/2002	09:12	SDU	POA	Inacio
4010	15	20/12/2002	21:12	SDU	BSB	Daniel
4011	3	15/12/2002	13:02	SDU	GOI	Karl
3010	3	15/12/2002	17:51	GOI	SDU	Karl
3110	5	17/12/2002	11:03	POA	SDU	Inacio
3011	12	17/12/2002	08:00	CGH	SDU	Pedro

E o modelo de dados para o caso em estudo.



Vamos às questões a serem respondidas sobre esta base de dados.

1ª Questão

Quais são as matrículas dos aviões que têm autonomia superior a 7.500 quilômetros?

$$\pi_{\text{MATR}} (\sigma_{\text{MARCA}.\text{MARCA}=\text{AVIÃO}.\text{MARCA} \wedge \text{MARCA}.AUTONOMIA > 7500} (\text{MARCA} \times \text{AVIÃO}))$$

A tabela resultante do produto cartesiano (utilizamos neste caso) com a seleção de colunas comuns ($\sigma_{\text{marca}.marca}=\text{aviao}.marca$) é a seguinte:

Marca	Lugares	Autonomia	Matr	NomeA	Marca
Boeing 747	180	10.000	12	PPT-XP	Boeing 747
Boeing 747	180	10.000	15	PPT-CD	Boeing 747
Boeing 737-300	120	8000	20	PPT-AS	Boeing 737-300
Boeing 737-300	120	8000	3	PPT-FR	Boeing 737-300
Boeing 737-500	118	8000	5	PPT-TD	Boeing 737-500
Airbus 300	150	8500	40	PPT-XJ	Airbus 300
Airbus 200	110	7500	32	PPT-AM	Airbus 200

Completando a seleção com $\wedge_{\text{MARCA.AUTONOMIA} > 8000}$, temos a tabela resultado:

Marca	Lugares	Autonomia	Matr	NomeA	Marca
Boeing 747	180	10.000	12	PPT-XP	Boeing 747
Boeing 747	180	10.000	15	PPT-CD	Boeing 747
Boeing 737-300	120	8000	20	PPT-AS	Boeing 737-300
Boeing 737-300	120	8000	3	PPT-FR	Boeing 737-300
Boeing 737-500	118	8000	5	PPT-TD	Boeing 737-500
Airbus 300	150	8500	40	PPT-XJ	Airbus 300

E com a projeção final π_{MATR} , o resultado será a tabela seguinte:

Matr
12
15
20
3
5
40

2ª Questão

Quais são os nomes e endereços dos pilotos que alguma vez pilotaram um avião de marca Boeing 737?

```

 $\pi_{\text{NOMEPI, ENDEREÇO}} (\text{PILOTO} \bowtie_{\text{PILOTO.NOMEPI}=\text{NOMEPI}} \text{PILOTO})$ 

( $\pi_{\text{NOMEPI}} (\sigma_{\text{AVIAO.MARCA}=\text{"BOEING 737-300"}} \vee \sigma_{\text{AVIAO.MARCA}=\text{"BOEING 737-500"}}) \bowtie_{\text{AVIAO.MATR} = \text{VOO.MATR}} \text{AVIAO}$ )
)
)
)
```

Vamos analisar esta solução.

Em primeiro lugar temos de pensar que necessitamos descobrir em cada voo o nome do avião nele utilizado, uma vez que só temos a coluna Matr em cada linha da tabela voo.

Logo, necessitamos de uma junção da tabela avião com a tabela voo, selecionando dessa junção as linhas em que aviao.marca="Boeing 737-300" ou aviao.marca="Boeing 737-500".

Isso nos fornece como resultado sete voos que utilizaram esse tipo de aeronave:

Matr	NomeA	Marca	Num	Matr	Data	Hora	De	Para	NomeP
5	PPT-TD	Boeing 737-500	104	5	16/12/2002	17:42	BSB	SDU	Daniel
20	PPT-AS	Boeing 737-300	108	20	16/12/2002	09:12	SDU	POA	Pedro
20	PPT-AS	Boeing 737-300	324	20	17/12/2002	11:03	POA	SDU	Pedro
20	PPT-AS	Boeing 737-300	4020	20	20/12/2002	09:12	SDU	POA	Inacio
3	PPT-FR	Boeing 737-300	4011	3	15/12/2002	13:02	SDU	GOI	Karl
3	PPT-FR	Boeing 737-300	3010	3	15/12/2002	17:51	GOI	SDU	Karl
5	PPT-TD	Boeing 737-500	3110	5	17/12/2002	11:03	POA	SDU	Inacio

Resta agora obter os endereços dos pilotos assinalados em cada um desses voos.

Vamos realizar a junção da tabela piloto com o resultado obtido anteriormente (vamos utilizar uma junção especificando novamente a condição de junção) e projetar as colunas que desejamos no resultado final.

π NOME, ENDEREÇO

(PILOTO  PILOTO.NOME=NOME (Resultado anterior)

Tabelas obtidas na sequência (junção):

Matr	NomeA	Marca	Num	Matr	Data	Hora	De	Para	NomeP	NomeP	Endereço	Local	Idade
5	PPT-TD	Boeing 737-500	104	5	16/12/2002	17:42	BSB	SDU	Daniel	Daniel	E-8	Local-8	45
20	PPT-AS	Boeing 737-300	108	20	16/12/2002	09:12	SDU	POA	Pedro	Pedro	E-2	Local-2	36
20	PPT-AS	Boeing 737-300	324	20	17/12/2002	11:03	POA	SDU	Pedro	Pedro	E-2	Local-2	36
20	PPT-AS	Boeing 737-300	4020	20	20/12/2002	09:12	SDU	POA	Inácio	Inácio	E-3	Local-3	42
3	PPT-FR	Boeing 737-300	4011	3	15/12/2002	13:02	SDU	GOI	Karl	Karl	E-7	Local-7	48
3	PPT-FR	Boeing 737-300	3010	3	15/12/2002	17:51	GOI	SDU	Karl	Karl	E-7	Local-7	48
5	PPT-TD	Boeing 737-500	3110	5	17/12/2002	11:03	POA	SDU	Inácio	Inácio	E-3	Local-3	42

Com a projeção temos a tabela seguinte, que é o resultado final.

NomeP	Endereço
Daniel	E-8
Pedro	E-2
Inácio	E-3
Karl	E-7

3^a Questão

Quais são os nomes dos pilotos que já pilotaram aviões de todas as marcas existentes?

A palavra todas lembra que nesta questão é preciso realizar uma divisão. Vamos adiante.

Acompanhe a solução passo a passo:

1º passo: obter nome do piloto e avião na tabela voo.

$$\pi_{\text{MATR, NOMEP}}(\text{VOO})$$

Matr	NomeP
3	Karl
3	Karl
5	Inacio
5	Inacio
12	Inacio
12	Pedro
12	Pedro
12	Pedro
15	Daniel
15	Daniel
15	Daniel
15	Inacio
15	Pedro
20	Inacio
20	Inacio
20	Pedro
20	Pedro
32	Inacio
40	Inacio

2º passo: obter nome do piloto e marca de avião pela projeção das colunas desejadas na junção da tabela avião com o resultado do comando anterior:

$$\begin{aligned} & \pi_{\text{MARCA,NOME}} \\ & (\\ & \text{AVIAO} \bowtie \text{AVIAO.MATR=MATR} (\pi_{\text{MATR, NOME}}(\text{VOO})) \\ &) \end{aligned}$$

E já temos uma tabela com pilotos e aviões que eles pilotaram.

Marca	NomeP
Boeing 747	Daniel
Boeing 747	Daniel
Boeing 747	Daniel
Boeing 737-500	Inacio
Boeing 737-500	Inacio
Boeing 747	Inacio
Boeing 747	Inacio
Boeing 737-300	Inacio
Boeing 737-300	Inacio
Airbus 200	Inacio
Airbus 300	Inacio
Boeing 737-300	Karl
Boeing 737-300	Karl
Boeing 747	Pedro
Boeing 747	Pedro
Boeing 747	Pedro
Boeing 737-300	Pedro
Boeing 737-300	Pedro

Agora vamos obter as marcas de avião existentes pela projeção da tabela marca.

$\pi_{\text{MARCA}}(\text{MARCA})$

Marca
Boeing 747
Boeing 737-300
Boeing 737-500
Airbus 300
Airbus 200

Podemos agora dividir o resultado obtido anteriormente pelo resultado dessa projeção.

$\pi_{\text{MARCA}, \text{NOME}P}(\text{AVIÃO} \bowtie_{\text{AVIAO.MATR}=\text{MATR}} (\pi_{\text{MATR}, \text{NOME}P}(\text{VOO})))$

/

$\pi_{\text{MARCA}}(\text{MARCA})$

A pergunta que não quer calar:

A solução não poderia ser mais simples eliminando um passo?

Sim, efetivamente está de parabéns o leitor que chegar a esta conclusão.

A solução mais simples é:

$$\pi_{\text{MARCA}, \text{NOME}P} (\text{AVIÃO} \bowtie \text{AVIAO.MATR}=\text{MATR} (\text{Voo}))$$

/

$$\pi_{\text{MARCA}} (\text{MARCA})$$

Observe que na junção que estamos utilizando já obtivemos com a projeção uma tabela que possui marca e nome do piloto, dispensando um passo dos apresentados anteriormente.

O resultado final será a tabela seguinte:

NomeP
Inacio

Que é o quociente entre as duas projeções utilizadas.

4^a Questão

Quais são os aviões que, num mesmo dia, passaram mais de uma vez por um mesmo aeroporto?

Parece que complicou, meu amigo leitor?

Não muito, só que para resolver este exercício vamos aprender algumas funções adicionais existentes na álgebra relacional.

Funções em Álgebra Relacional

Algumas operações muito comuns sobre base de dados não podem ser executadas usando apenas as operações de álgebra relacional vistas até agora.

Tais operações, entretanto, são implementadas na linguagem de consulta à base de dados (SQL) fornecida por fabricantes dos principais SGBDR existentes no mercado.

O primeiro grupo de funções que não pode ser expresso usando álgebra relacional é composto por funções matemáticas de agregação executadas com uma coleção de valores de uma base de dados.

Por exemplo, calcular média e o total de salários pagos para um grupo de empregados de uma empresa.

Entre as funções de agregação mais comuns aplicadas em uma coleção de valores numéricos estão a SOMA, a MÉDIA, o MÁXIMO e o MÍNIMO, além da função CONTADOR, que é usada para contar linhas.

Cada uma destas funções pode ser aplicada em uma coleção de linhas, seja de uma tabela, seja de um resultado de operação relacional.

Considere a seguinte tabela:

Funcionário

Número	Nome	Departamento	Salário
31445	António Silva	Contabilidade	130
30442	Isabel Sousa	Armazém	240
27710	Mário Gomes	Vendas	240
27720	Maria Mendes	Vendas	300
27730	João Pereira	Vendas	130

Para responder à questão:

Qual é a soma dos salários de todos os empregados?

É necessário aplicar uma função de agregação à tabela.

As funções de agregação usuais são:

COUNT	contagem do número de linhas
SUM	cálculo da soma
AVERAGE	cálculo da média
MAXIMUM	identificação do valor máximo
MINIMUM	identificação do valor mínimo

Outra questão que se pode colocar é:

Por departamento, qual o valor máximo dos salários dos seus empregados?

A resposta a esta questão implica:

Agrupar as linhas da tabela pelo valor de alguns dos seus atributos e aplicar uma função de agregação a cada um dos grupos de linhas.

- ▶ Vamos definir o operador de agregação Σ que permite escrever estas questões.
- ▶ Sendo uma tabela A, o operador de agregação Σ tem o formato geral:

<Atributos para Agrupamento> Σ <Funções de Agregação> (Tabela A)

Então sobre a tabela exemplo de funcionários, para saber o valor máximo de salários, temos a seguinte expressão:

<Departamento> Σ <Max_{salário}> (FUNCIONÁRIO)

Que retorna uma única coluna de resultado que é o valor do maior salário:

Salário
300

E na questão da soma dos salários:

$\Sigma \text{Salário} > (\text{FUNCIONÁRIO})$

Retorna somente o valor 1040.

Por outro lado, podemos ter de utilizar operações aritméticas simples, como soma, subtração etc. que denominamos de projeção generalizada.

Projeção Generalizada

As operações de projeção ditas generalizadas estendem a lista de atributos de uma projeção de forma a permitir a utilização de operações aritméticas nessa lista de atributos.

O formato nada mais é do que colocar na lista uma expressão aritmética com ou sobre os atributos existentes na tabela que está sendo projetada.

Para exemplificar melhor, vamos apresentar a tabela seguinte:

InfoCredito

NomeCliente	LimiteCredito	SaldoCredito
João	6.000,00	700,00
Silva	2.000,00	400,00
Pedro	1.500,00	1.500,00
Cintia	2.000,00	1.750,00

Supondo que desejamos saber o quanto uma pessoa ainda pode gastar e temos a tabela info_credito que lista limites de crédito e valores de crédito já utilizados (saldo_credito).

Projeção generalizada é uma operação no seguinte formato:

$\pi_{\text{NOMECLIENTE}, \text{LIMITECREDITO} - \text{SALDOCREDITO}} (\text{INFO_CREDITO})$

Que fornece como resultado a tabela seguinte:

João	5.300,00
Silva	1.600,00
Pedro	0,00
Cintia	250,00

Apresentadas as duas extensões da álgebra relacional, vamos continuar na resolução da questão número 4:

Quais são os aviões que, num mesmo dia, passaram mais de uma vez por um mesmo aeroporto?

Neste exercício vamos precisar de dois conjuntos iguais inicialmente, para posteriormente realizarmos a sua união e obter assim a matrícula do avião, as datas e os aeroportos por onde passaram, porque devemos considerar a origem (De) e o destino (Para) como sendo aeroportos por onde passaram os aviões.

Como fazer isso?

Devemos utilizar um artifício para obter uma projeção das linhas com a coluna 'Para' modificada para aeroporto, assim como a coluna também modificada para aeroporto, e obter duas tabelas de mesma estrutura de dados para possibilitar a união das duas. Podemos utilizar o operador de rename, a letra grega rho ρ , também para dar nome às colunas do resultado de uma projeção

Para isso utiliza-se o operador de renomear, porém com o acréscimo de renomear também seus atributos:

$$\rho_{\text{VooDE}(\text{MATRICULA}, \text{DATA}, \text{AEROPORTO})}(\pi_{\text{MATRICULA}, \text{DATA}, \text{DE}(\text{VOO})})$$

Desta forma temos a projeção da tabela de voos agora como a apresentada em seguida:

VooDe

Matr	Data	Aeroporto
15	15/12/2002	SDU
12	16/12/2002	SDU
5	16/12/2002	SDU
12	16/12/2002	CGH
5	16/12/2002	BSB
12	16/12/2002	SDU
15	15/12/2002	CGH
20	16/12/2002	SDU
40	18/12/2002	SDU
32	19/12/2002	SDU
20	17/12/2002	POA
15	15/12/2002	SDU
15	15/12/2002	CGH
20	20/12/2002	SDU
15	20/12/2002	SDU
3	15/12/2002	SDU
3	15/12/2002	GOI
5	17/12/2002	POA
12	17/12/2002	CGH

Vamos aplicar a mesma operação considerando a coluna Para:

ρ VOO PARA (MATRICA, DATA, AEROPORTO) (π MATRICA, DATA, PARA (Voo))

Temos então a tabela seguinte, que é a renomeação da coluna Para.

VooPara

Matr	Data	Aeroporto
15	15/12/2002	CGH
12	16/12/2002	CGH
5	16/12/2002	BSB
12	16/12/2002	SDU
5	16/12/2002	SDU
12	16/12/2002	CGH
15	15/12/2002	SDU
20	16/12/2002	POA
40	18/12/2002	FOR
32	19/12/2002	REC
20	17/12/2002	SDU
15	15/12/2002	CGH
15	15/12/2002	SDU
20	20/12/2002	POA
15	20/12/2002	BSB
3	15/12/2002	GOI
3	15/12/2002	SDU
5	17/12/2002	SDU
12	17/12/2002	SDU

Fazemos agora a união das duas tabelas resultantes de projeção.

A próxima página vai ser dedicada a apresentar esta enorme tabela resultante:

TodosVoos \leftarrow VooPARA \cup VooDE

TodosVoos

Matr	Data	Aeroporto
15	15/12/2002	SDU
12	16/12/2002	SDU
5	16/12/2002	SDU
12	16/12/2002	CGH

Matr	Data	Aeroporto
5	16/12/2002	BSB
12	16/12/2002	SDU
15	15/12/2002	CGH
20	16/12/2002	SDU
40	18/12/2002	SDU
32	19/12/2002	SDU
20	17/12/2002	POA
15	15/12/2002	SDU
15	15/12/2002	CGH
20	20/12/2002	SDU
15	20/12/2002	SDU
3	15/12/2002	SDU
3	15/12/2002	GOI
5	17/12/2002	POA
12	17/12/2002	CGH
15	15/12/2002	CGH
12	16/12/2002	CGH
5	16/12/2002	BSB
12	16/12/2002	SDU
5	16/12/2002	SDU
12	16/12/2002	CGH
15	15/12/2002	SDU
20	16/12/2002	POA
40	18/12/2002	FOR
32	19/12/2002	REC
20	17/12/2002	SDU
15	15/12/2002	CGH
15	15/12/2002	SDU
20	20/12/2002	POA
15	20/12/2002	BSB
3	15/12/2002	GOI
3	15/12/2002	SDU
5	17/12/2002	SDU
12	17/12/2002	SDU

Temos os aviões, o dia e por onde passaram, resta descobrir quem passou mais de uma vez em um aeroporto no mesmo dia.

Vamos para uma aplicação de função de agregação.

$$\pi_{\text{MATR}, \text{DATA}, \text{AEROPORTO}} \sigma_{\text{COUNT}(\text{TODOSVOOS})}$$

Agrupamos as linhas pelas colunas Matr, Data e Aeroporto.

Isso nos fornece como resultado a tabela:

Matr	Data	Aeroporto	Count
3	15/12/2002	GOI	2
3	15/12/2002	SDU	2
5	16/12/2002	BSB	2
5	16/12/2002	SDU	2
5	17/12/2002	POA	1
5	17/12/2002	SDU	1
12	16/12/2002	CGH	3
12	16/12/2002	SDU	3
12	17/12/2002	CGH	1
12	17/12/2002	SDU	1
15	15/12/2002	CGH	4
15	15/12/2002	SDU	4
15	20/12/2002	BSB	1
15	20/12/2002	SDU	1
20	16/12/2002	POA	1
20	16/12/2002	SDU	1
20	17/12/2002	POA	1
20	17/12/2002	SDU	1
20	20/12/2002	POA	1
20	20/12/2002	SDU	1
32	19/12/2002	REC	1
32	19/12/2002	SDU	1
40	18/12/2002	FOR	1
40	18/12/2002	SDU	1

Como estamos pedindo somente os aviões que estiveram mais de uma vez no mesmo aeroporto, interessa selecionar somente aqueles cujo contador é maior que 1.

Logo:

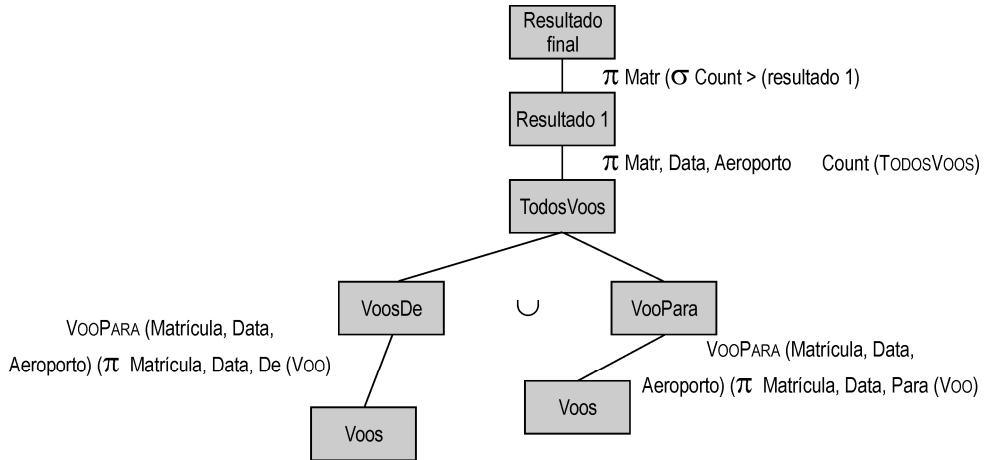
$$\pi_{\text{MATR}} (\sigma_{\text{COUNT} > 1} (\text{MATR}, \text{DATA}, \text{AEROPORTO}) \tau_{\text{COUNT}(\text{TODOSVOOS})})$$

E temos, finalmente, a tabela que responde à questão:

Matr
3
5
12
15

Complicadinho, mas na realidade é somente uma sequência lógica de como realizamos mentalmente a solução deste problema passo a passo.

Observe o gráfico ilustrativo da sequência de operações.



1. Projetamos por avião, data e local (Para) todos os voos.
2. Projetamos por avião, data e local (De) todos os voos.
3. Unimos as duas tabelas resultantes e temos agora os dados projetados por avião, data e um único local (De e Para como uma única coluna 'Aeroporto').
4. Aplicamos a contagem de linhas iguais agrupadas por avião, data e local.
5. Aplicamos uma agregação (função) e selecionamos as linhas com contador de valor maior que 1.

Agora pode parecer algo meio novo, mas ainda não terminamos de ver todas as operações de álgebra relacional. Vimos a maior parte dos operadores, porém ainda existem mais operações a serem estudadas. Vamos seguir com a apresentação dos operadores e seus exemplos.

Operação de Semijunção

A operação de semijunção de uma tabela A definida com um conjunto de atributos por outra tabela B com seu conjunto de atributos e relacionada com a tabela A são as linhas da tabela A que participam da junção da tabela A com a tabela B.

A notação da sintaxe de semijunção é:

Tabela A \bowtie_f Tabela B

A vantagem de utilizar a semijunção é o fato de diminuir o número de linhas que precisam ser manipuladas para formar uma junção.

Ela é muito importante em bancos de dados distribuídos, pois reduz, na maioria das vezes, a quantidade de dados que precisam ser transmitidos entre sites.

Para facilitar o entendimento da semijunção e da junção, vamos exemplificar utilizando as tabelas de funcionários e cargos já vistas neste capítulo, que repetimos aqui para você não ter de voltar páginas do livro.

Funcionário

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/8/2003	M	C3	D5
104	Carlos Pereira	2/3/2004	M	C4	D6
134	Jose Alves	23/5/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
195	Marta Silveira	5/1/2002	F	C1	D5
139	Ana Luiza Magalhães	12/1/2003	F	C4	D6
123	Pedro Sergio Doto	29/6/2003	M	C7	D3
148	Larissa Silva	1/6/2002	F	C4	D6
115	Roberto Fernandes	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/2/2000	M	C2	D4

Cargo

CdCargo	NumCargo	VlrSalario
C1	Aux Vendas	350,00
C3	Vendedor	800,00
C7	Diretor	2500,00
C2	Vigia	400,00
C5	Gerente	1000,00
C4	Aux Cobrança	250,00

O resultado da junção expressa por:

FUNCIONÁRIO \bowtie FUNCIONARIO.CDCARGO=CARGO.CDCARGO CARGO

é a tabela seguinte:

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept	CdCargo	NumCargo	VlrSalario
101	Luis Sampaio	10/08/2003	M	C3	D5	C3	Vendedor	800,00
104	Carlos Pereira	02/03/2004	M	C4	D6	C4	Aux Cobrança	250,00
134	Jose Alves	23/05/2002	M	C5	D1	C5	Gerente	1000,00
121	Luis Paulo Souza	10/12/2001	M	C3	D5	C3	Vendedor	800,00
195	Marta Silveira	05/01/2002	F	C1	D5	C1	Aux Cobrança	350,00
139	Ana Luiza Magalhães	12/01/2003	F	C4	D6	C4	Diretor	250,00
123	Pedro Sergio Doto	29/06/2003	M	C7	D3	C7	Aux Cobrança	2500,00
148	Larissa Silva	01/06/2002	F	C4	D6	C4	Vendedor	250,00
115	Roberto Fernandes	15/10/2003	M	C3	D5	C3	Vendedor	800,00
22	Sergio Nogueira	10/02/2000	M	C2	D4	C2	Vigia	400,00

Já o resultado da semijunção de funcionário e cargo expresso pela sintaxe:

FUNCIONÁRIO \bowtie FUNCIONARIO.CDCARGO=CARGO.CDCARGO CARGO

é a tabela seguinte:

NumReg	NomeFunc	DtAdmissão	Sexo	CdCargo	CdDept
101	Luis Sampaio	10/08/2003	M	C3	D5
104	Carlos Pereira	02/03/2004	M	C4	D6
134	Jose Alves	23/05/2002	M	C5	D1
121	Luis Paulo Souza	10/12/2001	M	C3	D5
195	Marta Silveira	05/01/2002	F	C1	D5
139	Ana Luiza Magalhães	12/01/2003	F	C4	D6
123	Pedro Sergio Doto	29/06/2003	M	C7	D3
148	Larissa Silva	01/06/2002	F	C4	D6
115	Roberto Fernander	15/10/2003	M	C3	D5
22	Sergio Nogueira	10/02/2000	M	C2	D4

Observe que a tabela resultante não tem os outros atributos de cargo, portanto temos como resultado uma tabela menor.

A semijunção, entretanto, tem comutatividade, ou seja, o resultado é diferente se ela for aberta à direita ou à esquerda.

A semijunção a seguir, por exemplo, tem resultado diferente da anterior.

CdCargo	NumCargo	VlrSalario
C1	Aux Vendas	350,00
C3	Vendedor	800,00
C7	Vendedor	800,00
C7	Diretor	2500,00
C2	Vigia	400,00
C5	Gerente	1000,00
C4	Aux Cobrança	250,00

Resultando praticamente na tabela cargo.

Junção Externa (Outer-Join)

Esta é outra extensão da operação e junção e objetiva apresentar informações omitidas em uma operação de junção normalmente.

Para explicar melhor, diríamos que uma junção externa permite descobrir quais elementos de uma tabela não estão associados, relacionados a elementos de outra tabela.

A junção externa possui comutatividade, ou seja, temos junção externa à esquerda e junção externa à direita:

- ▶ Formato para junção externa à direita:

Tabela A  Tabela B

- ▶ Formato para junção externa à esquerda:

Tabela A  Tabela B

Vamos trabalhar com as tabelas bancárias um pouco modificadas em seus elementos para exemplificar as opções de junção externa.

Agência

CodAgencia	NomAgencia	CidadeAgencia
1	Rio Branco	Rio de Janeiro
2	Icarai	Niterói
3	Leblon	Rio de Janeiro
4	Ipanema	Rio de Janeiro
5	Copacabana	Rio de Janeiro

Cliente

CodCliente	NomCliente	RuaCliente	CidadeCliente
1	Luis Sampaio	Rua A	Rio de Janeiro
2	Carlos Pereira	Rua B	Niterói
3	Jose Alves	Rua C	Rio de Janeiro
4	Luis Paulo Souza	Rua B	Niterói
5	Silvio Luis	Rua D	Rio de Janeiro

Conta-Corrente

CodAgencia	NumConta	CodCliente	Saldo
1	256589	1	1200,00
3	328941	1	845,00
4	749621	3	512,00
2	856200	2	2650,00

Observe que nem todas as agências da tabela possuem contas-correntes referenciando-as, assim como temos clientes cadastrados que agora não possuem conta-corrente.

Utilizando a operação de junção, não conseguimos visualizar essas ocorrências, porém com a junção externa podemos obter a informação faltante.

Para vermos as agências que possuem conta-corrente e também as que não possuem, usamos uma junção à direita:

AGÊNCIA  CONTACORRENTE

Observe a tabela resultado da operação:

CodAgencia	NomAgencia	CidadeAgencia	CodAgencia	NumConta	CodCliente	Saldo
1	Rio Branco	Rio de Janeiro	1	256589	1	1200,00
2	Icarai	Niterói	2	856200	2	2650,00
3	Leblon	Rio de Janeiro	3	328941	1	845,00
4	Ipanema	Rio de Janeiro	4	749621	3	512,00
5	Copacabana	Rio de Janeiro	nulo	nulo	nulo	nulo

A agência CodAgencia = 5 não possui nenhuma linha de conta-corrente associada, porém é apresentada com as colunas referentes à tabela ContaCorrente com valor nulo.

Se realizarmos a operação inversa:

AGÊNCIA  CONTACORRENTE

Não existirão linhas com conteúdo nulo:

CodAgencia	NomAgencia	CidadeAgencia	CodAgencia	NumConta	CodCliente	Saldo
1	Rio Branco	Rio de Janeiro	1	256589	1	1200,00
2	Icarai	Niterói	2	856200	2	2650,00
3	Leblon	Rio de Janeiro	3	328941	1	845,00
4	Ipanema	Rio de Janeiro	4	749621	3	512,00

E se houvesse, apresentariam falha de integridade do banco de dados, pois teríamos ocorrências de conta-corrente sem agência.

Entretanto, podemos ver um outro caso dessa junção em:

CONTA CORRENTE  CLIENTE

Que fornece como resultado:

CodAgencia	NumConta	CodCliente	Saldo	CodCliente	Nom Cliente	RuaCliente	CidadeCliente
1	256589	1	1200,00	1	Luis Sampaio	Rua A	Rio de Janeiro
3	328941	1	845,00	1	Luis Sampaio	Rua A	Rio de Janeiro
2	856200	2	2650,00	2	Carlos Pereira	Rua B	Niterói
4	749621	3	512,00	3	Jose Alves	Rua C	Rio de Janeiro
nulo	nulo	nulo	nulo	4	Luis Paulo Souza	Rua B	Niterói
nulo	nulo	nulo	v	5	Silvio Luis	Rua D	Rio de Janeiro

Como nas tabelas apresentadas somente três clientes possuem conta-corrente, aparecem os outros dois clientes, porém com os valores relativos à tabela ContaCorrente nulos.

Exercícios Propostos

1. Dadas as tabelas da figura seguinte, que expressões utilizando apenas as tabelas R e S permitem construir as tabelas T e U?

Tabela R

A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

Tabela S

B	C	D
b	c	d
b	c	e
a	d	b

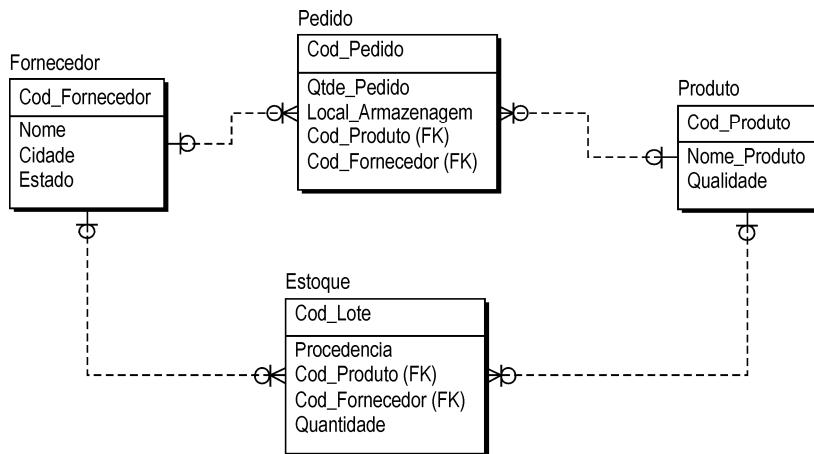
Tabela T

A	B	C	D
a	b	c	d
a	b	c	e
d	b	c	d
d	b	c	e
c	a	d	b

Tabela U

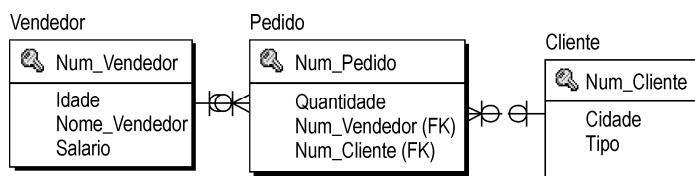
A	D
a	d
a	e
d	d
d	e

2. Construa expressões em álgebra relacional para as seguintes consultas, relativas ao banco de dados para controle de estoque apresentado em seguida. É muito importante simular os conteúdos das tabelas do modelo de dados.



- Encontre os nomes dos produtos de primeira qualidade.
 - Encontre os nomes dos fornecedores da Região Sul do Brasil.
 - Forneça as quantidades de produtos pedidas para cada local de armazenamento.
 - Localize os nomes dos produtos para os quais há pedido(s) cadastrado(s).
 - Encontre as cidades dos fornecedores para os quais há pedidos cadastrados.
 - Encontre os nomes de produtos em estoque procedentes de fornecedores da Região Sul do Brasil.
 - Localize os nomes dos produtos para os quais há pedidos e mercadorias em estoque.
 - Encontre os nomes dos produtos cadastrados para os quais não há registros em estoque nem em pedido.
3. Construa mais expressões em álgebra para este modelo conforme as solicitações de consulta apresentadas em seguida:
- Encontre os fornecedores que contribuíram com produtos de primeira qualidade no estoque atual.
 - Encontre os fornecedores que contribuíram com produtos de primeira qualidade no estoque atual ou para os quais haja pedido(s) de algum produto de primeira qualidade.
 - Quais são os nomes dos produtos esgotados no estoque para os quais não há pedido(s)?
 - Quais são os nomes dos produtos para os quais há estoque e pedido(s)?

- e. Quais produtos do estoque têm procedência diferente das cidades dos respectivos fornecedores? Retorne o nome do fornecedor, o nome do produto, a cidade do fornecedor e a procedência do produto.
- f. Quais são os fornecedores localizados nas mesmas cidades dos armazéns a que se destinam os pedidos? Retorne o nome do fornecedor, o nome do produto e a cidade.
4. Dado o modelo seguinte:



Responda:

- a. Quais são os nomes de todos os vendedores?
- b. Quais são os números dos vendedores que realizaram algum pedido?
- c. Quais são os números dos vendedores que não têm pedidos?
- d. Quais são os vendedores que realizaram pedidos para clientes do tipo indústria?
- e. Que tipo de clientes foram atendidos pelo vendedor de nome Luis Paulo?
5. Fornecidas as estruturas de dados (sem o modelo de dados), responda às consultas seguintes utilizando as operações de álgebra relacional.

Formato → Tabela (Atributos)

Empregado (IDEMP, NOMEEMP, DTNASCIMENTO, ENDERECOEMP, SEXOEMP, SLARIOEMP, NUMDEP)

Departamento (NUMDEP, NOMEDEP, IDGERENTE, HORARIO)

Localizacao (LOCALIZAÇÃO, NUMDEP)

Trabalha_Em (IDEMP, NUMPROJETO, HORAS)

Projeto (NUMPROJETO, NOMEPROJETO, NUMDEP)

Dependente (IDEMP, IDDEPENDENTE, NOMEDEPEDENDENTE, SEXODEPENDENTE, DTNASCDEPEND, PARENTESCO).

- a. Obter o nome e o endereço de todos os empregados que trabalham no departamento de compras.
- b. Para cada projeto localizado no Rio de Janeiro exibir o número do projeto, o número do departamento que o controla e a identidade de seu gerente, seu endereço e a data de nascimento dele.

- c. Descobrir os nomes dos projetos nos quais trabalham empregados com o nome Antonio.
- d. Listar os empregados que não têm dependentes.

Não basta saber como se escrevem as operações relacionais, é preciso pensar e estruturar as consultas.

Acreditamos ter apresentado de uma forma bem didática as operações relacionais e fornecido os caminhos de solução e raciocínio que normalmente não são explanados em sala de aula. O objetivo é contribuir para o aprendizado e domínio da utilização de álgebra relacional por todos os que dela necessitam e a utilizam.

SQL

A Importância da Linguagem SQL

SQL significa *Structured Query Language* - Linguagem Estruturada de Pesquisa. De grande utilização, teve seus fundamentos no modelo relacional de Codd (1970). Sua primeira versão recebeu o nome de SEQUEL (*Structured English Query Language*), 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 essa revisão foi posto em prática um projeto ambicioso da IBM chamado System R e novas alterações foram introduzidas na SQL, graças às ideias 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 aumentando cada vez mais. Com isso uma grande quantidade de SGBDs foi utilizando como linguagem básica a SQL - SQL/DS e DB2 da IBM, Oracle da Oracle Corporation, RDB da Digital, Sybase da Sybase Inc., Microsoft® SQL Server™, entre outras.

A SQL se tornou um padrão de fato no mundo dos ambientes de banco de dados relacionais. Bastava agora se tornar de direito. 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 vamos seguir o padrão ANSI da SQL, tentando ser o mais isento possível de implementações específicas de cada fabricante.

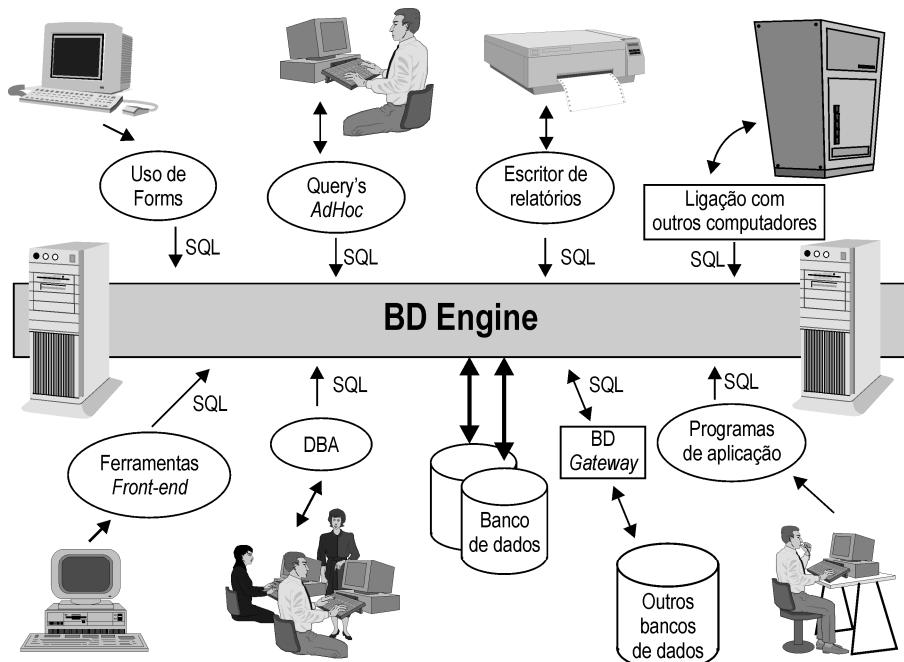
Como vimos no capítulo anterior, o modelo relacional é constituído basicamente de tabelas, cada qual contendo linhas (registros, tuplas) e colunas. Os registros na tabela não são ordenados e sua localização é feita por um campo-chave, ou seja, um campo que assume o papel de chave primária da tabela. Com essa chave identifica-se uma, e somente uma, ocorrência do valor contido no campo.

Uma das razões da popularidade dos sistemas relacionais é a sua facilidade de manutenção e entendimento.

A linguagem SQL foi desenvolvida especialmente para o ambiente relacional, podendo ser adaptada a qualquer ambiente não relacional.

A Linguagem SQL

A ideia 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.

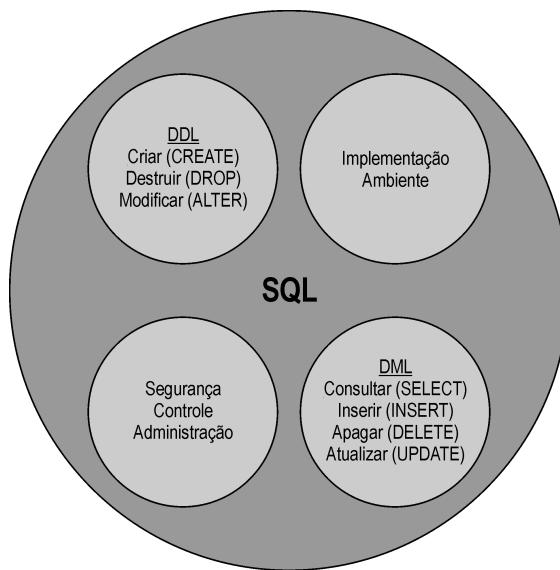


Atualmente, a linguagem SQL assume um papel muito importante nos Sistemas de Gerenciamento de Banco de Dados, podendo ter muitos enfoques, como mostra a figura.

- ▶ **Linguagem interativa de consulta (Query AdHoc):** com os comandos SQL os usuários podem montar consultas poderosas sem a necessidade de criar 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 através de vários nós conectados ao sistema de aplicação. 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 diversas máquinas (de micro até mainframe).

Por ser uma linguagem de numerosas aplicações, a SQL pode manipular objetos de diferentes classes entre as funções de um SGBD:



- ▶ **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.

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 SGBDs, e os que ainda não têm estão se encaminhando para lá. Com isso é possível mudar de SGBD sem se preocupar com o novo que vai chegar.
- ▶ **Portabilidade entre computadores:** pode ser utilizada tanto em um computador pessoal, passando por uma estação de trabalho, como em 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 acesso rápido aos dados, fornecendo respostas ao usuário, a questões complexas, em minutos ou segundos.
- ▶ **Múltiplas visões dos dados:** permite ao criador do banco de dados levar diferentes visões dos dados a vários usuários.
- ▶ **Definição dinâmica dos dados:** com a SQL é possível 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.

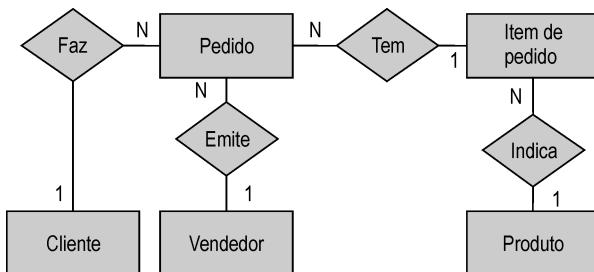
E é sobre esse auxílio que o capítulo trata, mostrando comandos e funcionalidades da SQL com exemplos práticos.

Não vamos mostrar todos os comandos, principalmente os que foram definidos para serem utilizados dentro de uma linguagem hospedeira (cursor), porém apresentamos 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 MS-SQL Server 2008, um produto da Microsoft®.

O Exemplo

Todo o percurso pela linguagem SQL será efetuado com base no exemplo de modelo de dados apresentado na figura a seguir, criado no capítulo sobre normalização, quando da apresentação das três principais formas normais.



A seguir, são apresentadas as tabelas referentes ao modelo.

Tabela Cliente

Cód. do Cliente	Nome do Cliente	Endereço	Cidade	CEP	UF	CNPJ	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. Clímerio 45	São Paulo	25679300	SP	32485126/7326-8	9280
157	Paulo	Tv. Moraes c/3	Londrina		PR	32848223/324-2	1923

Cód. do Cliente	Nome do Cliente	Endereço	Cidade	CEP	UF	CNPJ	IE
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 123 bl.2 sl. 345	São Paulo	30225900	SP	13276571/1231-4	1820
390	Sebastião	Rua da Igreja 10	Uberaba	30438700	MG	32176547/213-3	9071
234	José	Quadra 3 bl. 3 sl. 1003	Brasília	22841650	DF	21763576/1232-3	2931

Tabela Vendedores

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	B
250	Mauricio	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 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
22	M	Linho	0,11
30	SAC	Açúcar	0,3
53	M	Linha	1,8
13	G	Ouro	6,18
45	M	Madeira	0,25
87	M	Cano	1,97
77	M	Papel	1,05

Tabela Item de Pedido

Número do Pedido	Código do Produto	Quantidade
121	25	10
121	31	35
97	77	20
101	31	9
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

Número do Pedido	Código do Produto	Quantidade
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

As informações das tabelas serão utilizadas pelos comandos SQL, apresentados ao longo deste capítulo.

Criação e Distribuição de Tabelas

São as operações para que seja possível 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 em que residirão as tabelas de nosso sistema.

Por exemplo, no Microsoft® SQL Server™ 2008 essa operação é realizada normalmente por equipes de suporte. Consiste em dois passos, no mínimo, que são pela ordem:

1. Inicializar os arquivos em que serão armazenados os databases das 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™ 2008 o comando utilizado é o DISK INIT, mas não é objeto de nosso estudo.

2. Criar os databases nos devices já criados anteriormente.

Para isso vamos ao Microsoft® SQL Server™ 2008 usar o comando CREATE DATABASE com a sintaxe seguinte:

```
CREATE DATABASE database_name  
[ON {DEFAULT| database_device} [= size]  
[, database_device [= size] ...]  
[LOG ON database_device [= size]  
[, database_device [= size] ...]]
```

Exemplos:

```
CREATE DATABASE vendas
```

- ▶ Cria o database vendas no device default com tamanho default de 2 MB.

```
CREATE DATABASE vendas ON default = 256
```

- ▶ Cria o database vendas no device default com 256 MB.

```
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.

```
CREATE DATABASE vendas ON library_dev1 = 10 LOG ON librlog_dev2 = 4
```

- ▶ Cria o database vendas, 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 banco de dados da aplicação, podemos partir para a criação das tabelas.

Criação de 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 tabela a ser criada.
- ▶ **<descrição das colunas>**: é uma lista de colunas (campos) e seus respectivos tipos de dados. O tipo é determinado quando a tabela é criada e não pode ser alterado posteriormente. Os tipos de dados existentes são:

Para dados	Tipo	Tamanho
Caractere	<i>char(n), varchar(n), nvarchar(n), nchar(n)</i>	até <i>n</i> bytes
Numérico exato	<i>decimal(p,e)</i> ou <i>numeric(p,e)</i>	depende
Numérico aproximado	<i>float, real</i>	8, 4 bytes
Numérico inteiro	<i>int, smallint, tinyint</i>	4, 2, 1 byte
Monetário	<i>money, smallmoney</i>	8, 4 bytes
Data e hora	<i>datetime, smalldatetime</i>	8, 4 bytes
Binário	<i>binary(n), varbinary(n)</i>	<i>n</i> bytes
Texto e imagens	<i>text, image, ntext</i>	-variável-
Outros	<i>bit, timestamp</i>	1 bit, 8 bytes

Para dados contendo caracteres, *char(n)*, será armazenado um número fixo de caracteres. Por exemplo, uma coluna do tipo *char(30)* tem sempre 30 caracteres. Se forem informados menos, o restante será completado com espaços. Já o tipo *varchar(n)* armazena uma quantidade variável de caracteres até o máximo informado. Os tipos *nchar(n)* e *nvarchar(n)* armazenam dados Unicode, de comprimento fixo ou variável, e usam o conjunto de caracteres UNICODE UCS-2.(Microsoft® SQL Server™ 2008).

Por exemplo, *decimal (9,2)* permite guardar sete dígitos antes do ponto decimal e dois após, num total de nove, assim o maior valor possível é 9999999,99.

Os tipos numéricos inexatos, *float* e *real*, armazenam dados numéricos, mas nem sempre mantêm a precisão suficiente para armazenar corretamente números de vários dígitos.

O tipo *money* é usado para valores monetários, ocupando 8 bytes em disco e permitindo valores entre -922.337.203.685.477,5808 e +922.337.203.685.477,5807 (922 trilhões). O tipo *smallmoney* permite valores entre - 214.748,3648 e +214.748,3647 (214 mil) e ocupa 4 bytes em disco.

Dos tipos inteiros, *int* usa 32 bits (4 bytes), permitindo armazenar até +/-2.147.483.647, *smallint* usa 16 bits (2 bytes), permitindo +/-32767 e *tinyint* usa 8 bits (1 byte), permitindo números não negativos de 0 a 255.

O tipo *datetime* armazena valores contendo a data e hora, com precisão de 1/300 de segundo, entre 1º de janeiro de 1753 e 31 de dezembro de 9999 (o século é sempre armazenado). O tipo *smalldatetime* ocupa menos espaço e armazena datas e horas de 1º de janeiro de 1900 até 6 de junho de 2079 com precisão de 1 minuto.

Tipos binários são usados para dados que o Microsoft® SQL Server™ 2008 não interpreta, por exemplo, o conteúdo de um arquivo binário. O tipo *text* é usado para colunas com dados "memo", ou seja, com texto de tamanho variável; o tipo *ntext* armazena dados Unicode de tamanho variável. O tipo *image* armazena imagens também de tamanho variável.

Os tipos *text* e *ntext* armazenam dados de tamanho variável, mas podem armazenar 1.073.741.823 caracteres para o caso do *ntext*, e 2.146.483.647 caracteres para o caso do tipo *text*. Enquanto isso, os tipos *varchar* e *nvarchar* armazenam "somente" 8.000 caracteres (*varchar*) ou 4.000 caracteres (*nvarchar*).

O tipo *bit* armazena valor 1 ou 0. Uma coluna do tipo *timestamp* não pode ser alterada pelo usuário. Ela é definida automaticamente com a data e hora atual quando a linha é inserida ou atualizada.

Os tipos "numéricos exatos", *decimal* e *numeric*, permitem armazenar dados exatos, sem perdas devido a arredondamento. Ao usar esses tipos, você pode especificar uma *precisão*, que indica quantos dígitos podem ser usados no total e uma *escala*, que indica quantos dígitos podem ser usados à direita do ponto.

Alguns campos podem receber o valor NULL (nulo) e o campo definido como chave primária, além de não poder receber nulo (NOT NULL), deve ser um campo UNIQUE (sem repetições - chave primária). Para o banco de dados da figura anterior temos os seguintes comandos:

- ▶ <**descrição das chaves**>: é a lista de colunas tratadas como chave estrangeira.

Então, para o nosso modelo de dados temos:

```
CREATE TABLE CLIENTE
(codigo_cliente smallint not null unique,
nome_cliente    char(20),
endereco       char(30),
cidade          char(15),
CEP             char(8),
UF              char(2),
CNPJ            char(20),
IE              char(20));

CREATE TABLE PEDIDO
numero_pedido      int          not null        unique,
prazo_de_entrega  smallint     not null,
codigo_cliente     smallint     not null,
codigo_vendedor   smallint     not null,
```

```

FOREIGN KEY          (codigo_cliente
                      REFERENCES CLIENTE,
FOREIGN KEY          (codigo_vendedor)
                      REFERENCES VENDEDORES);

CREATE TABLE ITEM_DE_PEDIDO
(numero_pedido      int          not null unique,
codigo_produto     smallint     not null unique,
quantidade         decimal,
FOREIGN KEY          (numero_pedido)
                      REFERENCES PEDIDO,
FOREIGN KEY          (codigo_produto)
                      REFERENCES PRODUTO);

```

Observe que a cláusula **REFERENCE** estabelece a restrição de integridade referencial entre as tabelas no SQL padrão, porém só podemos incluir essas restrições se as tabelas referidas na cláusula **REFERENCE** já foram criadas antes desta.

No Microsoft® SQL Server™ 2008 a integridade referencial necessita de um acréscimo na estrutura da sintaxe do comando com a inclusão da declaração de uma constraint. As constraints são propriedades que devem ser declaradas para determinadas colunas, como chaves primárias e chaves estrangeiras.

Essas propriedades implicam em regras de validação para essas colunas, objetivando impedir operações de inclusão, alteração ou deleção que possam tornar os dados do banco de dados inconsistentes.

Quando desejamos criar a chave primária da tabela no próprio comando CREATE, a sintaxe do Microsoft® SQL Server™ 2008 é a seguinte:

```

CREATE TABLE CLIENTE
(codigo_cliente smallint      not null unique,
nome_cliente   char(20),
endereco       char(30),
cidade          char(15),
CEP             char(8),
UF              char(2),
CNPJ            char(20),
IE              char(20) )
CONSTRAINT PK_Cliente Primary Key (codigo_cliente);

```

Desta forma a tabela cliente foi criada com a chave primária com restrição (constraint).

Como em nosso exemplo a tabela produto ainda não foi criada, teríamos um erro ao executar esses comandos. A tabela item de pedido deve ser criada após a tabela produto.

Da mesma forma a tabela pedido somente pode ser criada com a restrição de integridade referencial após a criação de cliente e vendedor.

Esses fatores de sequência de criação são muitas vezes ignorados por quem executa a criação das tabelas do banco de dados, sendo muito comum para a fuga do erro, 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 deste comando.

```

CREATE TABLE VENDEDORES
(codigo_vendedor      smallint not null,
nome_vendedor         char(20),
salario_fixo          money,
faixa_de_comissao    char(1)
PRIMARY KEY (codigo_vendedor));

```

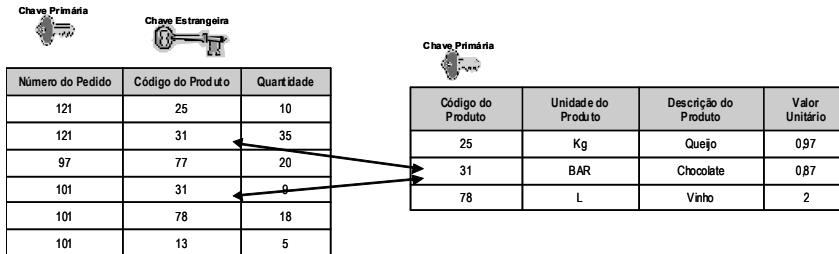
Note que a chave primária já está definida juntamente com o registro da tabela. A criação do índice, por razões óbvias, após a tabela, naturalmente é um comando totalmente independente do primeiro comando create, que serviu para criar a tabela e suas características básicas.

```

CREATE TABLE PRODUTO
(codigo_produto       smallint not null unique,
unidade_produto      char(3),
descricao_produto    char(30),
val_unitario          money
CONSTRAINT PK_Produtos Primary Key (codigo_produto));

```

No SQL Server e no Oracle, a integridade referencial baseia-se nas relações entre chaves estrangeiras e chaves primárias ou entre chaves estrangeiras e chaves exclusivas, por meio de restrições FOREIGN KEY e CHECK. A integridade referencial assegura que os valores chave permaneçam consistentes em todas as tabelas. Esse tipo de consistência requer que não haja referências a valores não existentes e que se um valor chave é alterado, todas as referências a ele são consistentemente alteradas em todo o banco de dados.



Criação de Chaves Primárias Compostas

Até este momento criamos a tabela item_de_pedido sem declaração nenhuma de chave primária. Criamos somente com as chaves estrangeiras.

Vamos analisar essa tabela para criá-la com chave primária.

Em nosso modelo de dados essa tabela representa uma entidade associativa entre pedido e produto, logo sua chave primária será composta das duas chaves primárias das tabelas adjacentes do relacionamento, produto e pedido.

```

CREATE TABLE ITEM_DE_PEDIDO
(numero_pedido        int           not null unique,
codigo_produto        smallint      not null unique,
quantidade            decimal,

```

```
CONSTRAINT PK_ITEM_DE_PEDIDO
    PRIMARY KEY (numero_pedido, codigo_produto)
CONSTRAINT FK_PEDIDO
    FOREIGN KEY (numero_pedido)
    REFERENCES PEDIDO
CONSTRAINT FK_PRODUTO
    FOREIGN KEY (codigo_produto)
    REFERENCES PRODUTO;
```

Eliminação de uma Tabela

Para eliminar uma tabela criada, utiliza-se o comando **DROP**:

Sintaxe Básica

```
DROP TABLE <tabela>;
```

Exemplo:

```
DROP TABLE PEDIDO
```

Elimina a tabela de pedidos que foi previamente criada, seus dados e suas referências a outras tabelas. Esse comando somente resulta erro se existir alguma tabela que seja dependente dele quando da criação.

Por exemplo, se existe no nosso caso a tabela item de pedido e já criamos as restrições de integridade referencial, o comando retorna erro, pois não será permitida a eliminação da tabela por existirem referências a ela em outra tabela.

Alteração da Estrutura das Tabelas

Depois que uma tabela for criada, é possível mudar várias das opções que foram definidas quando a tabela foi originalmente criada, incluindo:

Colunas podem ser acrescentadas, modificadas ou excluídas. Por exemplo, o nome da coluna, comprimento, tipo de dados, precisão, escala e o fato de aceitar ou não valores nulos podem ser mudados, embora existam algumas restrições.

As restrições PRIMARY KEY e FOREIGN KEY podem ser acrescentadas ou excluídas.

É importante considerar que a mudança de tipo de dados em uma coluna pode causar truncamento dos dados, ou mesmo ser impossível de ser feita (por exemplo, se você quiser converter um tipo char em um tipo inteiro e já houver valores não numéricos armazenados nessa coluna).

O nome ou o dono de uma tabela também pode ser modificado.

Quando você faz isso, também deve mudar o nome da tabela em qualquer procedimento armazenado, *scripts* SQL, ou outro código de programação que utilize o nome ou proprietário antigo da tabela.

Para alterar uma tabela, utiliza-se o comando ALTER TABLE.

Adicionar uma chave primária (PRIMARY KEY) é feito com a inclusão de uma restrição na tabela, uma constraint, e a sua sintaxe é:

```
ALTER TABLE cliente  
ADD Constraint PK_Cliente  
Primary Key (codigo_cliente)
```

Se quando da execução do ALTER TABLE já existirem dados na tabela, o SQL verifica se existem dados repetidos nessa coluna. Se não existirem, a chave primária será criada com sucesso; caso contrário, será emitida uma mensagem de erro do banco, e a chave primária não será acrescentada à tabela.

Adicionar uma chave estrangeira (FOREIGN KEY/REFERENCES) também se realiza com a inclusão de uma restrição na tabela, uma constraint, e a sua sintaxe é:

```
ALTER TABLE pedido  
ADD Constraint FK_Pedido  
Foreign Key (codigo_cliente)  
References Cliente(codigo_cliente)
```

Da mesma forma se já existirem dados na tabela pedido, é verificado se os valores existentes na coluna codigo_cliente da tabela pedido existem na tabela cliente na mesma coluna. Se todos existirem, o comando é executado com sucesso e a chave estrangeira é criada na tabela; caso contrário, será emitida mensagem de erro e não será acrescentada a chave estrangeira.

Para acrescentar colunas a uma tabela, a sintaxe é:

```
ALTER TABLE produto  
ADD tipo_produto Char(20) NULL
```

Se considerarmos que já havíamos criado a tabela produto e carregado seus dados como na tabela apresentada no início deste capítulo, teremos agora a seguinte tabela resultado:

Código do Produto	Unidade do Produto	Descrição do Produto	Valor Unitário	Tipo Produto
25	Kg	Queijo	0,97	Null
31	BAR	Chocolate	0,87	Null
78	L	Vinho	2	Null
22	M	Linho	0,11	Null
30	SAC	Açúcar	0,3	Null
53	M	Linha	1,8	Null
13	G	Ouro	6,18	Null
45	M	Madeira	0,25	Null
87	M	Cano	1,97	Null
77	M	Papel	1,05	Null

Observe que foi colocada a coluna como NULL para permitir que você insira dados nas linhas da tabela já existente, mesmo que esses dados venham a ter características de serem

sempre obrigatórios. Desta forma você vai inserir valores na coluna em todas as linhas e depois modificar novamente a coluna para NOT NULL.

```
ALTER TABLE produto  
ALTER COLUMN tipo_produto Char(20) NOT NULL
```

E da mesma forma podemos agora retirar esta coluna da tabela:

```
ALTER TABLE produto  
DROP COLUMN tipo_produto
```

E nossa tabela volta à situação anterior.

No caso de a tabela possuir uma coluna que tem uma constraint associada, você deve primeiro eliminar a constraint para depois realizar a alteração de eliminar a coluna.

Digamos que queremos eliminar a coluna codigo_cliente da tabela pedido, porém ela possui uma propriedade de constraint, pois é uma chave estrangeira.

```
ALTER TABLE pedido  
DROP Constraint FK_Pedido  
ALTER TABLE pedido  
DROP COLUMN codigo_cliente
```

Em alguns casos é possível alterar o datatype de uma coluna.

Vamos supor que queremos alterar na tabela Produto o datatype da coluna unidade de CHAR(3) para Int.

Obviamente isso não é possível se já carregamos dados na tabela, pois a coluna contém dados no formato caractere que não podem ser convertidos em números. Mas se, por exemplo, desejássemos transformar a coluna valor_unitario em caractere, seria possível, pois podemos transformar um datatype money em char.

```
ALTER TABLE produto  
ALTER COLUMN valor_unitario char (12)
```

Coluna Calculada

No Microsoft® SQL Server™ 2008 podemos ter uma tabela com colunas calculadas que não existem fisicamente, pois não são armazenadas. Não possuem datatype nem condição de NULL, já que assumem esses valores das expressões que as definem, mas podem ser definidas quando da criação da tabela.

```
CREATE TABLE VENDEDORES  
(codigo_vendedor      smallint      not null,  
nome_vendedor        char(20),  
salario_fixo          money,  
faixa_de_comissao    char(1),  
Comissao_maxima as salario_fixo * 2,  
Constraint PK_VENDEDORES  
Primary Key (codigo_vendedor)
```

Criação de Ações em Cascata

A Cláusula ON DELETE CASCADE e ON UPDATE CASCADE

Esta é uma cláusula que deve ser utilizada com muito critério quando construímos um banco de dados pelos riscos implícitos na sua utilização, porém ela é de grande valia na garantia de integridade referencial no banco de dados.

A utilização de uma cláusula ON DELETE CASCADE em uma declaração de constraint de FOREIGN KEY especifica que se houver uma tentativa de apagar uma linha com uma chave primária referenciada por chaves estrangeiras em linhas existentes em outras tabelas, também serão apagadas todas as linhas que contêm essas chaves estrangeiras.

Vejamos então um comando possível no nosso exemplo:

```
ALTER TABLE Pedido  
ADD Constraint FK_Pedido  
Foreign Key (codigo_cliente)  
References cliente(codigo_cliente)  
ON DELETE CASCADE;
```

O que implica na operação das tabelas a existência de ON DELETE CASCADE?

Se for deletada alguma linha de cliente que tenha referência em pedido, as linhas correspondentes de pedido também serão deletadas.

A opção de ON UPDATE CASCADE impede que sejam feitas mudanças na chave referenciada caso existam linhas referenciando o valor desta.

Caso fosse possível mudar o valor de uma chave primária de cliente, a existência de ON UPDATE CASCADE na tabela pedido provocaria a alteração dos valores das chaves estrangeiras associadas ao valor original.

```
ALTER TABLE Pedido  
ADD Constraint FK_Pedido  
Foreign Key (Codigo_cliente)  
References Cliente(Codigo_cliente)  
ON DELETE CASCADE  
ON UPDATE CASCADE;
```

Podemos utilizar as duas cláusulas juntas sem problema.

Regras de Validação

Vamos considerar que nossa tabela de vendedores só permita por regra de negócio a existência de três faixas de comissão: A, B e C.

Desta forma não queremos permitir em hipótese nenhuma que seja cadastrado um vendedor, por exemplo, com faixa D.

Para isso utilizamos uma constraint check.

```
CREATE TABLE VENDEDORES
(codigo_vendedor      smallint      not null,
nome_vendedor         char(20),
salario_fixo          money,
faixa_comissao        char(1),
Comissao_maxima as salario_fixo * 2,
Constraint PK_VENDEDORES
Primary Key (codigo_vendedor),
CONSTRAINT CH_Vendedor
Check (faixa_comissao IN ('A', 'B', 'C'))
```

Como apresentamos em modelo físico, você pode aqui colocar a regra de negócio de cada coluna por meio de um constraint check.

Você pode também assinalar valores default para s colunas através de uma *constraint default* como no exemplo para a faixa de comissão também.

```
faixa_de_comissao char(1) CONSTRAINT DF_Vendedores Default 'A'
```

As constraints, quando criadas após a carga dos dados, podem ser utilizadas para validar os conteúdos e valores das colunas, pois se houver valores em desacordo com os checks definidos, por exemplo, será emitida mensagem de erro.

Extração de Dados de uma Tabela: SELECT

Uma das operações mais comuns, realizadas com um banco de dados, é examinar (selecionar) as informações armazenadas. Essas operações são realizadas pelo comando SELECT.

Neste item vamos mostrar várias situações de utilização do comando SELECT.

O comando SELECT tem palavras-chave em um comando básico:

- ▶ **SELECT:** especifica as colunas da tabela que queremos selecionar.
- ▶ **FROM:** especifica as tabelas.
- ▶ **WHERE:** especifica a condição de seleção das linhas.

Seleção de Colunas Específicas da Tabela

Sintaxe Básica

```
select <nome(s) da(s) coluna(s)>
      from <tabela>;
```

Problema: Listar todos os produtos com respectivas descrições, unidades e valores unitários.

Diagrama:

Produto
Cod_Produto
Descricao_Produto
Unidade_Produto
Valor_Unitario

Sintaxe:

```
SELECT unidade_produto, descricao_produto,  
       valor_unitario  
  FROM produto;
```

O que equivale em álgebra relacional a:

$\pi_{UNIDADE_PRODUTO, DESCRIÇÃO_PRODUTO, VALOR_UNITARIO} (PRODUTO)$

A execução deste comando neste formato lista todas as linhas da tabela produto, pois não apresenta nenhuma condição para a projeção dos dados.

SELECT sem **WHERE** lista todas as linhas de uma tabela.

Resultado:

Descrição	Unidade	Valor Unitário
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 CNPJ, o nome do cliente e seu endereço.

Diagrama: no 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.

Cliente
Codigo_Cliente
Nome_Cliente
Endereco
Cidade
CEP
UF
CNPJ
IE

Sintaxe:

```
SELECT CNPJ, nome_cliente, endereco  
      FROM cliente;
```

Resultado:

CNPJ	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. Clímerico 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

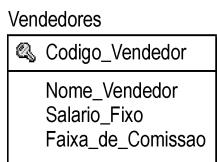
Seleção de Todas as Colunas da Tabela

Sintaxe Básica

```
SELECT *  
      FROM <tabela>;
```

Problema: Listar todo o conteúdo de vendedor.

Diagrama:



Sintaxe ANSI1:

```
SELECT *  
      FROM vendedores;
```

¹ A utilização do comando SELECT sem a cláusula WHERE causa uma desnecessária carga nos recursos de sistema. Por esta razão, em SQL Server 2008 ou em qualquer outro SGBD utilize sempre a cláusula WHERE.

Resultado:

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

Alteração do *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 a nosso gosto.

Sintaxe Básica

```
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,
       comissao = faixa_de_comissao
      FROM vendedores
```

Resultado:

Número	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

Manipulação de Dados Numéricos: Operadores Aritméticos

Operadores aritméticos podem ser usados em qualquer coluna numérica, incluindo colunas de tipo de dado *int*, *smallint*, *tinyint*, *float*, *real*, *money* ou *smallmoney*².

Os operadores aritméticos são:

Símbolo	Operação	Pode ser usado com (Microsoft® SQL Server™ 2008)
+	Adição	<i>Int</i> , <i>smallint</i> , <i>tinyint</i> , <i>numeric</i> , <i>decimal</i> , <i>float</i> , <i>real</i> , <i>money</i> and <i>smallmoney</i>
-	Subtração	<i>Int</i> , <i>smallint</i> , <i>tinyint</i> , <i>numeric</i> , <i>decimal</i> , <i>float</i> , <i>real</i> , <i>money</i> and <i>smallmoney</i>
/	Divisão	<i>Int</i> , <i>smallint</i> , <i>tinyint</i> , <i>numeric</i> , <i>decimal</i> , <i>float</i> , <i>real</i> , <i>money</i> and <i>smallmoney</i>
*	Multiplicação	<i>Int</i> , <i>smallint</i> , <i>tinyint</i> , <i>numeric</i> , <i>decimal</i> , <i>float</i> , <i>real</i> , <i>money</i> and <i>smallmoney</i>
%	Módulo	<i>Int</i> , <i>smallint</i> e <i>tinyint</i>

Exemplo:

```
SELECT nome_vendedor,
       salario_fixo = (salario_fixo * 2)
  FROM vendedores
```

Nome do Vendedor	Salário 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
Mauricio	5,860.00

O resultado apresenta os salários com valores multiplicados por dois.

Seleção de Somente Algumas Linhas da Tabela

A cláusula WHERE em um comando SELECT especifica quais linhas queremos obter baseada em condições de seleção.

Chamamos isso de observar uma seleção horizontal de informações e estamos aplicando uma operação de álgebra relacional à seleção.

² Datatypes de SQL Server 2008.

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>

► 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

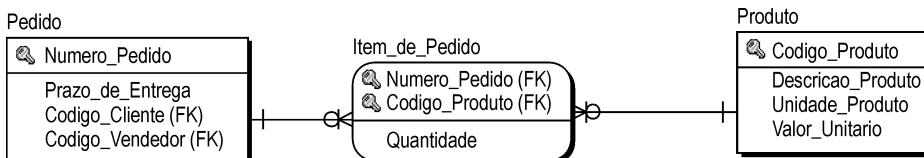
Quando a coluna é do tipo caractere, o <valor> deve estar entre aspas ('').

Exemplo: '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_de_pedido.

Diagrama:



Sintaxe:

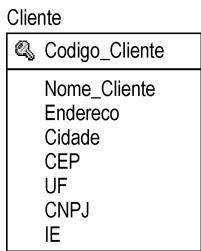
```
SELECT numero_pedido, codigo_produto, quantidade
FROM item_de_pedido
WHERE quantidade = 35;
```

Resultado:

Número do Pedido	Código do Produto	Quantidade
121	31	35
138	77	35

Problema: Quais são os clientes que moram em Niterói?

Diagrama:



Sintaxe:

```
SELECT nome_cliente  
      FROM cliente  
     WHERE cidade = 'Niteroi';
```

Resultado:

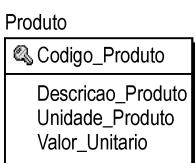
Nome do Cliente
Ana
Susana

► 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:



Sintaxe:

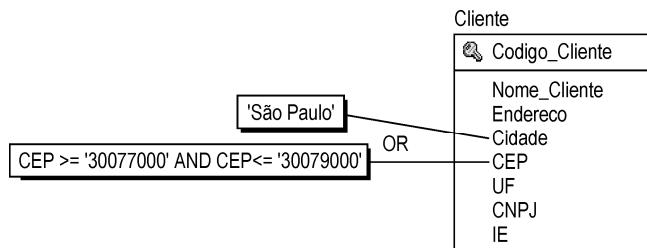
```
SELECT descricao_produto  
      FROM produto  
     WHERE unidade_produto = 'M' AND  
           valor_unitario = 1.05;
```

Resultado:

Descrição do Produto
Papel

Problema: Listar os clientes e seus respectivos endereços, que moram em 'SÃO PAULO' ou estejam na faixa de CEP entre '30077000' e '30079000'.

Diagrama:



Sintaxe:

```
SELECT nome_cliente, endereco
FROM cliente
WHERE (CEP >= '30077000' AND CEP <= '30079000')
OR cidade = 'Sao Paulo';
```

Isso equivale em álgebra relacional a:

$$\pi_{\text{NOME_CLIENTE}, \text{ENDERECO}} (\sigma_{(\text{CEP} \geq '30077000' \text{ AND } \text{CEP} \leq '30079000') \text{ OR } \text{CIDADE} = 'SAO PAULO'} (\text{CLIENTE}))$$

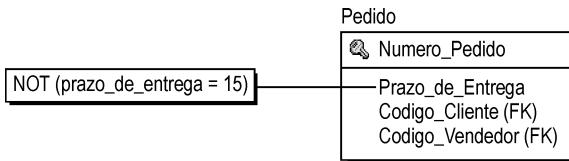
Resultado:

Nome do Cliente	Endereço do Cliente
Flávio	Av. Pres. Vargas 10
Jorge	Rua Caiapo 13
Maurício	Av. Paulista 1236 sl/2345
Rodolfo	Largo da Lapa 27 sobrado
Beth	Av. Clímerico 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:



Sintaxe:

```
SELECT numero_pedido  
FROM pedido  
WHERE NOT (prazo_de_entrega = 15);
```

```
π NUMERO_PEDIDO (σ PRAZO_DE_ENTREGA <> 15 (PEDIDO))
```

Ou podemos alternativamente utilizar um operador de comparação `<>` que vai realizar a mesma operação de seleção.

Sintaxe:

```
SELECT numero_pedido  
FROM pedido  
WHERE (prazo_de_entrega <> 15);
```

Resultado:

Número do Pedido
121
97
137
148
104
203
98
143
111
103
91
138
119
127

► Operadores Between e NOT Between

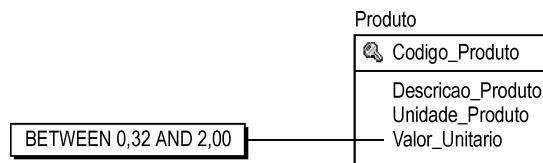
```
WHERE <nome da coluna> BETWEEN <valor1> AND <valor2>
WHERE <nome da coluna> NOT BETWEEN <valor1> AND <valor2>
```

Esse 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 \geq , \leq e AND. Tanto o VALOR1 quanto o VALOR2 precisam ser do mesmo tipo de dado da coluna.

Quando executado sobre colunas do tipo *char*, *varchar*, *text*, *datetime* e *smalldatetime*, devemos colocar estes valores entre aspas.³

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:



Sintaxe:

```
SELECT codigo_produto, descricao_produto
FROM produto
WHERE valor_unitario between 0.32 and 2.00;
```

Resultado:

Código do Produto	Descrição
25	Queijo
31	Chocolate
78	Vinho
53	Linha
87	Cano
77	Papel

³ Especificação para SQL Server 2008.

► Operadores Baseados em String de Caracteres LIKE e NOT LIKE

```
WHERE <nome da coluna> LIKE <valor>;  
WHERE <nome da coluna> NOT LIKE <valor>;
```

Os operadores LIKE e NOT LIKE só trabalham com 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

Exemplo:

LIKE 'LÁPIS %' pode enxergar os seguintes registros:

```
'LAPIS PRETO'  
'LAPIS CERA'  
'LAPIS 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:

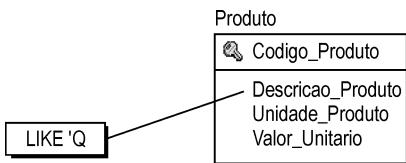
```
'BROCA N1'  
'BROCA N9'  
'BROCA N3'
```

- 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 'o' como segunda letra.

Vamos ver mais alguns exemplos de utilização da cláusula LIKE.

Problema: Listar todos os produtos cujo nome comece por Q.

Diagrama:



Sintaxe:

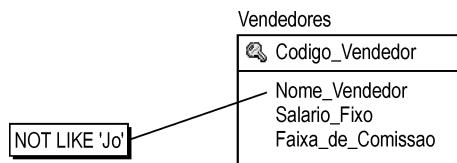
```
SELECT codigo_produto, descricao_produto  
FROM produto  
WHERE descricao_produto LIKE 'Q_';
```

Resultado:

Código do Produto	Descrição
25	Queijo

Problema: Listar os vendedores que não começam por 'Jo'.

Diagrama:



Sintaxe ANSI e Microsoft® SQL Server™ 2008:

```
SELECT codigo_vendedor, nome_vendedor
FROM vendedores
WHERE nome_vendedor NOT LIKE 'Jo%';
```

Resultado NOT LIKE:

Código do Vendedor	Nome do Vendedor
111	Carlos
240	Antônio
720	Felipe
250	Mauricio

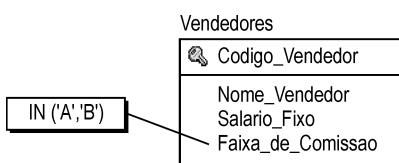
► Operadores Baseados em Listas IN e NOT IN

```
WHERE <nome da coluna> IN <valores>;
WHERE <nome da coluna> NOT IN <valores>;
```

Estes operadores pesquisam registros que estão ou não no conjunto de <valores> fornecido. Eles minimizam o uso dos operadores =, <>, AND e OR.

Problema: Listar os vendedores que são da faixa de comissão A e B.

Diagrama:



Sintaxe:

```
SELECT nome_vendedor
FROM vendedores
WHERE faixa_de_comissao IN ('A', 'B');
```

Resultado:

Nome do Vendedor
Carlos
Felipe
Jonas
Josias
Maurício

► Operadores Baseados em Valores Desconhecidos: IS NULL e IS NOT NULL

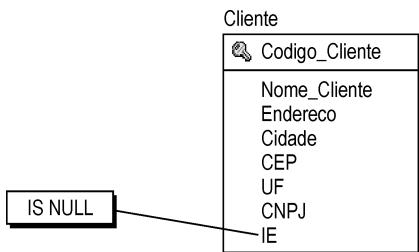
```
WHERE <nome da coluna> IS NULL;  
WHERE <nome da coluna> IS NOT NULL;
```

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 desses 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:



Sintaxe:

```
SELECT *  
FROM cliente  
WHERE IE IS NULL;
```

Resultado:

Código do Cliente	Nome do Cliente	Endereço	Cidade	CEP	UF	CNPJ	IE
110	Jorge	R. Caiapo 13	Curitiba	30078500	PR	145127645/983493-9	
180	Lívio	Av. Beira Mar 1256	Florianópolis	30077500	SC	127365713/2347-4	

Ordenação dos 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 a ordenação dos dados selecionados.

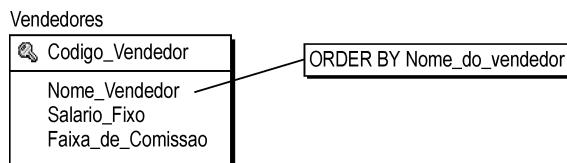
Sintaxe Básica

```
SELECT <nome da(s) coluna(s)>
FROM <tabela>
WHERE <condição(ões)> 
ORDER BY <nome da(s) coluna(s)> [ASC/DESC]
      ou
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:



Sintaxe:

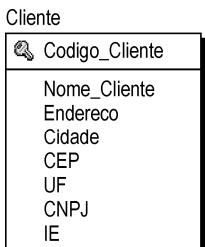
```
SELECT nome_vendedor, salario_fixo
FROM vendedores
ORDER BY nome_vendedor;
```

Resultado:

Nome do 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:



Sintaxe:

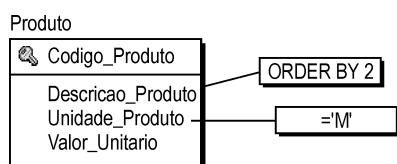
```
SELECT nome_cliente, cidade, UF  
FROM cliente  
ORDER BY UF DESC, cidade DESC;
```

Resultado:

Nome do 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:



Sintaxe:

```
SELECT descricao_produto, valor_unitario  
FROM produto  
WHERE unidade_produto = 'M'  
ORDER BY 2 ASC;
```

Resultado:

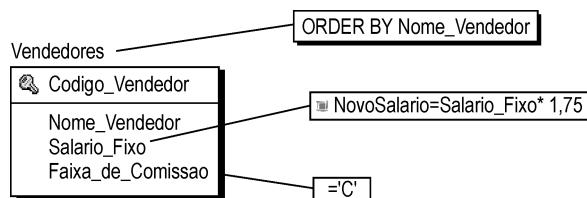
Descrição	Valor Unitário
Linho	0,11
Madeira	0,25
Papel	1,05
Linha	1,80
Cano	1,97

Realização de Cálculos com Informação Selecionada

Com a linguagem SQL é possível criar um campo que não pertença à tabela original e seja fruto de cálculo de alguns campos da tabela para ser exibido no momento da consulta.

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:



Sintaxe:

```
SELECT nome_vendedor,
       novo_salario = (salario_fixo * 1.75) + 120
  FROM vendedores
 WHERE faixa_de_comissao = 'C'
 ORDER BY nome_vendedor;
```

Resultado:

Nome do Vendedor	Novo Salário
Antônio	16.745,00
João	4.985,00
João	4.757,50
José	3270,00

Utilização de Funções de Agregação sobre Conjuntos

As funções de agregação resultam sempre uma nova coluna no resultado da pesquisa.

Busca de Máximos e Mínimos (MAX, MIN)

Problema: Mostrar o menor e o maior salários da tabela vendedores.

Diagrama:



Sintaxe:

```
SELECT MIN(salario_fixo), MAX(salario_fixo)
FROM vendedores;
```

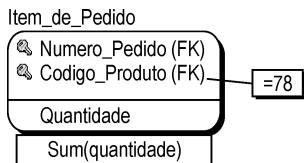
Resultado:

MIN (salario_fixo)	MAX (salario_fixo)
870,00	9.500,00

Totalização dos Valores de Colunas (SUM)

Problema: Mostrar a quantidade total pedida para o produto 'VINHO' de código '78' na tabela item_de_pedido.

Diagrama:



Sintaxe:

```
SELECT SUM(quantidade) ,
FROM item_de_pedido
WHERE código_produto = '78';
```

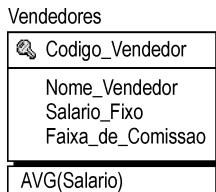
Resultado:

SUM(Quantidade)
183

Cálculo de Médias (AVG)

Problema: Qual a média dos salários fixos dos vendedores?

Diagrama:



Sintaxe:

```
SELECT AVG(salario_fixo),  
      FROM vendedores;
```

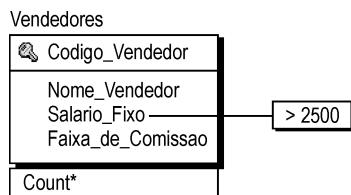
Resultado:

AVG (salario_fixo)
3.324,44

Contagem dos Registros (COUNT)

Problema: Quantos vendedores ganham acima de R\$ 2.500,00 de salário fixo?

Diagrama:



Sintaxe:

```
SELECT COUNT(*),  
      FROM vendedores  
     WHERE salario_fixo > 2500;
```

O comando COUNT, quando utilizado sem a cláusula WHERE, realiza a contagem das linhas da tabela.

Resultado:

COUNT (*)
5

Utilização da 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 são as unidades de produtos, diferentes, na tabela produto?

Diagrama:



Sintaxe:

```
SELECT DISTINCT unidade_produto,  
FROM produto;
```

Resultado:

Unidade
Kg
BAR
L
M
SAC
G

Importante: Com a utilização de DISTINCT não se classificam os dados de saída⁴.

Agrupamento de 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.

Podemos igualmente continuar com a cláusula WHERE selecionando as condições da seleção.

⁴ Especificação para SQL Server 2008.

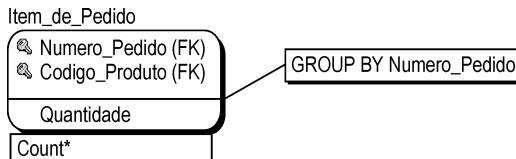
```

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:



Sintaxe:

```

SELECT numero_pedido,
total_produtos = COUNT(*)
FROM item_de_pedido
GROUP BY numero_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:

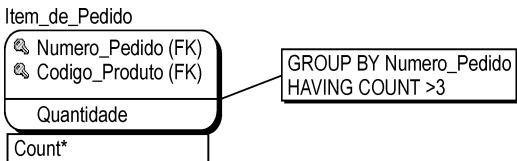
Número do Pedido	Total de Produtos
91	1
97	1
98	1
101	3
103	1
104	1
105	1
108	1
111	2
119	4
121	2
138	3
143	2
148	5
189	1
203	1

Geralmente, a cláusula GROUP BY é utilizada em conjunto com as operações COUNT e AVG.

Utilização com HAVING

Problema: Listar os pedidos que têm mais de três produtos.

Diagrama:



Sintaxe:

```
SELECT numero_pedido,  
total_produtos = COUNT(*)  
FROM item_de_pedido  
GROUP BY numero_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.

Recuperação de Dados de Várias Tabelas (JOINS)

Até agora trabalhamos com a recuperação de dados em 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, muitas vezes temos a necessidade de acessar simultaneamente várias tabelas relacionadas entre si. Algumas dessas consultas necessitam realizar uma junção (JOIN) entre tabelas, para que dessa junção possa extrair as informações necessárias à consulta formulada.

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.descricao`

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 através de chaves estrangeiras, como vimos em álgebra relacional, e isso implica na existência de colunas com o mesmo nome em tabelas diferentes.

Existem duas sintaxes que vamos considerar no estudo:⁵ a sintaxe ANSI SQL e a do Microsoft® SQL Server™ 2008 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™ 2008

```
SELECT < nome_da_tabela.nome_da_coluna  
[nome_da_tabela.nome_da_coluna....]>  
FROM <nome_da_tabela, nome_da_tabela>  
ON <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* conforme o processado, escolhendo a palavra que o identifica.

Quando usamos INNER JOIN como tipo de *join*, são incluídas somente as linhas que satisfazem a condição do *join*.

Quando usamos CROSS JOIN, incluímos cada uma das 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 Microsoft® SQL Server™ 2008 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.

Nos *joins* do Microsoft® SQL Server™ 2008 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.

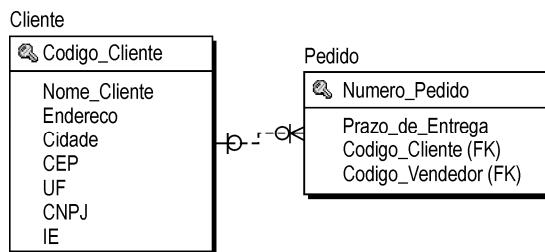
Os operadores a seguir são usados como *join operators* no Microsoft® SQL Server™ 2008:

⁵ Vale a pena destacar que existe um padrão de sintaxe ANSI e um padrão de sintaxe Microsoft® SQL Server™ para a execução de *joins*, mas não podemos usar os dois simultaneamente.

Símbolo	Significado
=	Igual a
>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual
<>	Diferente

Problema: Ver os pedidos de cada cliente.

Diagrama:



Inner Joins

Sintaxe ANSI SQL:

```

SELECT Cliente.nome_cliente,
       pedido.codigo_cliente,
       pedido.numero_pedido
FROM cliente INNER JOIN pedido
ON cliente.codigo_cliente
= pedido.codigo_cliente

```

Sintaxe Microsoft® SQL Server™ 2008:

```

SELECT Cliente.nome_cliente,
       pedido.codigo_cliente,
       pedido.numero_pedido
FROM cliente, pedido
ON cliente.codigo_cliente
= pedido.codigo_cliente

```

Resultado:

Nome_do_cliente	Pedido.Codigo_do_cliente	Pedido.numero_pedido
Ana	720	97
Ana	720	101
Ana	720	137
Ana	720	148
Flávio	870	189
Jorge	110	104
Mauricio	830	203
Rodolfo	410	121
Rodolfo	410	98

Nome_do_cliente	Pedido.codigo_do_cliente	Pedido.numero_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

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**.

Cross Join ou Produto Cartesiano

Problema: Juntar clientes com pedidos.

Sintaxes ANSI SQL e Microsoft® SQL Server™ 2008 são idênticas.

```
SELECT nome_cliente,
       pedido.codigo_cliente,
       numero_pedido
  FROM cliente CROSS JOIN pedido
```

Resultado:

Nome_do_cliente	Pedido.codigo_do_cliente	Pedido.numero_pedido
Ana	720	97
Ana	260	111
Ana	870	54
Ana	390	119
Ana	260	103
Ana	830	203
Ana	410	121
Ana	110	104
Ana	180	105
Ana	720	83
Ana	290	108
Ana	410	89
Flávio	720	97

Nome_do_cliente	Pedido.codigo_do_cliente	Pedido.numero_pedido
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.

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 as linhas de uma tabela que estão relacionadas com as da outra tabela e quais linhas não estão.

Exemplificando no mundo real, poderíamos dizer que queremos ver quais clientes têm 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 ANSI SQL possui três tipos de qualificador 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 quanto da segunda tabelas.

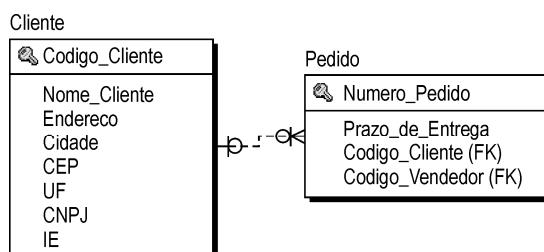
O Microsoft® SQL Server™ 2008 utiliza os mesmos operadores para OUTER JOIN:

*=	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 entender melhor a sua funcionalidade.

Problema: Quais são os clientes que têm pedido e os que não têm pedido?

Diagrama:



Sintaxe ANSI SQL e Microsoft® SQL Server™ 2008:

```
SELECT nome_cliente,
       pedido.codigo_cliente,
       numero_pedido
  FROM cliente LEFT OUTER JOIN pedido
  ON cliente.codigo_cliente =
     Pedido.codigo_cliente
```

Resultado:

Nome_do_cliente	Pedido.Codigo_do_cliente	Pedido.numero_pedido
Ana	720	97
Ana	720	101
Ana	720	137
Ana	720	148
Flávio	870	189
Jorge	110	104
Mauricio	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	138
Renato	290	108
Sebastião	390	119
Lúcia	NULL	NULL
Edmar	NULL	NULL
Paulo	NULL	NULL
Jose	NULL	NULL

Como os clientes Lúcia, Edmar, Paulo e José não têm nenhuma linha da tabela pedido, a informação de seu código é apresentada como NULL.

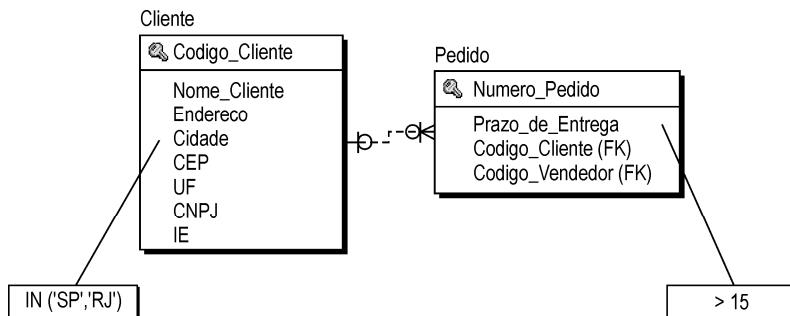
Por este motivo não devemos utilizar NULL na condição de seleção, pois teremos de utilizar os resultados mais imprevisíveis e imagináveis possíveis.

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 de tabelas.

Vamos estudar comandos de seleção de dados com esses operadores.

Problema: Quais clientes têm pedidos com prazo de entrega superior a 15 dias e que pertencem aos estados de São Paulo ('SP') ou Rio de Janeiro ('RJ')?

Diagrama:



Sintaxe:

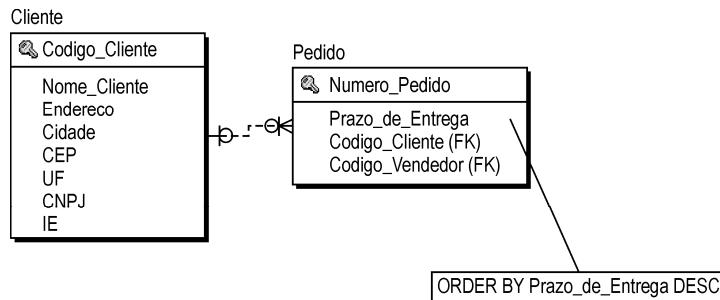
```
SELECT Cliente.nome_cliente,
       pedido.codigo_cliente,
       pedido.numero_pedido
  FROM cliente INNER JOIN pedido
    ON cliente.codigo_cliente = pedido.codigo_cliente
 WHERE UF IN ('SP', 'RJ') AND prazo_de_entrega > 15
```

Resultado:

Nome do Cliente	UF	Prazo de 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:



Sintaxe ANSI SQL e Microsoft® SQL Server™ 2008:

```
SELECT nome_cliente, prazo_de_entrega
FROM cliente, pedido
ON cliente.codigo_cliente = pedido.codigo_cliente
ORDER BY prazo_de_entrega desc;
```

Resultado:

Nome do Cliente	Prazo de 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

Uso de Aliases

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 que tenham salários fixos iguais ou superiores a R\$ 1.000,00.

Sintaxe:

```
SELECT nome_vendedor, prazo_de_entrega
FROM vendedores V, pedido P
WHERE salario_fixo >= 1000.00 AND
prazo_de_entrega > 15 AND
V.codigo_vendedor = P.codigo_vendedor
ORDER BY nome_vendedor;
```

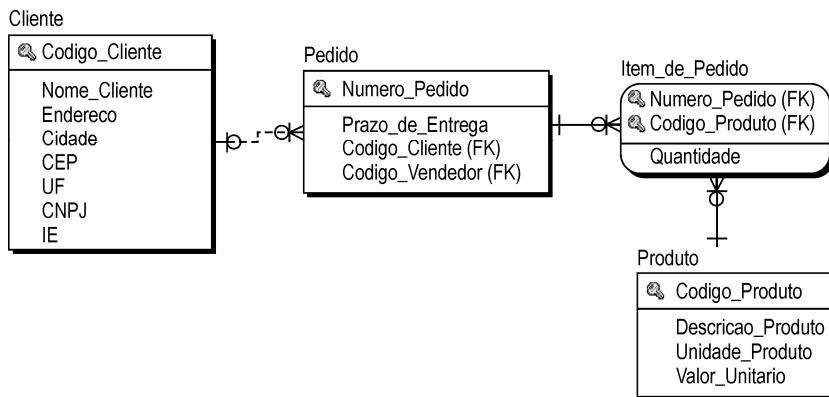
Resultado:

Nome do Vendedor	Prazo de Entrega
Antonio	20
Carlos	30
João	20
José	20
Maurício	30

Junção de Mais de Duas Tabelas

Problema: Mostrar os clientes (ordenados) que têm prazo de entrega maior que 15 dias para o produto 'QUEIJO' e que sejam do Rio de Janeiro.

Diagrama:



Novamente vamos apresentar a sintaxe ANSI e a sintaxe específica do Microsoft® SQL Server™ 2008.

Sintaxe ANSI e Microsoft® SQL Server™ 2008:

```
SELECT cliente.nome_cliente,
FROM cliente INNER JOIN pedido
ON cliente.codigo_cliente = pedido.codigo_cliente
INNER JOIN item_de_pedido
ON pedido.numero_pedido = item_de_pedido.numero_pedido
INNER JOIN produto
ON item_de_pedido.codigo_produto = produto.codigo_produto
WHERE pedido.prazo_de_entrega > 15 AND
      produto.descricao_produto = 'queijo' AND
      cliente.UF = 'RJ'
ORDER BY cliente.nome_cliente
```

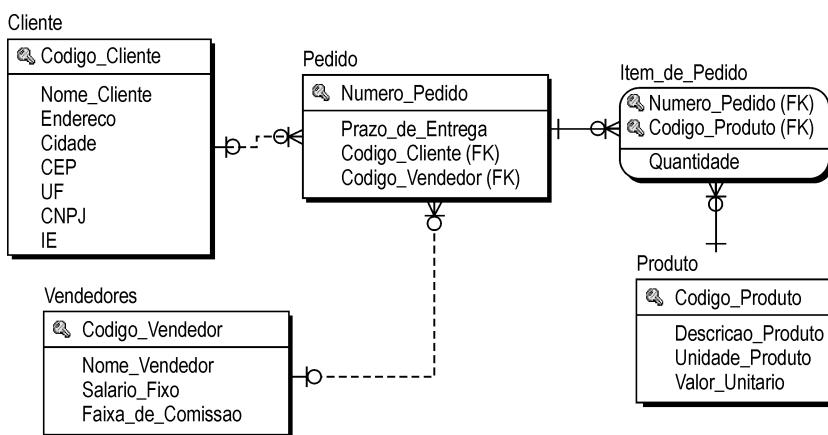
Resultado:

Nome Cliente
Ana
Rodolfo
Susana

Vamos a mais um exemplo.

Problema: Mostrar todos os vendedores que venderam chocolate em quantidade superior a dez quilos.

Diagrama:



Sintaxe:

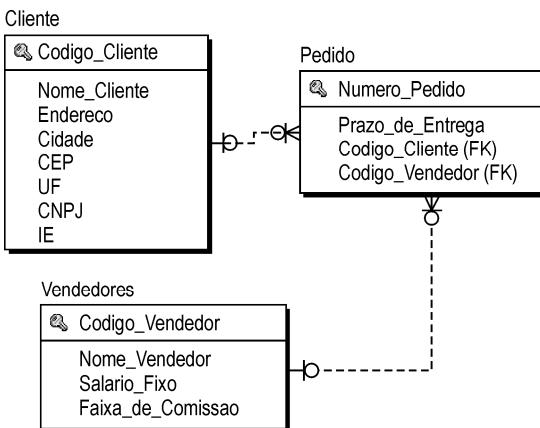
```
SELECT DISTINCT nome_vendedor
FROM vendedores, pedido, item_de_pedido, produto
WHERE Vendedores.codigo_vendedor = Pedido.codigo_vendedor AND
      Pedido.numero_pedido = Item_de_pedido.numero_pedido AND
      Item_de_pedido.codigo_produto = Produto.codigo_produto AND
      quantidade > 10 AND descricao_produto = 'Chocolate';
```

Resultado:

Nome do Vendedor
José
Carlos

Problema: Quantos clientes fizeram pedido com o vendedor João?

Diagrama:



Sintaxe Microsoft® SQL Server™ 2008:

```
SELECT COUNT (codigo_cliente)
FROM cliente, pedido, vendedores
WHERE Cliente.codigo_cliente = Pedido.codigo_cliente AND
Pedido.codigo_vendedor = Vendedores.codigo_vendedor AND
Vendedores.nome_vendedor = 'Joao';
```

Resultado:

COUNT (Código Cliente)
4

Problema: Quantos clientes das cidades do Rio de Janeiro e Niterói tiveram seus pedidos tirados com o vendedor João?

Sintaxe Microsoft® SQL Server™ 2008:

```
SELECT cidade, numero = COUNT (nome_cliente)
FROM cliente, pedido, vendedores
WHERE nome_vendedor = 'Joao' AND
CIDADE IN ('Rio de Janeiro', 'Niteroi') AND
Vendedores.codigo_vendedor = Pedido.codigo_vendedor AND
Pedido.codigo_cliente = Cliente.codigo_cliente
GROUP BY cidade;
```

Resultado:

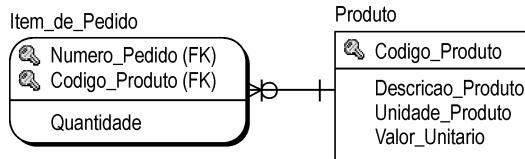
Cidade	Número
Niterói	2
Rio de Janeiro	1

Utilização de Consultas Encadeadas (Subqueries)

O que é *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:



Sintaxe:

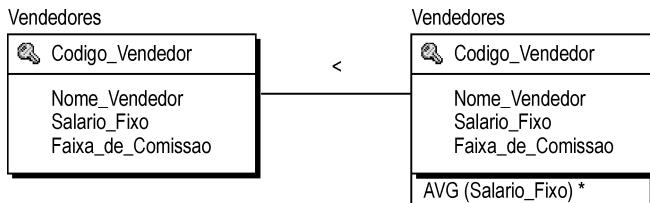
```
SELECT descricao_produto
  FROM produto
 WHERE codigo_produto IN
 (SELECT codigo_produto
    FROM item_de_pedido
   WHERE quantidade = 10)
```

Resultado:

Descrição
Queijo
Vinho
Linho

Problema: Utilizando AVG - Quais vendedores ganham um salário fixo abaixo da média?

Diagrama:



Sintaxe:

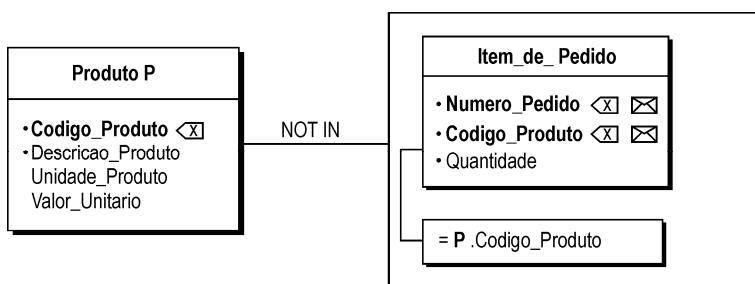
```
SELECT nome_vendedor
  FROM vendedores
 WHERE salario_fixo <
 (SELECT AVG(salario_fixo)
    FROM vendedores);
```

Resultado:

Nome do Vendedor
José
Carlos
João
Jonas
Josias
Mauricio

Problema: Utilizando NOT IN - Quais os produtos que não estão presentes em nenhum pedido?

Diagrama:



Sintaxe:

```
SELECT código_produto, descrição_produto
FROM produto
WHERE código_produto NOT IN
(SELECT * FROM item_de_pedido
WHERE item_de_pedido.código_produto = produto.código_produto)
```

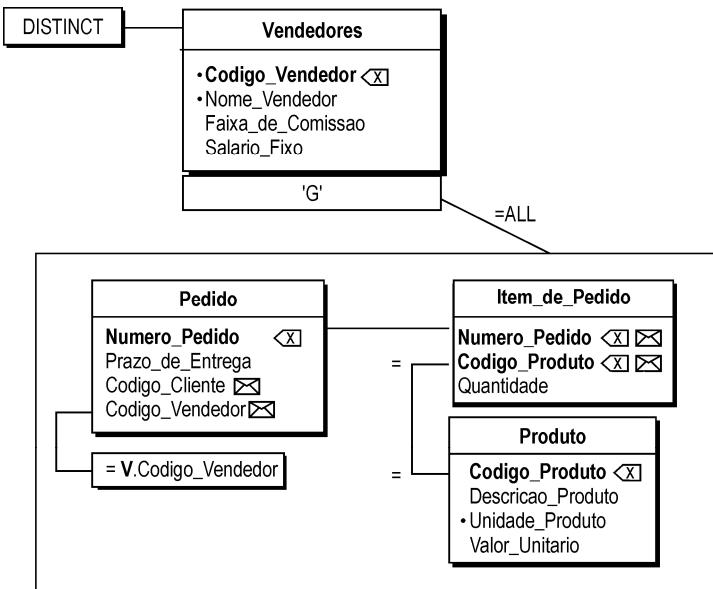
Resultado:

Código do Produto	Descrição do Produto
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:



Sintaxe:

```
SELECT DISTINCT codigo_vendedor, nome_vendedor
FROM vendedores
WHERE unidade ALL =('G')
(SELECT unidade_produto
FROM pedido, item_de_pedido, produto
WHERE Pedido.numero_pedido = item_de_pedido.numero_pedido
AND Item_de_pedido.codigo_produto = Produto.codigo_produto
AND Pedido.codigo_vendedor = Vendedores.codigo_vendedor);
```

Resultado:

Código do Vendedor	Nome do Vendedor
720	Felipe
310	Josias

Interpretando o comando aplicado:

- Selecione todas as ocorrências da tabela vendedores que estão associadas à tabela pedidos e na tabela item de pedido relacionada com a tabela pedidos, relacionada com a tabela produtos, todos os itens de pedido sejam de produtos com unidade 'G', ou seja, Grama (não existe correspondente no Microsoft® SQL Server™ 2008).

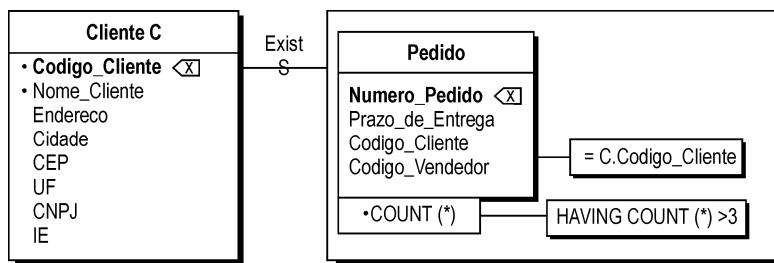
Condição que satisfaz a query:

- ▶ Todos os itens de pedido de um pedido têm unidade = G.
- ▶ Mostrar o nome dos vendedores associados a esses pedidos.

Esse comando é complicado de entender, por isso tentamos uma forma de explicá-lo em linguagem mais natural, mas mesmo assim talvez não tenhamos conseguido esclarecer completamente.

Problema: Subquery testando a existência - Quais clientes estão presentes em mais de três pedidos?

Diagrama:



Sintaxe:

```
SELECT nome_cliente
FROM cliente
WHERE EXIST
(SELECT COUNT(*)
FROM pedido
WHERE código_cliente = cliente.codigo_cliente
HAVING COUNT(*) >3);
```

Resultado:

Nome Cliente
Ana
Susana

Entendendo o comando:

- ▶ Sempre analisamos a query entre parênteses primeiro.
- ▶ Nesse 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).

Destaca-se aqui a diferença entre o comando select com EXIST e com NOT IN (IN):

- ▶ 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 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, ele falha.

Sintaxe Básica Microsoft® SQL Server™ 2008

```
SELECT <lista de colunas>
INTO <nova tabela>
FROM < lista de tabelas>
WHERE <condições de seleção>
```

No Microsoft® SQL Server™ 2008 a cláusula INTO cria uma tabela permanente apenas se a opção **select into/bulkcopy** estiver setada; caso contrário, cria uma tabela temporária.

Se as colunas selecionadas na lista de colunas não possuírem nomes (SELECT *), as colunas criadas na nova tabela também não vão possuir nomes e somente poderão ser selecionadas por meio de um SELECT * FROM <nome da tabela>.

Inserir, Modificar e Apagar Registros

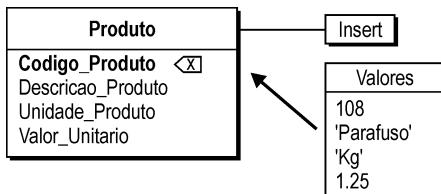
Adição de Registro à Tabela

Sintaxe Básica

```
INSERT INTO <nome da tabela>
(<nome da(s) coluna(s)>)
VALUES (<valores>);
```

Problema: Adicionar o produto 'parafuso' à tabela produto.

Diagrama:



Sintaxe:

```
INSERT INTO produto  
VALUES (108, 'Parafuso', 'Kg', 1.25);
```

A cláusula VALUE especifica os dados que você deseja inserir na tabela. Ela é uma palavra requerida para introduzir no comando uma lista de valores para cada coluna.

Se não forem especificados os nomes de colunas, essa lista de valores deve estar na ordem das colunas definidas no comando CREATE TABLE.

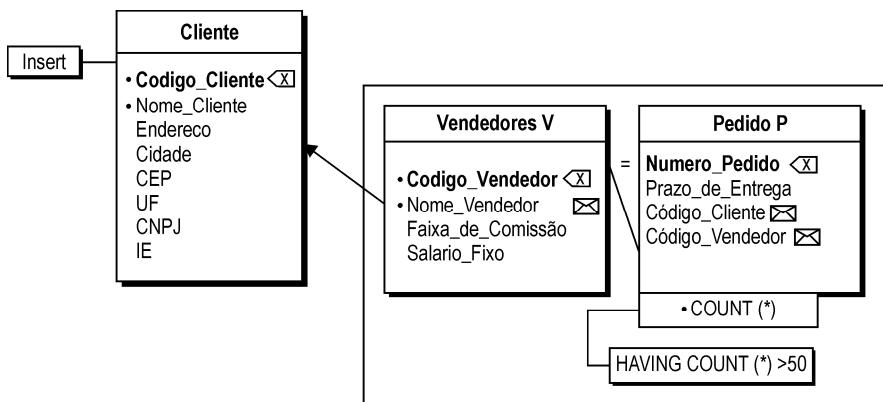
Adição de Registros com um SELECT

Sintaxe Básica

```
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 clientes os vendedores que emitiram mais de 50 pedidos. Usar para código de cliente o mesmo código de vendedor.

Diagrama:



Sintaxe ANSI 92:

```
INSERT INTO cliente (codigo_cliente, nome_cliente)
SELECT codigo_vendedor, nome_vendedor, COUNT(*)
FROM vendedores, pedido
WHERE Vendedores.codigo_vendedor = Pedido.codigo_vendedor
HAVING COUNT(*) > 50;
```

Sintaxe Microsoft® SQL Server™ 2008:

```
INSERT cliente (codigo_cliente, nome_cliente)
SELECT codigo_vendedor, nome_vendedor, COUNT(*)
FROM vendedores, pedido
WHERE Vendedores.codigo_vendedor = Pedido.codigo_vendedor
HAVING COUNT(*) > 50;
```

A diferença de sintaxe se resume somente à existência da cláusula INTO no ANSI 92, e cumpre destacar que o Microsoft® SQL Server™ 2008 não exige a cláusula INTO, porém não causa nenhum erro se você a utilizar.

Atualização de um Registro - UPDATE

A atualização de dados em linhas existentes na tabela permite que:

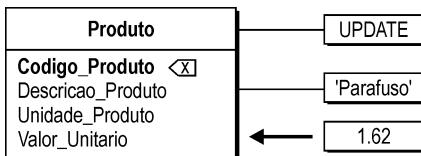
- ▶ Especifique-se uma determinada coluna e altere-se seu valor.
- ▶ Seja indicada uma linha específica ou uma condição de identificação de linhas para que sejam alterados os valores de determinadas colunas.

Sintaxe Básica

```
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:



Sintaxe:

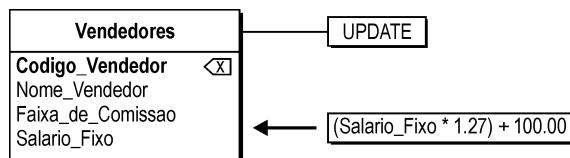
```
UPDATE produto
    SET valor_unitario = 1.62
    WHERE descricao_produto = 'Parafuso';
```

Resultado:

Código do Produto	Descrição	Unidade	Valor Unitário
108	parafuso	Kg	1.62

Problema: Atualizar o salário fixo de todos os vendedores em 27% mais uma bonificação de R\$ 100,00.

Diagrama:



Sintaxe:

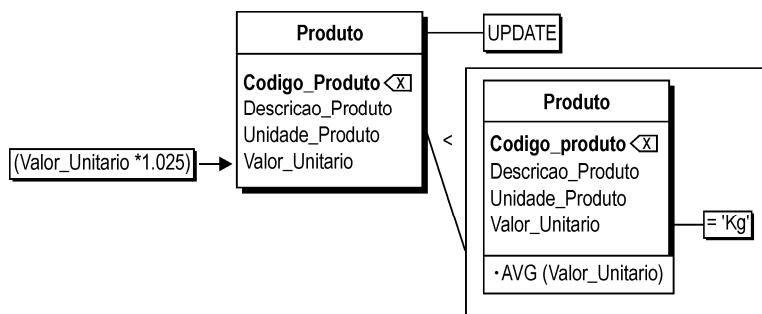
```
UPDATE vendedores  
SET salario_fixo = (salario_fixo * 1.27) + 100.00;
```

O resultado desse comando provoca que todos os vendedores tenham o mesmo valor de salário fixo.

A sintaxe é idêntica para Microsoft® SQL Server™ 2008.

Problema: Acrescentar 2,5% ao preço unitário dos produtos que estejam abaixo da média dos preços para os comprados a quilo.

Diagrama:



Sintaxe:

```
UPDATE produto  
SET valor_unitario = valor_unitario * 1.025  
WHERE valor_unitario <  
(SELECT AVG(valor_unitario)  
FROM produto  
WHERE unidade_produto = 'Kg');
```

Alteração de Registros com Dados de Outra Tabela

Para explicar esse comando, vamos supor que nossa tabela de produtos possua um campo de vendas acumuladas ao ano.

Problema: Atualizar as vendas acumuladas ao ano para cada produto.

Diagrama:

Produto
Codigo_Produto
Descricao
Unidade_Produto
Valor_Unitario
Vendas_Acumuladas

Sintaxe:

```
UPDATE produto
SET vendas_acumuladas = 0
UPDATE produto
SET vendas_acumuladas = (SELECT SUM (quantidade)
                           FROM item_de_pedido)
```

Resultado:

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

Apagar Registros da Tabela

Sintaxe Básica

```
DELETE FROM <nome da tabela>
WHERE <condição>;
```

Problema: Apagar todos os vendedores com faixa de comissão nula.

Diagrama:

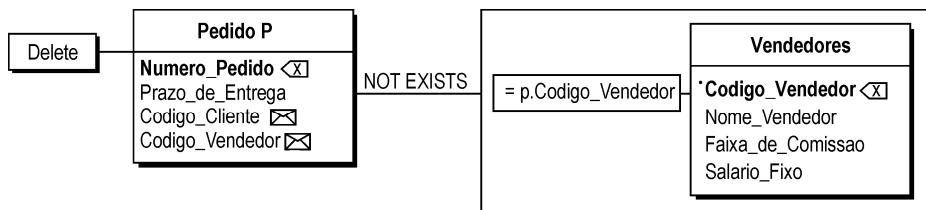


Sintaxe:

```
DELETE FROM vendedores
WHERE faixa_de_comissao IS NULL;
```

Problema: Apagar todos os registros de pedidos realizados por vendedores fantasmas (operação caça-fantasma).

Diagrama:



Sintaxe:

```
DELETE FROM pedido
WHERE NOT EXIST
(SELECT vendedor
FROM vendedores
WHERE codigo_vendedor = Pedido.codigo_vendedor);
```

Apagar Registros da Tabela com Base em Dados de Outra Tabela

Problema: Apagar todos os registros de *item de pedidos* realizados para produtos que tenham "lh" no nome.

Sintaxe ANSI 92:

```
DELETE FROM item_de_pedido
  WHERE pedido.codigo_produto IN
    (SELECT * FROM produto, item_de_pedido
     WHERE produto.codigo_produto = item_de_pedido.codigo_produto
       AND produto.descricao_produto LIKE %lh%)
```

Sintaxe Microsoft® SQL Server™ 2008:

```
DELETE FROM item_de_pedido
  FROM produto P, item_de_pedido I
 WHERE P.codigo_produto = I.codigo_produto
   AND produto.descricao_produto LIKE %lh%)
```

Utilização de Views

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. As views também podem ser utilizadas para informação calculada ou derivada, como preço total de um item de pedido que é calculado pela 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. Estas são chamadas de views (tabelas virtuais).

Como as tabelas reais, as views devem ser criadas:

Sintaxe Básica

```
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];
```

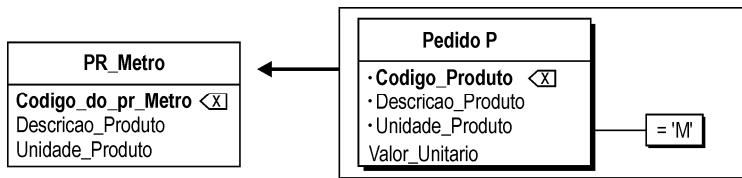
As views são utilizadas para se ter uma particular visão de uma tabela, para que não seja necessária a utilização do conjunto como um todo.

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:

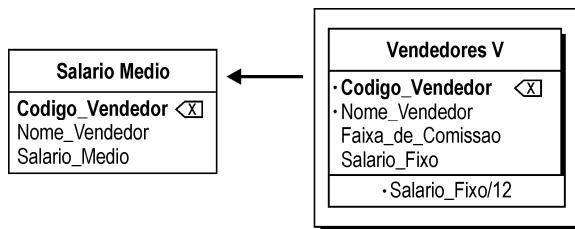


Sintaxe:

```
CREATE VIEW PR_metro (codigo_PR_metro, descricao, unidade) AS
SELECT codigo_produto, descricao_produto, unidade_produto
FROM produto
WHERE unidade_produto = 'M';
```

Problema: Criar uma view contendo o código do vendedor, o seu nome e o salário fixo médio no ano.

Diagrama:



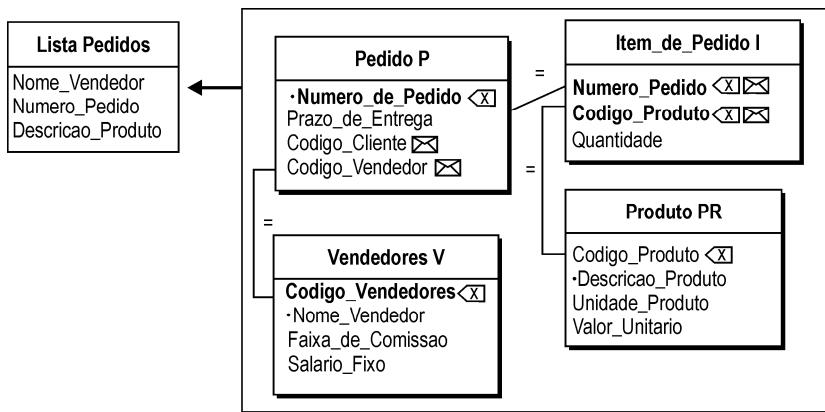
Sintaxe:

```
CREATE VIEW salario_medio (codigo_vendedor, nome_vendedor, salario_medio) AS
SELECT codigo_vendedor, nome_vendedor, salario_fixo/12
FROM vendedores;
```

Criação de uma View por meio de um Join

Problema: Criar uma view contendo os vendedores, seus pedidos efetuados e os respectivos produtos.

Diagrama:



Sintaxe:

```
CREATE VIEW lista_pedidos AS
    SELECT nome_vendedor, numero_pedido, descricao
    FROM vendedor V, pedido P, item_pedido I, produto PR
    WHERE V.codigo_vendedor = P.codigo_vendedor and
        P.numero_pedido = I.numero_pedido and
        I.codigo_produto = PR.codigo_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.).

Utilização de uma View

Listagem

Problema: Com base na view salario_medio mostrar os vendedores que possuem média salarial superior a R\$ 2.000,00.

Diagrama:



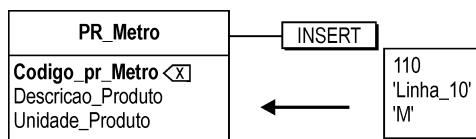
Sintaxe:

```
SELECT nome_vendedor, salario_medio
FROM salario_medio
WHERE salario_medio > 2000.00;
```

Inserção de Linhas numa View

Problema: Inserir o registro: 110, Linha_10, M; na view PR_Metro.

Diagrama:



Sintaxe:

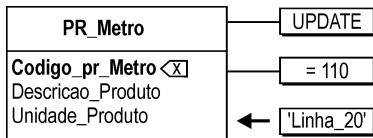
```
INSERT INTO pr_metro
VALUES (110, 'Linha_10', 'M');
```

É importante observar que a inserção de linhas em uma view causa a inserção da linha na tabela à qual a view está associada, ou seja, à tabela origem da view.

Modificação de uma Linha da View

Problema: Alterar a descrição de 'Linha_10' para 'Linha_20' no código 110 da view PR_Metro.

Diagrama:



Sintaxe:

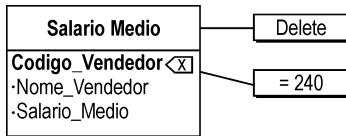
```
UPDATE pr_metro
    SET descricao = 'Linha_20'
  WHERE codigo_pr_metro = 110;
```

Da mesma forma vamos alterar a tabela original da view.

Apagar

Problema: Apagar da view salario_medio o registro de código do vendedor igual a 240.

Diagrama:



Sintaxe:

```
DELETE FROM salario_medio
  WHERE codigo_vendedor = 240;
```

Eliminação de uma View

Sintaxe Básica

```
DROP VIEW <nome da VIEW>;
```

Problema: Eliminar a view salario_medio.

Sintaxe:

```
DROP VIEW salario_medio;
```

Garantia dos Privilégios de Acesso - GRANT e REVOKE

Muitos sistemas de banco de dados relacionais podem ser acessados por diversos usuários. Cada usuário tem 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.

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 Básica

```
GRANT {ALL |Lista de privilégios}
ON {nome da tabela/view [lista de colunas]}
TO {PUBLIB |lista de usuários}
[WITH GRANT OPTION]
```

A palavra ALL é usada quando qualquer privilégio é aplicável ao objeto, ou então especificamos o privilégio 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 privilégio.

Importante: Somente pode ser utilizada uma tabela ou view por comando.

As diferentes versões de SQL dos SGBDs relacionais incluem diferentes conjuntos de privilégios. SELECT, UPDATE, DELETE e INSERT estão sempre presentes em todos esses conjuntos, indiferentemente do SGBD.

Os privilégios podem ser especificados para algumas colunas, porém devem ser todas elas da mesma tabela. Se não for especificada nenhuma coluna, os privilégios valem para todas.

A cláusula opcional WITH GRANT OPTION permite que quando se dá o privilégio a um usuário, ele o passe a outros usuários.

Lista de Opções de Privilégios

- ▶ *Select*: pode executar uma consulta à tabela.
- ▶ *Insert*: permite executar uma inserção na tabela.
- ▶ *Delete*: pode apagar registros da tabela.
- ▶ *Update*: modifica registros na tabela.
- ▶ *All privileges/all*: pode executar qualquer operação com a tabela.
- ▶ <usuário>: nome do usuário que vai receber os privilégios. Deve ser cadastrado dentro do ambiente.
- ▶ *PUBLIC*: concede os privilégios especificados a todos os usuários do ambiente.

Exemplos:

```
GRANT SELECT ON Produto TO Felipe;
```

- ▶ Permite só consultas ao usuário Felipe na 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 Mauricio e Felipe o privilégio de seleção na tabela CLIENTE.

Problema: Disponibilizar para seleção a view salario_medio a todos os usuários.

Sintaxe:

```
GRANT SELECT ON salario_medio TO public;
```

Utilizando o comando GRANT, é possível 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 (codigo_vendedor, nome_vendedor)
ON vendedores
TO public;
```

Podemos passar nossa concessão de privilégios a outros usuários por meio da cláusula WITH GRANT OPTION, como explicamos anteriormente

Problema: Dar ao usuário FELIPE o poder de concessão de todos os privilégios a outros usuários sobre a tabela PEDIDO.

Sintaxe:

```
GRANT ALL
ON pedido
TO Felipe
WITH GRANT OPTION
```

Comando REVOKE (Revogação)

Da mesma forma que o criador da tabela pode garantir (GRANT) privilégios de acesso aos outros usuários, ele pode revogar esses privilégios por meio do comando REVOKE:

Sintaxe Básica

```
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
```

Trabalho 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. Com os ponteiros se localiza a linha em que o valor desejado está armazenado.

As tabelas de índices são utilizadas internamente pelo SGBD, ficando totalmente transparente ao usuário sua utilização.

Um Checklist para Criação de Índices

Quando vamos criar índices em colunas, devemos considerar:

- ▶ O índice deve ser de coisas que vamos pesquisar com frequência.
- ▶ Indexe as suas chaves estrangeiras quando precisar de *joins* mais eficientes e performáticos.

- ▶ As colunas que são regularmente utilizadas em *joins* devem ser indexadas, porque o sistema pode executar esses *joins* de modo muito mais rápido, e como tempo de resposta é algo vital no desenvolvimento de projetos físicos de bancos de dados, é recomendável a sua utilização.
- ▶ Crie índices sempre em colunas que são pesquisas por um intervalo de valores.
- ▶ Por fim, crie índices 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 nessa coluna sempre vai retornar uma alta quantidade de linhas. Como é um índice não agrupado, pode ocupar muito espaço e não ser utilizado para consultas.

O índice *cluster* ou agrupado tem os valores constantes na coluna e pode ser agrupado em pequenos conjuntos, 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. O objetivo aqui é somente conceituar. 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 colocados na mesma ordem física das linhas da tabela, e índices não agrupados ou *nonclustered* são dispostos na ordem lógica das linhas da tabela.

Vamos ver como se criam índices.

Criação de Índices

Cada índice é aplicado a uma tabela, especificando uma ou mais colunas dessa tabela.

Índices são criados pelo comando CREATE INDEX que cria um índice com as colunas de uma tabela.

Sintaxe Básica

```
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 ANSI SQL.

Exemplos:

```
CREATE INDEX nome_pro  
ON produto (descricao_produto);
```

- ▶ Cria a tabela de índices chamada nome_pro baseada no campo descrição da tabela produto.

```
CREATE INDEX ped_pro  
ON item_produto (numero_pedido, codigo_produto);
```

- ▶ Cria a tabela de índices ped_pro baseada na concatenação dos campos numero_pedido e codigo_produto da tabela item_de_pedido.

É importante considerar que praticamente todas as sintaxes em se tratando de SGBDs relacionais exigem que se identifique o database proprietário da tabela, principalmente em Microsoft® SQL Server™ 2008.

```
CREATE UNIQUE INDEX clientex  
ON [nome do database] cliente (codigo_cliente);
```

- ▶ Cria o índice único para a tabela cliente baseada no código do cliente, não podendo haver duplicidade de informação armazenada.

Eliminação de Índices

Da mesma forma que um índice é criado ele pode ser eliminado, dependendo das necessidades do projeto do banco de dados.

Sintaxe Básica

```
DROP index <nome do indice>;
```

Exemplos:

```
DROP index nome_pro;  
DROP index ped_pro;
```

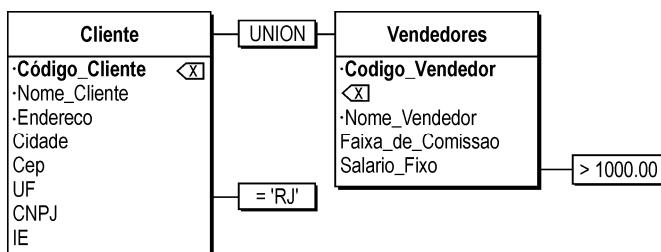
Tópicos Avançados de SQL

Combinação de Resultados de Pesquisas (UNION)

Eventualmente, é necessário combinar os resultados de duas ou mais consultas feitas sobre tabelas. Para realizar essa operação, a linguagem SQL oferece o operador UNION e uma listagem com os resultados das consultas combinadas.

Problema: Listar nomes e códigos dos vendedores que têm salário fixo maior que R\$ 1.000,00 e clientes que residem no Rio de Janeiro.

Diagrama:



Sintaxe:

```
SELECT codigo = codigo_cliente, nome = nome_cliente
FROM cliente
WHERE UF = 'RJ'
UNION
SELECT codigo_vendedor, nome_vendedor
FROM vendedores
WHERE salario_fixo > 1000.00
```

Observe que as duas listas de colunas dos SELECT contêm o mesmo número de itens (duas colunas) e os tipos de dados são compatíveis.

Resultado:

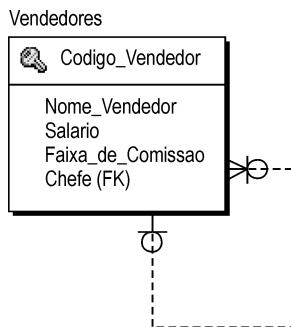
Código	Nome
209	José
111	Carlos
11	João
240	Antônio
720	Felipe
213	Jonas
101	João
250	Maurício
410	Rodolfo

Realização de um *Join* entre uma Tabela e Ela Mesma

Muitas vezes, quando temos autorrelacionamentos, necessitamos apresentar os seus dados. Para isso utilizamos um *join* entre a tabela e ela mesma, é o *self join*.

O autorrelacionamento acontece quando temos em uma tabela uma chave estrangeira que está relacionada com a chave primária dessa mesma tabela.

Vamos ampliar o exemplo até o momento e colocar na tabela de vendedores o chefe de cada um, que também é um vendedor.



Vamos preencher a tabela com os chefes de cada vendedor:

Nova tabela vendedores:

Código do Vendedor	Nome do Vendedor	Salário Fixo	Faixa de Comissão	Chefe
209	José	1.800,00	C	250
111	Carlos	2.490,00	A	720
11	João	2.780,00	C	250
240	Antônio	9.500,00	C	250
720	Felipe	4.600,00	A	null
213	Jonas	2.300,00	A	720
101	João	2.650,00	C	250
310	Josias	870	B	720
250	Maurício	2.930,00	B	null

Observe que as linhas correspondentes aos dois vendedores que são chefes de outros possuem a chave estrangeira com valor nulo, pois não possuem nenhum chefe acima deles.

Para realizar um *self join*, você precisa utilizar o conceito de alias das tabelas.

Então, para listarmos o nome dos funcionários e de seus chefes:

```

Select A.Nome_Vendedor as Chefe,
B. Nome_Vendedor as Funcionario
From Funcionario A, Funcionario B
Where A.Codigo_vendedor = B.Chefe
Order by 1

```

Resultado:

Chefe	Funcionario
Felipe	Carlos
Felipe	Jonas
Felipe	Josias
Maurício	José
Maurício	João
Maurício	Antônio
Maurício	João

NVL

Converte valores nulos em um valor real. A função está apta a trabalhar com os tipos de dados DATE, CHARACTER e NUMBER. Os parâmetros informados devem ser do mesmo tipo de dado.

A utilização de NVL é importante para o controle de valores nulos quando da utilização de uma expressão de cálculo, evitando-se divisões por zero, por exemplo.

Frequentemente nos deparamos com algum select de banco de dados em que alguns campos retornam null.

O mais comum é que o tratamento desta situação seja feito do lado da aplicação, gerando um aumento de código.

Entretanto, para facilitar a vida do programador, no Oracle, a função que faz esse serviço é a NVL. Ela recebe dois parâmetros; o primeiro é o nome do campo, ou expressão, que queremos tratar e o segundo parâmetro é o valor que será retornado quando for encontrado o null no campo escolhido.

Supondo uma tabela TESTE de exemplo:

codigo	descricao	valor
1	codigo 1	1
2	codigo 2	2
3	null	null

Podemos fazer o tratamento da Descrição e do Valor.

Select Código, NVL (Descrição, 'Inexiste'), NVL (Valor, 0.0) from teste;

O retorno seria então:

codigo	descricao	valor
1	codigo 1	1
2	codigo 2	2
3	Inexiste	0

A utilização da expressão condicional DECODE

A expressão DECODE trabalha de um modo similar à lógica IF-THENELSE. Ela compara *uma expressão ou o conteúdo de uma coluna* a todas as condições, uma por vez. Se a expressão é igual à condição, então o Oracle retorna o resultado correspondente. Se não encontrar nenhuma correspondência, o Oracle retorna o valor *default*. Neste caso, se *default* estiver omitido, Oracle retorna Nulo.

```
DECODE  (expr
          ,search1,result1
          ,search2,result2,...,
          ,searchN,resultN]
          ,default)
```

Exemplos:

```
Select  funcao,
        salario,
        DECODE(funcao, 'IT_PROG', salario*1.1,
                'ST_MAN', salario*1.2,
                'MK_REP', salario*1.3,
                salario) reajuste
  from    empregados

select
  DECODE (PD.CLIENTE, 'S', VAREJO, 'ATACADO') cliente
from Pedidos PD
Where  trunc(PD.DIA_DE_DISTRIBUICAO)      >= '07/set/2011'
```

A utilização da expressão condicional CASE

Em uma expressão CASE, o Oracle pesquisa a partir da primeira cláusula WHEN, em que 'expr' é igual a compare e retorna result. Se nenhuma das cláusulas WHEN for selecionada e uma cláusula ELSE existir, então o Oracle retorna resultElse.

```
CASE 'expressão' WHEN 'valor de comparação1' THEN 'resultado1'
  WHEN 'valor de comparaçãoN' THEN 'resultado2'
  WHEN 'valor de comparaçãoN' THEN 'resultadoN'
  ELSE 'resultado de Else'
END
```

```
select
  funcao,
  salario,
```

```

CASE funcao
    WHEN 'IT_PROG' THEN salario * 1.1
    WHEN 'ST_MAN'  THEN salario * 1.2
    WHEN 'MK_REP'  THEN salario * 1.3
ELSE salario
END
from empregados

```

Trabalho com tempo (campos date)

Como todo desenvolvedor Oracle sabe, os campos do tipo **DATE** costumam ser usados em tabelas para a representação de informação referente à **data** e à **hora** de algum evento. Campos do tipo DATE ocupam apenas **7 bytes** e são capazes de armazenar datas com a **precisão** na casa dos **segundos**. Melhor detalhando, um campo DATE é capaz de manter as seguintes informações temporais associadas a um determinado evento: século, ano, mês, dia, hora, minuto e segundo de ocorrência do evento.

Em muitas situações práticas podemos estar interessados em adicionar ou subtrair não dias, mas **horas**, **minutos** ou **segundos** do valor de uma data. Isso também é possível no Oracle, no entanto o SGBD solicita que seja especificada uma “**fração de dia**” adequada para a operação aritmética. Sabemos que um dia possui:

- ▶ 24 horas
- ▶ $24 \times 60 = 1440$ minutos
- ▶ $24 \times 60 \times 60 = 86.400$ segundos

Com isso, se desejamos somar 5 minutos a uma determinada data no Oracle, devemos somar a fração **5/1440** a esta data! Observe este cálculo no exemplo a seguir:

```

SELECT
    TO_CHAR(SYSDATE, 'DD/MM/YYYY HH24:MI:SS') AS AGORA,
    TO_CHAR(SYSDATE + 5/1440, 'DD/MM/YYYY HH24:MI:SS') AGORA_MAIS_5_MIN
FROM DUAL;

```

Agora	Agora mais 5 min
25/06/2007 18:18:29	25/06/2007 18:23

Estudo de Caso - Banco de Dados

Para que você possa comparar as sintaxes dos comandos SQL, nesta revisão vamos apresentar os scripts gerados para o estudo de caso das cirurgias apresentado no livro, agora com o modelo ampliado em detalhes.

Utilizamos o software ERwin 4.0® para os servidores Microsoft® SQL Server™ 2008, assim como realizamos uma geração de script de banco de dados para Oracle™ 11g, que permite avaliar as estruturas de banco de dados após criá-lo em um desses dois SGBDs.

Desta forma você terá à sua disposição duas formas de criação para os bancos de dados físicos dos estudos de caso. Bom proveito, amigo!

Microsoft SQL Server®2008™

```
CREATE TABLE Cidade (
    ID_Cidade           char(18) NOT NULL,
    UF                 char(18) NOT NULL,
    Descricao          char(18) NULL
)
ALTER TABLE Cidade
    ADD PRIMARY KEY (ID_Cidade ASC, UF ASC)
CREATE TABLE Cirurgia (
    CRM                char(18) NOT NULL,
    Numero_Internacao  char(18) NOT NULL,
    Numero_Sala         char(18) NOT NULL,
    Data_Cirurgia       char(18) NOT NULL,
    Hora_Cirurgia       char(18) NOT NULL,
    Identificacao_Especialidade  char(18) NOT NULL,
    Qtde_sangue_solicitada  char(18) NULL,
    Qtde_sangue_utilizada  char(18) NULL
)
ALTER TABLE Cirurgia
    ADD PRIMARY KEY (CRM ASC, Numero_Internacao ASC, Numero_Sala ASC,
                      Data_Cirurgia ASC, Hora_Cirurgia ASC)
CREATE TABLE Consumo (
    Numero_Consumo      char(18) NOT NULL,
   Codigo_Produto       char(18) NOT NULL,
    Hora_Cirurgia        char(18) NOT NULL,
    Data_Cirurgia        char(18) NOT NULL,
    CRM                 char(18) NOT NULL,
    Numero_Internacao   char(18) NOT NULL,
    Numero_Sala          char(18) NOT NULL,
    Quantidade          char(18) NULL
)
ALTER TABLE Consumo
    ADD PRIMARY KEY (Numero_Consumo ASC, Código_Produto ASC)
CREATE TABLE Especialidade (
    Identificacao_Especialidade  char(18) NOT NULL,
    Descricao_Especialidade     char(18) NULL
)
ALTER TABLE Especialidade
    ADD PRIMARY KEY (Identificacao_Especialidade ASC)
CREATE TABLE Estado (
    UF                  char(18) NOT NULL,
    Nome_Estado         char(18) NULL
)
ALTER TABLE Estado
    ADD PRIMARY KEY (UF ASC)
CREATE TABLE Material_e_Medicamento (
    Código_Produto      char(18) NOT NULL,
    Princípio_Ativo     char(18) NULL,
    Descricao            char(18) NULL,
    Unidade              char(18) NULL,
    Saldo_Estoque        char(18) NULL
)
ALTER TABLE Material_e_Medicamento
    ADD PRIMARY KEY (Código_Produto ASC)
CREATE TABLE Medico (
```

```

CRM                                char(18) NOT NULL,
ID_Cidade                          char(18) NOT NULL,
UF                                 char(18) NOT NULL,
Id_Especialidade                  char(18) NOT NULL,
Nome                               char(18) NULL,
Endereco                           char(18) NULL,
Bairro                             char(18) NULL,
Salario                            char(18) NULL
)
ALTER TABLE Medico
ADD PRIMARY KEY (CRM ASC)
CREATE TABLE Paciente (
    Numero_Internacao                char(18) NOT NULL,
    CPF                               char(18) NULL,
    Nome                              char(18) NULL,
    Endereco                          char(18) NULL,
    Sexo                              char(18) NULL,
    Nascimento                        char(18) NULL,
    Profissao                         char(18) NULL
)
ALTER TABLE Paciente
ADD PRIMARY KEY (Numero_Internacao ASC)
CREATE TABLE Qualificacao_da_Sala (
    Numero_Sala                      char(18) NOT NULL,
    Identificacao_Especialidade     char(18) NOT NULL,
    Data_Inicio_qualificacao        char(18) NULL
)
ALTER TABLE Qualificacao_da_Sala
ADD PRIMARY KEY (Numero_Sala ASC, Identificacao_Especialidade ASC)
CREATE TABLE Sala (
    Numero_Sala                      char(18) NOT NULL,
    Andar                            char(18) NULL,
    Ala_do_Centro                    char(18) NULL
)
ALTER TABLE Sala
ADD PRIMARY KEY (Numero_Sala ASC)
ALTER TABLE Cidade
ADD FOREIGN KEY (UF)
REFERENCES Estado (UF)
ALTER TABLE Cirurgia
ADD FOREIGN KEY (Identificacao_Especialidade)
REFERENCES Especialidade (Identificacao_Especialidade)
ALTER TABLE Cirurgia
ADD FOREIGN KEY (Numero_Sala)
REFERENCES Sala (Numero_Sala)
ALTER TABLE Cirurgia
ADD FOREIGN KEY (Numero_Internacao)
REFERENCES Paciente (Numero_Internacao)
ALTER TABLE Cirurgia
ADD FOREIGN KEY (CRM)
REFERENCES Medico (CRM)
ALTER TABLE Consumo
ADD FOREIGN KEY (Codigo_Produto)
REFERENCES Material_e_Medicamento (Codigo_Produto)
ALTER TABLE Consumo
ADD FOREIGN KEY (CRM, Numero_Internacao, Numero_Sala,
Data_Cirurgia, Hora_Cirurgia)
REFERENCES Cirurgia (CRM, Numero_Internacao,
Numero_Sala, Data_Cirurgia, Hora_Cirurgia)
ALTER TABLE Medico
ADD FOREIGN KEY (ID_Cidade, UF)
REFERENCES Cidade (ID_Cidade, UF)

```

```

ALTER TABLE Medico
    ADD FOREIGN KEY (Id_Especialidade)
        REFERENCES Especialidade (Identificacao_Especialidade)
ALTER TABLE Qualificacao_da_Sala
    ADD FOREIGN KEY (Identificacao_Especialidade)
        REFERENCES Especialidade (Identificacao_Especialidade)
ALTER TABLE Qualificacao_da_Sala
    ADD FOREIGN KEY (Numero_Sala)
        REFERENCES Sala (Numero_Sala)

```

Oracle® 11g™

```

CREATE TABLE Cidade (
    ID_Cidade           INTEGER NOT NULL,
    UF                  CHAR(2) NOT NULL,
    Descricao           VARCHAR2(45) NULL
);
CREATE UNIQUE INDEX XPKCidade ON Cidade
(
    ID_Cidade           ASC,
    UF                  ASC
);
CREATE INDEX XIF15Cidade ON Cidade
(
    UF                  ASC
);
ALTER TABLE Cidade
    ADD ( PRIMARY KEY (ID_Cidade, UF) );
CREATE TABLE Cirurgia (
    CRM                SMALLINT NOT NULL,
    Numero_Internacao  INTEGER NOT NULL,
    Numero_Sala         INTEGER NOT NULL,
    Data_Cirurgia       DATE NOT NULL,
    Hora_Cirurgia       DATE NOT NULL,
    Identificacao_Especialidade  INTEGER NOT NULL,
    Qtde_sangue_solicitada DECIMAL(2,1) NULL,
    Qtde_sangue_utilizada DECIMAL(2,1) NULL
);
CREATE UNIQUE INDEX XPKCirurgia ON Cirurgia
(
    CRM                ASC,
    Numero_Internacao  ASC,
    Numero_Sala         ASC,
    Data_Cirurgia       ASC,
    Hora_Cirurgia       ASC
);
CREATE INDEX XIF1Cirurgia ON Cirurgia
(
    CRM                ASC
);
CREATE INDEX XIF11Cirurgia ON Cirurgia
(
    Identificacao_Especialidade  ASC
);
CREATE INDEX XIF2Cirurgia ON Cirurgia
(
    Numero_Internacao      ASC
);
CREATE INDEX XIF8Cirurgia ON Cirurgia
(

```

```

        Numero_Sala          ASC
);
ALTER TABLE Cirurgia
    ADD ( PRIMARY KEY (CRM, Numero_Internacao, Numero_Sala,
        Data_Cirurgia, Hora_Cirurgia) ) ;
CREATE TABLE Consumo (
    Numero_Consumo      CHAR(18) NOT NULL,
   Codigo_Produto       INTEGER NOT NULL,
    Hora_Cirurgia        DATE NOT NULL,
    Data_Cirurgia        DATE NOT NULL,
    CRM                 SMALLINT NOT NULL,
    Numero_Internacao   INTEGER NOT NULL,
    Numero_Sala          INTEGER NOT NULL,
    Quantidade           DECIMAL(2) NULL
);
CREATE UNIQUE INDEX XPKConsumo ON Consumo
(
    Numero_Consumo          ASC,
    Codigo_Produto          ASC
);
CREATE INDEX XIF17Consumo ON Consumo
(
    Codigo_Produto          ASC
);
CREATE INDEX XIF3Consumo ON Consumo
(
    CRM                      ASC,
    Numero_Internacao        ASC,
    Numero_Sala               ASC,
    Hora_Cirurgia             ASC,
    Data_Cirurgia             ASC
);
ALTER TABLE Consumo
    ADD ( PRIMARY KEY (Numero_Consumo, Codigo_Produto) ) ;
CREATE TABLE Especialidade (
    Identificacao_Especialidade INTEGER NOT NULL,
    Descricao_Especialidade VARCHAR2(35) NULL
);
CREATE UNIQUE INDEX XPKEspecialidade ON Especialidade
(
    Identificacao_Especialidade    ASC
);
ALTER TABLE Especialidade
    ADD ( PRIMARY KEY (Identificacao_Especialidade) ) ;
CREATE TABLE Estado (
    UF                  CHAR(2) NOT NULL,
    Nome_Estado         VARCHAR2(40) NULL
);
CREATE UNIQUE INDEX XPKEstado ON Estado
(
    UF                  ASC
);
ALTER TABLE Estado
    ADD ( PRIMARY KEY (UF) ) ;
CREATE TABLE Material_e_Medicamento (
    Codigo_Produto      INTEGER NOT NULL,
    Principio_Ativo     VARCHAR2(50) NULL,
    Descricao            VARCHAR2(40) NULL,
    Unidade              VARCHAR2(2) NULL,
    Saldo_Estoque        DECIMAL(5,2) NULL
);
CREATE UNIQUE INDEX XPKMaterial_e_Medicamento ON Material_e_Medicamento

```

```

(
   Codigo_Produto          ASC
);
ALTER TABLE Material_e_Medicamento
    ADD ( PRIMARY KEY (Codigo_Produto) ) ;
CREATE TABLE Medico (
    CRM                  SMALLINT NOT NULL,
    ID_Cidade            INTEGER NOT NULL,
    UF                  CHAR(2) NOT NULL,
    Id_Especialidade    INTEGER NOT NULL,
    Nome                VARCHAR2(40) NULL,
    Endereco            VARCHAR2(35) NULL,
    Bairro              VARCHAR2(25) NULL,
    Salario             NUMBER(24,4) NULL
);
CREATE UNIQUE INDEX XPKMedico ON Medico
(
    CRM          ASC
);
CREATE INDEX XIF16Medico ON Medico
(
    ID_Cidade      ASC,
    UF            ASC
);
CREATE INDEX XIF7Medico ON Medico
(
    Id_Especialidade   ASC
);
ALTER TABLE Medico
    ADD ( PRIMARY KEY (CRM) ) ;
CREATE TABLE Paciente (
    Numero_Internacao  INTEGER NOT NULL,
    CEP                VARCHAR2(11) NULL,
    Nome               VARCHAR2(40) NULL,
    Endereco           VARCHAR2(35) NULL,
    Sexo               VARCHAR2(1) NULL,
    Nascimento         DATE NULL,
    Profissao          VARCHAR2(35) NULL
);
CREATE UNIQUE INDEX XPKPaciente ON Paciente
(
    Numero_Internacao   ASC
);
ALTER TABLE Paciente
    ADD ( PRIMARY KEY (Numero_Internacao) ) ;
CREATE TABLE Qualificacao_da_Sala (
    Numero_Sala        INTEGER NOT NULL,
    Identificacao_Especialidade INTEGER NOT NULL,
    Data_Inicio_qualificacao DATE NULL
);
CREATE UNIQUE INDEX XPKQualificacao_da_Sala ON Qualificacao_da_Sala
(
    Numero_Sala          ASC,
    Identificacao_Especialidade   ASC
);
CREATE INDEX XIF5Qualificacao_da_Sala ON Qualificacao_da_Sala
(
    Numero_Sala          ASC
);
CREATE INDEX XIF6Qualificacao_da_Sala ON Qualificacao_da_Sala
(
    Identificacao_Especialidade   ASC
);

```

```

);
ALTER TABLE Qualificacao_da_Sala
    ADD ( PRIMARY KEY (Numero_Sala, Identificacao_Especialidade) ) ;
CREATE TABLE Sala (
    Numero_Sala           INTEGER NOT NULL,
    Andar                 VARCHAR2(5) NULL,
    Ala_do_Centro         VARCHAR2(25) NULL
);
CREATE UNIQUE INDEX XPKSala ON Sala
(
    Numero_Sala          ASC
);
ALTER TABLE Sala
    ADD ( PRIMARY KEY (Numero_Sala) ) ;
ALTER TABLE Cidade
    ADD ( FOREIGN KEY (UF)
          REFERENCES Estado ) ;
ALTER TABLE Cirurgia
    ADD ( FOREIGN KEY (Identificacao_Especialidade)
          REFERENCES Especialidade ) ;
ALTER TABLE Cirurgia
    ADD ( FOREIGN KEY (Numero_Sala)
          REFERENCES Sala ) ;
ALTER TABLE Cirurgia
    ADD ( FOREIGN KEY (Numero_Internacao)
          REFERENCES Paciente ) ;
ALTER TABLE Cirurgia
    ADD ( FOREIGN KEY (CRM)
          REFERENCES Medico ) ;
ALTER TABLE Consumo
    ADD ( FOREIGN KEY (Codigo_Produto)
          REFERENCES Material_e_Medicamento ) ;
ALTER TABLE Consumo
    ADD ( FOREIGN KEY (CRM, Numero_Internacao, Numero_Sala,
                        Data_Cirurgia, Hora_Cirurgia)
          REFERENCES Cirurgia ) ;
ALTER TABLE Medico
    ADD ( FOREIGN KEY (ID_Cidade, UF)
          REFERENCES Cidade ) ;
ALTER TABLE Medico
    ADD ( FOREIGN KEY (Id_Especialidade)
          REFERENCES Especialidade ) ;
ALTER TABLE Qualificacao_da_Sala
    ADD ( FOREIGN KEY (Identificacao_Especialidade)
          REFERENCES Especialidade ) ;
ALTER TABLE Qualificacao_da_Sala
    ADD ( FOREIGN KEY (Numero_Sala)
          REFERENCES Sala ) ;

```

MODELAGEM DE DADOS E MÉTODOS ÁGEIS

Os métodos ágeis são hoje uma metodologia de desenvolvimento consolidada, e sua aplicação torna-se cada dia mais indiscutível.

Nesse contexto, o projeto conceitual de bancos de dados precisa ser adaptado para que se representem as abstrações de dados em um ambiente dinâmico de validação delas.

Em projetos em que se utilizam métodos ágeis, todos os desenvolvedores estão orientados à metodologia, à evolução e a uma característica muito específica desse método, que é a refatoração do código, e estes processos de refatoração são realizados com naturalidade, produtividade e qualidade.

Refatoração (Refactoring) é uma das práticas dos métodos ágeis para um projeto simples. Essa prática mantém o foco das atividades de projeto (design) nas funcionalidades de cada versão do sistema, sem prever possíveis necessidades futuras. É nesse ponto que devemos nos concentrar para entender como ficam os projetos de bancos de dados com a utilização dos métodos ágeis.

A seguir, vamos apresentar resumidamente os conceitos básicos dos métodos ágeis, para facilitar o entendimento da modelagem conceitual com a utilização desses métodos e as consequências de projeto e implantação de banco de dados.

Métodos ágeis

O *Manifesto for Agile Software Development*, ou simplesmente Manifesto Ágil, formaliza os princípios básicos que dão suporte aos métodos ágeis de desenvolvimento de software.

O termo ágil (*agile*) foi adotado pelos criadores desses métodos, que formaram a Aliança Ágil (*Agile Alliance*) e disponibilizaram suas ideias, bem como diversos recursos relacionados a elas, em um website (AGILE ALLIANCE, 2005).

O Manifesto Ágil resume-se às linhas do quadro apresentado a seguir:

De acordo com o site do Manifesto, seus participantes estão descobrindo melhores formas de desenvolver software, enquanto o fazem e enquanto ajudam os outros a fazê-lo.

Por meio desse trabalho, passaram a valorizar:

- ▶ indivíduos e interações mais que processos e ferramentas;
- ▶ software funcionando mais que documentação abrangente;
- ▶ colaboração do cliente mais que negociação contratual;
- ▶ responder a mudanças mais que seguir um plano.

Embora também haja valor nos itens à direita, o manifesto dá mais valor aos itens à esquerda.

Os princípios fundamentais que dão suporte aos métodos ágeis são os seguintes:

1. A maior prioridade é satisfazer o cliente por meio da entrega pronta e contínua de software com valor agregado.
2. Devem-se receber bem as alterações em requisitos, mesmo tarde no desenvolvimento. Processos ágeis suportam mudanças para a vantagem competitiva do cliente. Responder a mudanças mais que seguir um plano significa uma rápida adaptação às mudanças.
3. Deve haver entregas de softwares funcionando frequentemente, de algumas semanas a alguns poucos meses, com preferência para a escala de tempo mais curta.
4. Interessados na aplicação (*stakeholders*) e desenvolvedores devem trabalhar juntos diariamente ao longo do projeto. Deve haver cooperação constante entre pessoas que entendem do “negócio” e desenvolvedores.
5. Os projetos devem ser construídos ao redor de indivíduos motivados, dando a eles o ambiente e o suporte de que precisam e confiando neles para que façam o serviço.
6. O método mais eficiente e eficaz de conduzir informações para e dentro de um time de desenvolvimento é a comunicação face a face. Devem-se valorizar indivíduos e interações mais que processos e ferramentas.
7. Software funcionando é a medida principal de progresso.
8. Processos ágeis promovem desenvolvimento sustentável. Patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente. Deve-se valorizar a colaboração com clientes mais que a negociação de contratos.
9. Atenção contínua à excelência técnica e ao bom projeto aumenta a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.
11. As melhores arquiteturas e os melhores requisitos e projetos surgem de times auto-organizados.
12. Em intervalos regulares, o time deve refletir sobre como pode se tornar mais eficaz, e então deve melhorar e ajustar seu comportamento de acordo com isso.

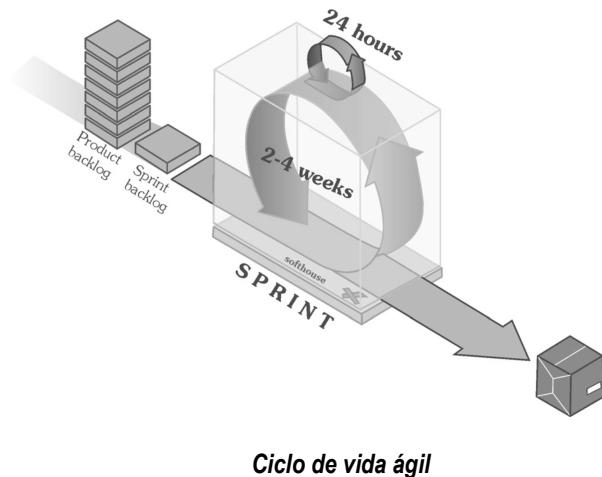
Fonte: Manifesto para o Desenvolvimento Ágil de Software. Disponível em: <http://manifestoagil.com.br/>.

Os valores e princípios do Manifesto Ágil são a fundamentação dos métodos ágeis.

Dentro da visão ágil, o projeto de sistemas é realizado por meio de *releases*, ou versões do projeto que vão sendo liberadas e implementadas evolutivamente, em síntese.

Em vez de etapas e atividades definidas no início do projeto, o ciclo de vida de um projeto ágil é composto por **iterações de software funcionando**, denominadas *sprints*.

Esse ciclo de vida é representado pela figura a seguir, conhecidíssima de vários artigos na web, mas que vale a pena apresentar:



Ciclo de vida ágil

Nessa visão e nesse formato de execução de um projeto, cria-se uma zona de conflito entre os desenvolvedores e os administradores de dados, pois estes últimos atuam com processos e ferramentas não tão flexíveis, além de trabalharem com a preocupação de manter estruturas de dados duradouras e estáveis.

Como o projeto de sistema vai sendo desenvolvido e são liberados *releases* periodicamente, as estruturas de bancos de dados também podem ser modeladas pouco a pouco, apesar de ser possível explorar suas características e detalhes mais profundamente, mesmo durante o desenvolvimento de um *sprint*, em que não sejam explorados ou apareçam todos os seus atributos durante a análise de requisitos na modelagem ágil¹.

¹ Conheça mais sobre métodos ágeis em: MACHADO, F. N. **Análise e gestão de requisitos de software**: onde nascem os sistemas. São Paulo: Érica, 2013.

A refatoração de banco de dados

Analisando o impacto sobre a implantação dos bancos de dados, observamos que refatorar qualquer estrutura de tabelas de um deles é uma tarefa complexa e que pode, caso exista uma administração de dados independente do desenvolvimento com os métodos ágeis, criar um ponto de perda de produtividade dos ganhos obtidos com a aplicação desses métodos.

Por outro lado, podemos considerar que esse formato de desenvolvimento pode criar estruturas de dados particionadas verticalmente, ou seja, uma tabela que representaria um objeto de negócio pode ser criada como duas tabelas com nomenclatura similar e com relacionamento “um para um” entre si, ou também pode levar a estruturas de dados com nível acentuado de redundâncias.

De qualquer maneira, uma vez que um projeto de banco de dados tenha sido implementado e seja necessário acrescentar colunas em tabelas já em produção, será necessário que, antes que um novo *release* de aplicação seja liberado, realize-se a recarga das tabelas com as novas colunas.

Lembre-se de que o modelo físico sempre é construído a partir de um modelo lógico, e que descreve as estruturas físicas de armazenamento de dados importantes, como restrições, gatilhos etc., o que faz com que, a cada *release* liberado, seja necessário implementar tabelas com seus relacionamentos ou colunas de *foreign key* e restrições para novos relacionamentos decorrentes de requisitos detalhados nesse *sprint* de desenvolvimento.

Considere que, entre os primeiros *sprints*, não é normal aparecerem muitas alterações desse tipo; porém, conforme o projeto vai avançando, a probabilidade aumenta, e assim as consequências, caso não haja atenção suficiente.

Quando mais avançado estiver o projeto e quanto maior o número de *sprints* realizados e versões liberadas para o ambiente de produção do software objeto do projeto, maior é o trabalho para a realização de refatoração do banco de dados. Isso irá, inclusive, criar momentos em que a produtividade obtida com os métodos ágeis será afetada, pois pode ser necessária a realização de análise de impacto nas aplicações já liberadas e em produção com relação às mudanças que serão necessárias na refatoração do banco de dados, o que, por si só, já demanda tempo de trabalho para a realização do cruzamento das estruturas do banco com a utilização delas pela aplicação já liberada e em produção.

Consequentemente, existirá a necessidade de refatoração de códigos já liberados e novos testes de aplicação e aderência das estruturas de banco, provocando a existência de *sprints* paralelos de manutenção e gestão de mudanças, antes da finalização do *sprint* gerador do fato.

O futuro da modelagem de dados

Ao trabalhar com métodos ágeis, é importante, antes de realizar qualquer refatoração no banco de dados, entender a necessidade preservar tanto a semântica comportamental das aplicações que já acessam o banco de dados implementado quanto a semântica informacional, ou seja, o significado das informações já implementadas no banco de dados do ponto de vista dos usuários dessas informações.

Assim, qualquer alteração na base, seja ela uma refatoração, uma transformação ou uma migração, precisa ser submetida constantemente a testes de integração, e posteriormente ser controlada em um ambiente de homologação, da mesma forma que as aplicações, antes que essas alterações sejam aplicadas na base de produção e seja possível liberar um novo release do projeto que está sendo realizado.

Com atenção e sem perder o foco na obtenção dos requisitos de dados, podemos realizar a modelagem de dados de forma ágil e precisa, e também controlar a evolução das instâncias do banco de dados.

Em breve, vamos explorar com mais detalhes as técnicas de controle de *scripts* de bancos de dados, em conjunto com as versões de sistemas, pois consideramos importante ter um ciclo de vida para as alterações da base de dados que envolvam a aplicação dos *scripts* SQL, obtendo uma estrutura/tabela nova. Também é importante, antes que a estrutura/tabela antiga seja considerada obsoleta, termos métodos para certificar-nos de que a alteração realmente é apropriada e adequada à versão da aplicação que está por ser liberada.

Sugerimos, então, de forma simplificada, que sejam adotadas como boas práticas:

- ▶ agrupar alterações pequenas, formando uma única alteração grande;
- ▶ identificar unicamente cada alteração;
- ▶ simplificar o processo de negociação de alterações com o time de desenvolvimento, assim como o processo de controle de alterações do banco de dados;
- ▶ não duplicar *scripts* SQL, dispondo-os em um único lugar, com controle de versionamento relacionado diretamente à versão de software.

Com este capítulo, esperamos ter dado uma contribuição aos que estudam tanto projetos de software quanto projetos de bancos de dados, disciplinas que caminham em conjunto.

Indicações de leitura

AMBLER, S. W.; SADALAGE, P. J. **Refactoring databases: evolutionary database design.** Michigan: Addison Wesley, 2006.

SADALAGE, P. J. **Recipes for continuous database integration: evolutionary database development.** Michigan: Addison Wesley, 2007.

Bibliografia

- BATINI, C.; CERI, S.; NAVATHE, S.B. **Conceptual Database Design: An Entity-Relationship Approach.** California: Benjamin/Cummings Publishing, 1992.
- BOEHM, B.W. **Software Engineering Economics.** New Jersey: Prentice-Hall, 1981.
- BOHM, C.; JACOPINI, G. **Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules.** EUA: Communications of the ACM, v. 9, n. 5, pp. 366-371, 1966.
- BOWMAN, J.S. **The Pratical SQL Handbook: using strutured query language.** USA: Addison-Wesley Developers Press, 1996.
- BRADLEY, J. **Introduction to DataBase Management in Business.** 2. ed. New York: HRW IE, 1987.
- CHEN, P. **The Entity-Relationship Model: Toward a Unified View of Data.** EUA: ACM Transactions on DataBase Systems, v. 1, n. 1, pp. 9-36, 1976.
- CODD, E. F. **Is Your DBMS Really Relational?** EUA: Computer World, out 1985.
- _____. **The Relational model for database management.** USA: Addison-Wesley Publishing, 1990.
- _____. **A relational Model of Data for Large Shared Data Banks.** EUA: Communications ACM. v. 13, n. 6, pp. 377-387, 1970.
- DEMARCO, T. **Structured Analisys and System Specification.** New Jersey: Prentice-Hall, 1978.
- DIJKSTRA, E.W. **Go To Statements Considered Harmful.** EUA: Communications of the ACM. v. 11, n. 3, pp. 147-148, maio, 1966.
- 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.
- FLAVIN, M. **Fundamental Concepts of Information Modeling.** New York: Yourdon Press, 1981.
- GANE, C.; SARSON, T. **Structured Systems Analysis: Tools and Techniques.** New Jersey: Prentice-Hall, 1979.
- KENT, W. **A Simple Guide to Five Normal Forms in Relational DataBase Theory.** EUA: Communications of the ACM. v. 26, n. 2, pp. 120-125, 1983.
- KIPPER, E.F.; et al. **Engenharia de Informações: Conceitos, Técnicas e Métodos.** Porto Alegre: SAGRA-DC LUZZATO, 1993.
- KOWAL, J.A. **Analyzing Systems.** New Jersey: Prentice-Hall, 1988.
- LEÃO, R.O.; SILVA, J.C. **SQL Server 2000: Estrutura e Implementação de Sistemas de Bancos de Dados.** São Paulo: Érica, 2002.
- LIMA, A.S. **ERwin 4.0: Modelagem de Dados.** São Paulo: Érica, 2002.
- MACHADO, FELIPE. **Analise e Gestão de Requisitos de Software: Onde Nascem os Sistemas.** São Paulo: Érica, 2013.
- MACIASZEK, L.A. **Database Design and Implementation.** Austrália: Prentice-Hall, 1990.

- _____. **Database Design and Implementation**. Austrália: Prentice-Hall, 1989.
- MARTIN, J.; FINKELSTEIN, C. **Information Engineering**. v. I / II. Inglaterra: Savant Institute, 1981.
- MATOS, A.V. **UML: Prático e Descomplicado**. São Paulo: Érica, 2002.
- MELO, L.E.V. **Gestão do Conhecimento: Conceitos e Aplicações**. São Paulo: Érica, 2003.
- MORELLI, E.T. **Oracle 8: SQL, PL/SQL e Administração**. São Paulo: Érica, 2000.
- ÖZSU, M.T.; VALDURIEZ, P. **Princípios de Bancos de Dados Distribuídos**. Rio de Janeiro: Campus, 2001.
- RODRIGUES, M. V. R. **Gestão Empresarial: Organizações que Aprendem**. Rio de Janeiro, 2002.
- SETZER, V.W. **Bancos de Dados: Conceito, Modelos, Gerenciadores, Projeto Lógico e Projeto Físico**. São Paulo: Edgard Blücher, 1986.
- SHLAER, S.; MELLOR, J.S. **Análise de Sistemas Orientada para Objetos**. São Paulo: McGraw-Hill/ Newstec, 1990.
- SILBERSCHATZ, A.; KORTH, H.F.; SUDARSHAN, S. **Sistema de Banco de Dados**. São Paulo: Makron Books, 1999.
- STAA, A.V. **Engenharia de Programas**. Rio de Janeiro: LTC, 1986.
- STEVENS, W.; MYERS, G.; CONSTANTINE, L. **Structured Design**. EUA, IBM Systems Journal, v. 13, n. 2, pp. 115-139, 1979.
- TASKER, D. **Fourth Generation Data: A Guide to Data Analysis for New and Old Systems**. Austrália: Prentice-Hall, 1989.
- TEOREY, T.J. **Database Modelling & Design**. San Francisco: Morgan Kaufmann Publisher, 1999.
- TRIMBLE Jr., J.H.; CHAPPELL, D. **A Visual Introduction to SQL**. New York: John Wiley & Sons / Metaphor Computer System., 1989.
- YOURDON, E. **Análise Estruturada Moderna**. Rio de Janeiro: Campus, 1990.
- YOURDON, E.; COAD, P. **Análise Baseada em Objetos**. Rio de Janeiro: Campus, 1992.

Sites

- AMBLER, S. W. **Agile Modeling and the Unified Process**. 2001.
Disponível em: <http://www.agilemodeling.com/essays/agileModelingRUP.htm>.
- NASCIMENTO, A. **O que é SCRUM**.
Disponível em http://www.oficinadanet.com.br/artigo/gerencia/o_que_e_scrum.
- SCHWABER, K. **The Impact of Agile Processes on Requirements Engineering**. 2002.
Disponível em: <http://www.agilealliance.org/articles/schwaberkentheimpacto/view?searchterm=requirements>.

Marcas Registradas

- ERWin 4.0® by Computer Associates International, Inc.
- Microsoft® SQL Server 2008™ by Microsoft Corporation.
- Oracle Database® 11g by Oracle Corporation.

Índice Remissivo

A

- Abstração, 14-17, 23, 27-29, 31-35, 37, 38, 69, 70, 73, 75, 76, 102, 129, 137, 150, 158, 195, 204-206, 208, 212
Administração de dados, 13, 25, 219
Administrador de dados, 24
Agregação, 15, 23, 28, 29, 31, 33, 35, 36, 104, 122, 123, 125, 128, 129, 135, 136, 138141, 143-145, 148-150, 153, 154, 156, 157, 209, 211, 240, 241, 290, 291, 297, 338, 340
reflexiva, 152
Aregar, 29
Álgebra Relacional, 25, 41, 114, 242, 243, 250, 260, 264, 283, 290, 297, 303-305, 323, 326, 329, 342
Anomalias, 24, 178, 179, 187, 190, 200
Árvore de visões, 173
Atributos, 53, 67, 72, 80, 81, 108, 116, 119, 127, 291, 304
de uma entidade, 72, 92, 191
Autorrelacionamento, 83, 85, 216, 217, 376

B

- Binário, 75, 76, 314
Boyce, 190

C

- Cardinalidade
máxima, 36
mínima, 34-36
Cascade, 59, 60
CASE, 93, 109, 111, 128, 143, 144, 213, 218
Char, 45, 314-318, 320, 322, 331, 380, 381
Chave
candidata, 54, 58, 63, 92, 191-193, 201, 235
estrangeira, 56-59, 61, 64, 117, 157, 217, 227, 234-236, 315, 319, 320, 343, 347, 376
primária, 46, 47, 49-59, 61, 73, 92, 109, 113, 181-185, 187, 188, 200, 201, 213, 218, 223, 225, 227, 233-235, 237-239, 241, 284, 306, 315, 317, 319, 321, 339, 347, 373, 376
substituta, 223
Classes de dados, 67
Classificação, 15, 23, 28, 30-32, 43, 101, 163, 167, 173, 175, 212
Codd, 23, 39, 40, 178, 190, 306
Comutatividade, 299, 300
Concatenação das colunas, 47, 48, 157
Condicional, 82

Conectividade, 71, 76-80, 82, 83, 85, 89, 96, 97, 108, 129, 130, 138, 148, 149, 158, 197, 207, 208
Conversão de modelo lógico, 219

D

Data Warehouse, 203, 223
DBA, 15, 307
DDL, 21, 308
Dependência
 de junção, 198
 funcional, 184, 185, 187, 192
 multivalorada, 193, 195, 196
Dependentes transitivos, 201
Desnormalização, 203
Diagrama ER, 70, 104, 156, 169, 175, 211, 212
Diferença, 64, 160, 242, 243, 269, 270, 272, 358, 360
Divisão, 196, 243, 280, 281, 288
DML, 308
Domínio, 44, 227

E

Efetivação lógica, 109, 113, 117, 119
E-nário, 75, 76
Encapsulamento, 174, 232
Entidade, 32, 65, 67, 68, 92, 93, 119
 fraca, 92
 generalizadora, 169, 173
Erros de modelagem, 191
Estruturas hierárquicas, 215

F

FNBC, 190-195, 201, 202
Funções em Álgebra Relacional, 290

G

Generalização, 15, 23, 28-31, 83, 102, 104, 105, 170, 212, 239

H

Herdar a chave, 196, 201

I

Índices, 20, 25, 218, 223, 242, 309, 372-374
Integer, 45
Integridade referencial, 57, 59

Interpretações, 115, 159, 160, 162-167, 170, 171, 172, 174, 175, 176, 206
Interseção, 242, 243, 274

J

Java, 14, 26
Junção, 72, 194, 198-199, 243, 275-279, 282, 287, 288, 290, 298-302, 342, 345, 348, 373

L

Levantamento de dados, 24, 213

M

Mapeamento, 33-35, 76, 224, 231-233
Metamodelo, 22, 23, 65
Minimundo, 16, 18, 20, 69, 85, 87, 88, 91, 95, 129, 143
Modelagem de dados, 13, 14, 16, 20, 23, 24, 26, 32, 39, 66, 67, 71, 72, 108, 158, 159, 167, 197, 202
Modelo
 conceitual, 16, 18, 19, 25
 de classes, 14, 66, 232, 233, 234, 241
 de classes de objetos, 14
 Entidade-Relacionamento, 22, 65, 68
 lógico de dados, 14, 19, 74, 239

N

Necessidades de informação, 17
Notação gráfica, 72, 93, 108

O

Opcionalidade, 82, 89, 117, 133
Operações relacionais, 249, 250, 251, 254, 262, 264, 280, 283, 305

P

Persistência, 13, 231, 232
Primeira forma normal, 178-180, 182, 185
Produto cartesiano, 242, 243, 251, 254, 259, 261, 267, 275, 276, 279
Projeção, 114, 199, 243-248, 250, 255, 258, 261, 280, 286, 288-290, 292, 293, 323
Projeto orientado a objetos, 13

Q

Quarta Forma Normal, 195, 200

R

- Redundâncias, 170, 177, 189, 200, 202, 210, 340
Relacionamento, 15, 18, 19, 23, 28, 29, 65, 67, 70, 71, 75, 76, 81, 83, 85, 87, 89, 92- 96, 98, 102, 104, 105, 106, 108, 109, 111, 113, 115, 117, 120, 122, 124, 126-128, 133-137, 140, 143, 144, 146, 147, 149, 150, 152, 154, 160, 162, 164-166, 169, 170, 174, 189, 196, 198, 199-203, 213, 223, 232, 239, 240, 275
reflexivo, 84, 85, 90, 153, 154, 216
Renomeação, 243, 303
Renomear, 259, 260, 293
Repositórios, 231
Restrição, 61, 227, 283
Rumbaugh, 72, 94

S

- Script, 21, 225, 232, 379
Segunda Forma Normal, 184, 184, 186, 187
Seleção, 114, 178, 232, 243, 246-248, 250, 255, 256, 260, 262, 263, 267, 275, 276, 278-280, 285, 286, 308, 322, 323, 326, 327, 330, 334, 340, 346, 348, 358, 370-372
SELECT, 217, 226, 322-347, 349-367, 369, 370, 375
Similaridades, 161
Subtipos, 102, 103
Supertipo, 102, 103

T

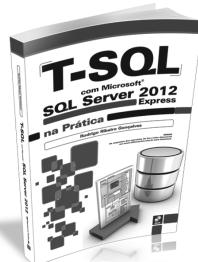
- Teoria Relacional, 39-41
Terceira Forma Normal, 187, 188
Ternários, 76, 106, 196, 201
Tipo de dado, 45, 218, 219, 318, 326, 331

U

- Um-para-muitos, 78, 79-81, 83, 85, 117, 217
União, 242, 243, 264, 266, 267, 269, 274, 293

V

- Valor nulo, 51-53, 64, 118, 301, 334, 376
Visões, 23, 153, 162, 167, 170, 218, 222, 309

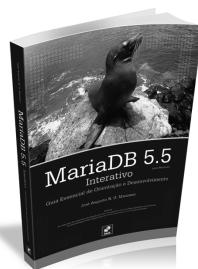


T-SQL com Microsoft SQL Server 2012 Express na Prática

Autor: Rodrigo Ribeiro Gonçalves

Código: 4537 • 120 páginas • Formato: 17 x 24 cm • ISBN: 978-85-365-0453-7 • EAN: 9788536504537

Esta obra serve como referência para o desenvolvimento de banco de dados com o Microsoft SQL Server 2012 Express, para profissionais, estudantes e interessados na área de Tecnologia da Informação. De forma didática, com exemplos e exercícios práticos, descreve técnicas de programação com a linguagem SQL, ensinando a instalar e utilizar softwares relacionados. Explica como criar e alterar tabelas, procedimentos, funções, views e triggers; utilizar o SQL dinâmico; trabalhar com datas, chaves primárias e estrangeiras, JOINs e strings. Oferece noções de administração de banco de dados, como rotinas de backup e restauração. Recomenda-se que o leitor tenha noções de programação e teoria de banco de dados.



MariaDB 5.5 - Interativo: Guia Essencial de Orientação e Desenvolvimento - para Windows

Autor: José Augusto N. G. Manzano

Código: 4155 • 256 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0415-5 • EAN: 9788536504155

Apresenta ao iniciante na administração de bancos de dados os recursos do programa gerenciador MariaDB versão 5.5, de forma simples e didática. Destaca os conceitos básicos, o surgimento da linguagem de consulta estruturada SQL; aquisição e instalação do programa; manuseio, criação e remoção de bancos de dados.

Descreve como criar tabelas e realizar consultas; operadores aritméticos; uso de funções; agrupamentos, uniões, junções e visualizações de dados; utilização de índices; chaves primária e estrangeira; stored procedures; triggers e functions.



Integração de Dados na Prática

Técnicas de ETL para Business Intelligence com Microsoft Integration Services 2012

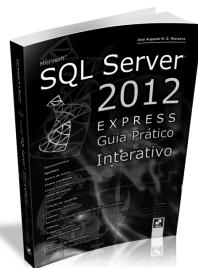
Autor: Rodrigo Ribeiro Gonçalves

Código: 4094 • 160 páginas • Formato: 17 x 24 cm • ISBN: 978-85-365-0409-4 • EAN: 9788536504094

Entender conceitos como Data Warehouse, Business Intelligence, ETL (extração, transformação e carga) e os jargões da indústria é essencial para quem quer trabalhar com integração de dados. Neste sentido, o livro foi escrito, explicando também tabelas fato e dimensão e a importância das dimensões que mudam lentamente, chamadas de Slowly Changing Dimensions.

Apresenta o Microsoft Integration Services 2012 e o Solution Explorer, introduzindo controles mais avançados como Control Flow e Data Flow, variáveis e arquivos de configuração. Para que o leitor entenda melhor a ferramenta, um pequeno projeto de Business Intelligence é desenvolvido, sendo criados três bancos de dados básicos. Conforme o projeto de ETL é montado, integrações são realizadas: Import/Export Wizard, arquivos texto, Microsoft SQL Server 2012, Data Conversion, Derived Column, Conditional Split, Merge e o MultiCast, além de mostrar como utilizar o Data Viewer para analisar a execução dos pacotes.

É indicado a analistas de sistemas, administradores de banco de dados e desenvolvedores de software com conhecimentos básicos de banco de dados e SQL para criação de tabelas e consultas, assim como inserts e updates.



Microsoft SQL Server 2012 Express - Guia Prático e Interativo

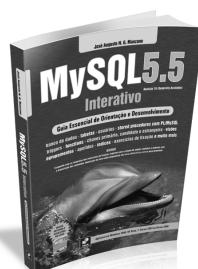
Autor: José Augusto N. G. Manzano

Código: 414A • 208 páginas • Formato: 17 x 24 cm • ISBN: 978-85-365-0414-8 • EAN: 978853654148

Estudantes e profissionais da área encontram nesta obra os principais recursos do programa gerenciador Microsoft SQL Server 2012 Express.

Descreve o surgimento da linguagem de consulta estruturada SQL; aquisição e instalação do programa; manuseio, criação e remoção de bancos de dados; tabelas e consultas; uso de funções; agrupamentos, uniões, junções e visualizações de dados; utilização de índices; chaves primária e estrangeira; stored procedures; triggers e functions.

Contempla também uma série de atividades para facilitar o aprendizado.



MySQL 5.5 - Interativo - Guia Essencial de Orientação e Desenvolvimento

Autor: José Augusto N. G. Manzano

Código: 3851 • 240 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0385-1 • EAN: 9788536503851

Esta obra apresenta ao leitor iniciante na prática de administração de bancos de dados o programa gerenciador Oracle MySQL 5.5. Abrange o surgimento da linguagem de consulta estruturada SQL; aquisição e instalação do programa; manuseio, criação e remoção de bancos de dados; tabelas e consultas; uso de funções; agrupamentos, uniões, junções e visualizações de dados; utilização de índices; chaves primária e estrangeira; stored procedures; triggers e functions.

O conteúdo segue a mesma estrutura didática do livro MySQL 5.1 - Interativo - Guia Básico de Orientação e Desenvolvimento, porém as telas, os comandos e os textos foram adaptados à versão 5.5 do programa.



PostgreSQL 8.3.0 - Interativo: Guia de Orientação e Desenvolvimento

Autor: José Augusto N. G. Manzano

Código: 1987 • 240 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0198-7 • EAN: 9788536501987

Objetiva, didática e interativa, esta obra fornece um estudo do programa gerenciador de banco de dados PostgreSQL em sua versão 8.3.0, para Windows.

Apresenta o histórico da linguagem de consulta estruturada SQL e do PostgreSQL. Descreve a aquisição e instalação do programa, manuseio, criação e remoção de bancos de dados, tabelas, consultas, uso de funções, agrupamentos, uniões, junções e visualização de dados, gerenciamento de usuários, chaves primária e estrangeira, rotinas armazenadas (funções e gatilhos) e ferramentas de administração.

Possui uma série de atividades de manuseio do programa e exercícios de fixação e é indispensável aos iniciantes na administração de banco de dados.



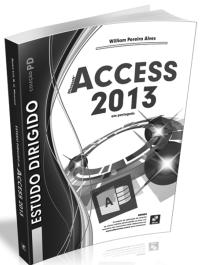
Oracle Database 10g Express Edition - Guia de Instalação, Configuração e Administração com Implementação PL/SQL Relacional e Objeto-Relacional

Autor: Robson Soares Silva

Código: 1628 • 240 páginas • Formato: 17 x 24 cm • ISBN: 978-85-365-0162-8 • EAN: 9788536501628

O livro detalha os principais recursos desse poderoso banco de dados, como a aquisição e instalação do programa, criação de tabelas, chaves primária e estrangeira, consultas SQL, utilização de índices, usuários do sistema, criação de usuários, concessão e restrição de direitos de acesso, programação PL/SQL, procedures, functions, packages, triggers, implementação objeto-relacional, criação de aplicações, acesso remoto, backup e restore, flashback, níveis de certificação e conexão Oracle com o Java.

Possui também diversos exercícios propostos, além de um simulado com questões para certificação.



Estudo Dirigido de Microsoft Access 2013

Autor: William Pereira Alves

Código: 4605 • 272 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0460-5 • EAN: 9788536504605

Com estilo agradável, este livro trata dos principais recursos do Access 2013 para o desenvolvimento de aplicações de banco de dados. O aprendizado é complementado com exercícios e exemplos, incluindo um projeto de aplicativo.

Conceitos e fundamentos de bancos de dados relacionais abrem o estudo. Em seguida, explicam-se criação e alteração de tabelas; manipulação de registros do banco de dados (inserção, alteração e exclusão); criação, alteração e execução de consulta; construção e uso de formulários de entrada de dados, relatórios e etiquetas de endereçamento; elaboração de macros e desenvolvimento de rotinas em Visual Basic for Applications (VBA), para automatizar tarefas. Ao final, ensina-se como trocar informações do Access 2013 com o Word 2013 e o Excel 2013.

A leitura é recomendada a estudantes, profissionais e demais interessados na área.



Banco de Dados - Projeto e Implementação - Edição Revisada

Autor: Felipe Nery Rodrigues Machado

Código: 0190 • 400 páginas • Formato: 17 x 24 cm • ISBN: 978-85-365-0019-5 • EAN: 9788536500195

Com uma apresentação diferenciada, utilizando metodologia e conceitos embasados na execução prática de projetos de banco de dados, o livro traz a experiência do autor, com uma linguagem simples e objetiva, que permite a compreensão das técnicas de forma gradativa e efetiva.

Destaca aspectos conceituais, a orientação à gestão de negócios e aborda a utilização de álgebra relacional, mapeamento OO → ER no projeto de banco de dados, comandos básicos da linguagem SQL ANSI, explanados com exemplos e estudos de caso.

A segunda reimpressão da segunda edição foi revisada e atualizada para Microsoft SQL 2008 e Oracle 10g.

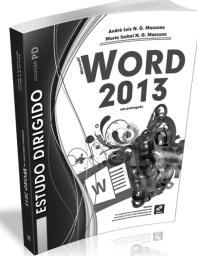


Banco de Dados - Teoria e Desenvolvimento

Autor: William Pereira Alves

Código: 2557 • 288 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0255-7 • EAN: 9788536502557

Didático e interativo, este livro aborda os principais conceitos e fundamentos de bancos de dados, como os tipos de acesso a arquivos, estruturas de dados (listas lineares e árvore), métodos de ordenação (Bubble Sort, QuickSort e Shell), arquitetura e organização, álgebra relacional, uso de índices, modelagem e normalização de dados, os cuidados necessários à implementação de um banco de dados, conceito e estrutura de Data Warehouse (OLAP, OLTP, Data Mining e Data Mart), bancos de dados distribuídos, dedutivos, hierárquicos, de rede e orientados a objetos (ODL e OQL), segurança e proteção com níveis de acesso dos usuários, processamento de transações e princípios da linguagem SQL.



Estudo Dirigido de Microsoft Word 2013

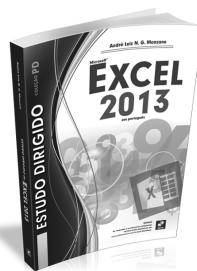
Autores: André Luiz N. G. Manzano e Maria Izabel N. G. Manzano

Código: 4568 • 160 páginas • Formato: 17 x 24 cm • ISBN: 978-85-365-0456-8 • EAN: 9788536504568

As principais ferramentas do Word 2013 para criação de documentos criativos e sofisticados compõem este livro. Com diversos exercícios, ensina como inserir e remover textos, movimentar o cursor, editar documentos, acentuar palavras, fazer a correção ortográfica, usar a área de transferência, salvar arquivos e imprimi-los.

Também explora a formatação de documentos, alinhamentos, recuos de parágrafo, marcadores, tabulação, cabeçalho, rodapé e numeração de páginas, inserção de tabelas e gráficos.

Descreve como elaborar um jornal e, ainda, abrange mala direta, uso de textos automáticos e etiquetas, mesclagem de documentos e impressão. Indica como sanar dúvidas sobre o programa, usar atalhos para executar comandos, personalizar a barra de status, revisar o texto e muito mais.



Estudo Dirigido de Microsoft Excel 2013

Autor: André Luiz N. G. Manzano

Código: 449A • 208 páginas • Formato: 17 x 24 cm • ISBN: 978-85-365-0449-0 • EAN: 9788536504490

O livro apresenta os principais recursos do Excel 2013, com abordagem simples e dinâmica. Estudantes e profissionais da área podem se beneficiar de explicações didáticas, exemplos práticos, descritos passo a passo, e exercícios, para reforçar o aprendizado. Introduz a nova interface do aplicativo, incluindo grupos, comandos e guias. Ensina a criar e formatar planilhas, inserir fórmulas, trabalhar com funções matemáticas, operar com bases de dados, criar gráficos, imprimir relatórios e usar comandos de congelamento. Trata do bloqueio de edição e da criação de senhas para planilhas. Apresenta planilhas de consolidação e traz dicas sobre personalização e teclas de atalho.



Estudo Dirigido de Microsoft Excel 2013 - Avançado

Autores: José Augusto N. G. Manzano e André Luiz N. G. Manzano

Código: 4506 • 288 páginas • Formato: 17 x 24 cm • ISBN: 978-85-365-0450-6 • EAN: 9788536504506

O livro destaca os recursos avançados do Excel 2013, sendo direcionado para estudantes e profissionais da área. Em dez capítulos, apresenta, demonstra e revisa funções de cálculos; abrange a criação e a análise de bases de dados; comprehende o uso de tabelas e gráficos dinâmicos. Oferece exemplos de folhas de pagamento, cadastros de alunos, planejamento financeiro e tabelas de vendas. Descreve a utilização de macros e recursos relacionados a atividades de programação, incluindo tipos de macro e sua execução, cadastros para armazenamento de dados, macros interativas e técnicas para a personalização de campos. Oferece, também, exemplos e exercícios.

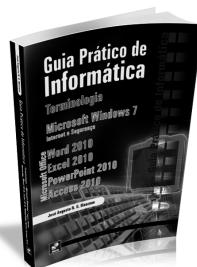


Integração de Dados com PowerPivot e Microsoft Excel 2010

Autor: Newton Roberto Nunes da Silva

Código: 4254 • 192 páginas • Formato: 17 x 24 cm • ISBN: 978-85-365-0425-4 • EAN: 9788536504254

Fornece explicações passo a passo sobre o PowerPivot para Excel 2010, com exercícios e exemplos para auxiliar estudantes e profissionais da área. Explica a instalação do programa e procedimentos para importar dados, formatar colunas, vincular dados do Excel e atualizá-los no PowerPivot. Aborda relacionamentos, Expressões de Análise de Dados (Data Analysis Expressions), segmentações, relatórios de tabela e gráfico dinâmico, além do uso de funções DAX para criar medidas específicas para relatórios dinâmicos. Concluindo, abrange a formatação final do relatório no PowerPivot, deixando-o com um aspecto mais profissional e com características de painel de controle, que consolida dados e exibe-os de forma inteligível.

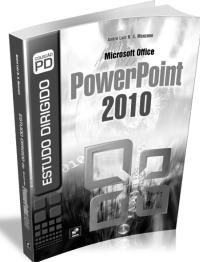


Guia Prático de Informática - Terminologia, Microsoft Windows 7 (Internet e Segurança), Microsoft Office Word 2010, Microsoft Office Excel 2010, Microsoft Office PowerPoint 2010 e Microsoft Office Access 2010

Autor: José Augusto N. G. Manzano

Código: 3349 • 376 páginas • Formato: 20,5 x 27,5 cm • ISBN: 978-85-365-0334-9 • EAN: 9788536503349

Esta obra apresenta os conceitos essenciais de informática para o dia a dia, principalmente para leitores nos primeiros estágios de aprendizagem. Mostra a terminologia da área, como computadores, sistemas operacionais, programas aplicativos e periféricos, bem como os recursos do Microsoft Windows 7, Internet e princípios de segurança. Abrange as principais ferramentas do Microsoft Office 2010: Word (processador de textos), Excel (planilha eletrônica), PowerPoint (gerenciador de apresentações) e Access (gerenciador de banco de dados).

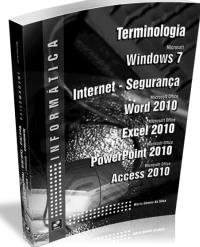


Estudo Dirigido de Microsoft Office PowerPoint 2010

Autor: André Luiz N. G. Manzano

Código: 2960 • 192 páginas • Formato: 17 x 24 cm • ISBN: 978-85-365-0296-0 • EAN: 9788536502960

A versão 2010 do PowerPoint proporciona mais criatividade e produtividade aos trabalhos desenvolvidos com essa ferramenta. O livro apresenta de forma didática e objetiva as técnicas de oratória, conceitos de apresentação, etapas para criação de slides, formatação, alinhamentos, gráficos, aplicação de design e cores, padrões, indicação dos meios para obter ajuda, atalhos. O conteúdo programático é útil a alunos e professores de instituições de ensino e também a profissionais da área.



Informática - Terminologia - Microsoft Windows 7 - Internet - Segurança - Microsoft Office Word 2010 - Microsoft Office Excel 2010 - Microsoft Office PowerPoint 2010 - Microsoft Office Access 2010

Autor: Mário Gomes da Silva

Código: 3103 • 360 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0310-3 • EAN: 9788536503103

Embasamento fundamental sobre o uso do computador com Windows 7 e o conjunto de aplicativos Office 2010 é encontrado nesta obra.

Apresenta a história do computador, unidades de armazenamento, periféricos, funcionalidades e tarefas básicas de Windows 7, conexão com Internet, navegação, e-mails e ferramentas de segurança. Destaca os principais recursos do Word 2010 para criação e formatação de textos, ortografia, impressão e revisão, rodapés e tabelas. Explora a criação de planilhas com Excel 2010, navegação, edição e manipulação de arquivos, operações básicas, cópias e formatação de dados, fórmulas, funções e gráficos. Com o PowerPoint 2010 ensina como criar apresentações, estruturar tópicos, usar formas, animações, transição de slides e impressão. Mostra como criar banco de dados com Access 2010, terminologias, edição de tabelas, digitação de dados, consultas, formulários e relatórios. Traz uma série de exercícios de fixação que objetivam aprimorar o conhecimento transmitido na obra.



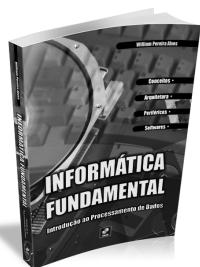
Informática - Conceitos e Aplicações

Autores: Marcelo Marçula e Pio Armando Benini Filho

Código: 0530 • 408 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0053-9 • EAN: 9788536500539

Este livro é indicado como material de apoio aos cursos de Informática e disciplinas afins dos demais cursos. Pode ser utilizado por professores (como uma diretriz básica para a disciplina), alunos (fonte de pesquisa para os principais conceitos) e profissionais de todas as áreas, que necessitem adquirir conhecimentos sobre informática.

Aborda conceitos básicos de informática, características dos componentes que formam o hardware, definição e classificação dos softwares, redes, arquiteturas, infraestrutura e serviços de Internet, segurança de dados, autenticação, criptografia, antivírus e firewall.



Informática Fundamental - Introdução ao Processamento de Dados

Autor: William Pereira Alves

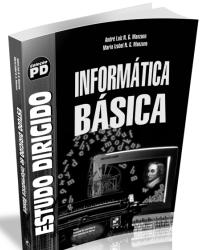
Código: 2724 • 224 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0272-4 • EAN: 9788536502724

Muitas pessoas utilizam computadores no dia a dia sem ter a menor ideia de como eles e seus diversos componentes e periféricos trabalham. Pensando neste aspecto, o livro apresenta conceitos e fundamentos de um sistema computacional, explicando como funcionam monitores, impressoras, escâneres, leitores de CD/DVD etc.

Aborda os circuitos lógicos existentes em todos os processadores, como portas AND, OR e XOR, e estuda as bases numéricas e os tipos de memória mais utilizados em computação.

Divide-se em três partes, sendo a primeira referente à arquitetura dos computadores, a segunda sobre os periféricos e a terceira relacionada ao sistema operacional e softwares mais comuns.

Os capítulos possuem diversas questões para fixação do aprendizado.



Estudo Dirigido de Informática Básica

Autores: André Luiz N. G. Manzano e Maria Izabel N. G. Manzano

Código: 1284 • 256 páginas • Formato: 17 x 24 cm • ISBN: 978-85-365-0128-4 • EAN: 9788536501284

A sétima edição do livro foi revisada e ampliada, pois há grande preocupação de trazer informações mais atualizadas frente às novas tecnologias, além de muitas novidades.

A obra manteve sua estrutura original no que tange à história e sua cronologia, preservando a linguagem simples e acessível aos novos usuários da informática. Apresenta informações riquíssimas sobre novos recursos computacionais, como, por exemplo, as tecnologias Bluetooth e Wireless, possibilidades novas dentro muitos recursos oferecidos, além de contemplar um assunto muito importante, a segurança de dados, seja em uma simples página web ou em um inocente bate-papo em salas de chat ou, ainda, em mensagens instantâneas.