

P rojeto

de Banco de Dados

Parte 1: Modelo Lógico

por Vinícius Lourenço de Sousa e Hamilton Oliveira

Nós, seres humanos, possuímos um limite sobre a quantidade de informações que podemos tratar de uma única vez. Quando estamos diante de um problema, procuramos identificar o que é relevante e o que pode ser descartado, de modo que podemos concentrar esforços na busca do meio mais adequado para solucioná-lo. A isso chamamos de *abstração*.

A técnica de abstração é utilizada por diversas áreas da computação, pois permite reduzir a complexidade de problemas, de modo que as soluções possam ser mais facilmente identificadas, descritas e gerenciadas.

A essência da ciência da computação é a identificação dos problemas contidos nos diversos segmentos da atividade humana, com o objetivo de apresentar uma solução computacional. Dividir o processo de solução desse problema em fases é uma tentativa de redução de sua complexidade.

Existem atualmente diversas propostas disponíveis que determinam como os softwares devem ser criados e mantidos. Embora existam diferenças, podemos encontrar três fases clássicas que, de uma maneira ou de outra, sempre aparecem nas abordagens. As fases são: análise, projeto e implementação.

Na fase da análise, o objetivo é modelar os conceitos envolvidos no domínio do problema e identificar de que forma esses conceitos se relacionam. Utilizando a técnica da abstração, o produto gerado nessa fase não deverá fazer qualquer referência, por exemplo, à linguagem de programação que utilizaremos para implementar o software, pois neste momento essa informação não é importante.

Quando iniciamos a *fase de projeto*, aspectos como ambiente de desenvolvimento, arquitetura do software e tecnologia que utilizaremos passam a fazer parte do processo de tomada de decisão. As informações identificadas nessa fase estão contidas no domínio da solução; ao contrário, as informações identificadas na *fase da análise* fazem parte do domínio do problema.

A *fase de implementação* consiste na transformação do projeto de baixo nível – produto gerado na fase de projeto – para uma linguagem de programação previamente selecionada.

Uma abordagem, nas fases iniciais do processo do software, voltada para questões que dizem respeito à tecnologia dificulta a visualização da solução mais adequada para o problema do cliente. Deve-se ter em mente que o cliente está interessado na solução para o seu negócio, independente de qual ambiente tecnológico a solução será implementada.

Sabemos que, em determinadas situações, a tecnologia já está definida mesmo antes de iniciarmos o projeto do software. Mesmo nesses casos, é importante desenvolver uma solução independente de tecnologia, de modo que possa ser reutilizada em projetos futuros. Quando iniciamos um processo de desenvolvimento a única certeza que temos é que o software irá mudar. Além disso, a velocidade de ascensão e queda das tecnologias no mundo da informática não encontra precedentes na história. Portanto, atrelar as regras de negócio a uma solução tecnológica pode trazer transtornos no futuro.

O projeto de banco de dados é uma atividade que tem como objetivo identificar, modelar e implementar um *modelo de dados* consistente com as necessidades dos usuários, que foram expressas na especificação de requisitos. Neste momento, as preocupações da equipe estarão voltadas para questões que dizem respeito à persistência dos dados.

N

As fases de um projeto de banco de dados recebem nomes diferentes de acordo com a literatura. O *Modelo Lógico* pode ser encontrado com a denominação de *Modelo Conceitual*; e o *Modelo Físico*, pode ser chamado de *Projeto Lógico*. A denominação da fase de implementação é utilizada de modo uniforme.

Realizar um projeto de banco de dados é importante porque: i) permite disponibilizar as informações de forma estruturada e eficiente; ii) evita a duplicação de informações e aumenta a confiabilidade dos sistemas; iii) define um planejamento que deverá ser seguido pelos membros da equipe; iv) possibilita a definição dos prazos, bem como os mecanismos de acompanhamento; v) possibilita a identificação e alocação dos recursos necessários e disponíveis; vi) possibilita a reutilização dos artefatos produzidos em outros projetos, entre outros.

Outra vantagem de um projeto de dados bem delineado é a documentação gerada, que servirá de meio de comunicação entre os membros da equipe. A ausência de documentação em geral é identificada como um dos maiores obstáculos para a manutenção dos produtos de software.

Ao iniciar o projeto de um banco de dados, normalmente somos forçados a pensar em como o banco será implementado. Quantas tabelas existirão, quantos campos, o tamanho e tipo de cada um, se existirão *Stored Procedures*, *Views* e qual o SGBD (Sistema Gerenciador de Banco de Dados) será utilizado.

Claro que são questões importantes para o sucesso do projeto, pois terão impacto na funcionalidade e até mesmo na performance do banco de dados; entretanto, nas fases iniciais do projeto essas questões são irrelevantes. Os administradores devem focar nas regras de negócio do cliente quando se inicia um projeto de banco de dados, deixando a parte física para as fases mais adiantadas do processo.

Da mesma forma e pelos mesmos motivos que o projeto do software, o projeto do banco de dados é dividido em três grandes fases. A primeira é o Modelo Lógico, a segunda é o Mode-

lo Físico e a terceira é a implementação. Neste artigo falaremos sobre o Modelo Lógico.

Modelo Lógico

Esta é a fase que modela a análise dos requisitos, onde o cliente indicou qual é o problema e o que ele espera obter com a solução. De posse das informações passadas pelo cliente, podemos iniciar a modelagem lógica do nosso banco de dados. Abaixo segue a análise dos requisitos realizada com o cliente.

Análise dos Requisitos

No exemplo nosso cliente será uma livraria que possui as seguintes informações e regras de negócio:

- A livraria Book. Net é destinada a clientes da área de informática. Ela possui cerca de 4.500 livros sobre desenvolvimento, internet, banco de dados, redes e sistemas operacionais, entre outros;
- A livraria tanto vende seus livros como também os aluga para clientes cadastrados em suas fichas, para fins de estudos;
- Os livros podem ser nacionais ou importados;
- Os clientes só podem alugar no máximo três livros de uma vez, cuja duração não poderá ultrapassar o período de duas semanas. Depois desse tempo, o cliente pode renovar o aluguel. Em toda renovação, deverá ser paga uma taxa para cada livro. Caso o cliente passe do tempo e não renove o aluguel, pagará uma multa na devolução e/ou renovação;
- Os clientes que desejam alugar um livro que já esteja alugado poderão fazer uma reserva do mesmo;
- Os livros que são pouco alugados ou não são alugados, a livraria realiza promoções para vendê-los;
- Os clientes que mais alugaram e mais compraram livros são notificados

antes dos demais sobre as promoções, tanto para compra como para aluguel;

- Os clientes podem ser pessoas físicas ou jurídicas. Os livros também são vendidos para universidades, centros educacionais e quaisquer cursos que desejam melhorar sua biblioteca;

- O cliente pessoa física, ao se cadastrar, deve informar os seguintes dados obrigatórios: nome completo, identidade e/ou CPF, data de nascimento, endereço completo e pelo menos um telefone de contato. As informações opcionais são e-mail, *home-page* pessoal e outros tipos de telefones;

- O cliente pessoa jurídica, ao se cadastrar, deve informar os seguintes dados obrigatórios: CNPJ, razão social, endereço completo e pelo menos um telefone de contato. As informações opcionais são e-mail, *home-page* e outros telefones;

- Ao cadastrar um livro, as seguintes informações são preenchidas: ISBN, nome do livro, nome da editora, autor, ano de publicação, assunto, se é para compra ou aluguel, se é nacional ou importado e quantidade que a livraria tem. Estas são informações obrigatórias. As informações opcionais são preço de venda, preço de aluguel e preço de renovação de aluguel.

Com base nos requisitos, o cliente deseja que o sistema forneça as seguintes funcionalidades:

- Controlar o cadastro de todos os clientes que compram e alugam os livros;
- Controlar o cadastro de todos os livros;
- Controlar os livros mais vendidos e alugados;
- Controlar os livros menos vendidos e alugados;
- Controlar as reservas dos livros;
- Controlar os livros que estão alugados e o tempo de aluguel dos mesmos;
- Obter relatório da quantidade de livros de um determinado assunto;
- Obter relatório da quantidade de livros de uma determinada editora.

P rojeto de Banco de Dados: Modelo Lógico

De posse das informações acima, o próximo passo é a identificação das entidades, seus atributos e relacionamentos. Para a realização desta etapa, utilizaremos o Modelo de Entidade-Relacionamento (MER).

O MER é o modelo mais utilizado quando se realiza o projeto de banco de dados, pois possui uma semântica que possibilita o mapeamento dos objetos identificados nas fases anteriores. O MER tem por base a idéia de que o mundo real (domínio do problema) é formado por um conjunto de objetos, que podem ser denominados *entidades*, e pelo conjunto de relacionamentos formados entre esses objetos. O paradigma de orientação a objetos constitui-se numa evolução das técnicas e conceitos que dão sustentação ao Modelo de Entidade-Relacionamento.

Entidade

É uma “coisa” ou “objeto” que pode ser identificado de forma inequívoca em relação a todos os outros objetos contidos no domínio do problema. Uma entidade pode ser um fichário, uma pasta ou qualquer elemento so-

bre o qual desejamos guardar alguma informação, podendo ainda ser classificadas em entidades fortes ou fracas.

Atributo

Atributos são propriedades que caracterizam uma entidade. Os atributos são preenchidos por valores que diferenciam duas instâncias de entidade de um mesmo conjunto. Para cada atributo existe ainda um conjunto de valores possíveis.

Relacionamento

Um relacionamento constitui-se numa associação entre uma ou mais entidades. Quando, num relacionamento, estão envolvidas duas entidades, chamamos de relacionamento binário; quando estão envolvidas três entidades, dizemos que temos um relacionamento ternário, e assim sucessivamente. Os relacionamentos binários são mais facilmente encontrados, sendo suficientes para modelar a maioria das associações entre as entidades.

Identificando as Entidades

O processo de identificação das en-

tidades ‘corretas’ para um modelo constitui-se numa atividade crucial para o sucesso do projeto de banco de dados. A utilização de entidades ‘erradas’ conduzirá a uma solução inadequada, que terá como consequência a necessidade de ajustes após a implementação do banco de dados. Tais ajustes serão necessários porque o domínio do problema não foi mapeado de forma ‘correta’ para o domínio da solução, tornando a aplicação resultante inconsistente com as reais necessidades do cliente.

A complexidade dessa atividade resulta da subjetividade envolvida e da impossibilidade de validar as entidades selecionadas para um dado problema. Um erro comum dos projetistas é pensar que o primeiro conjunto de entidades identificadas é suficiente para passar à próxima fase do projeto. Resalta-se ainda que, para um dado problema, duas equipes podem identificar conjuntos diferentes de entidades, e, não raro, ocorrerem situações em que ambas estejam ‘corretas’.

Seguindo o exemplo, realizaremos a identificação das entidades que participarão do nosso modelo.

Manter Cadastro dos Clientes

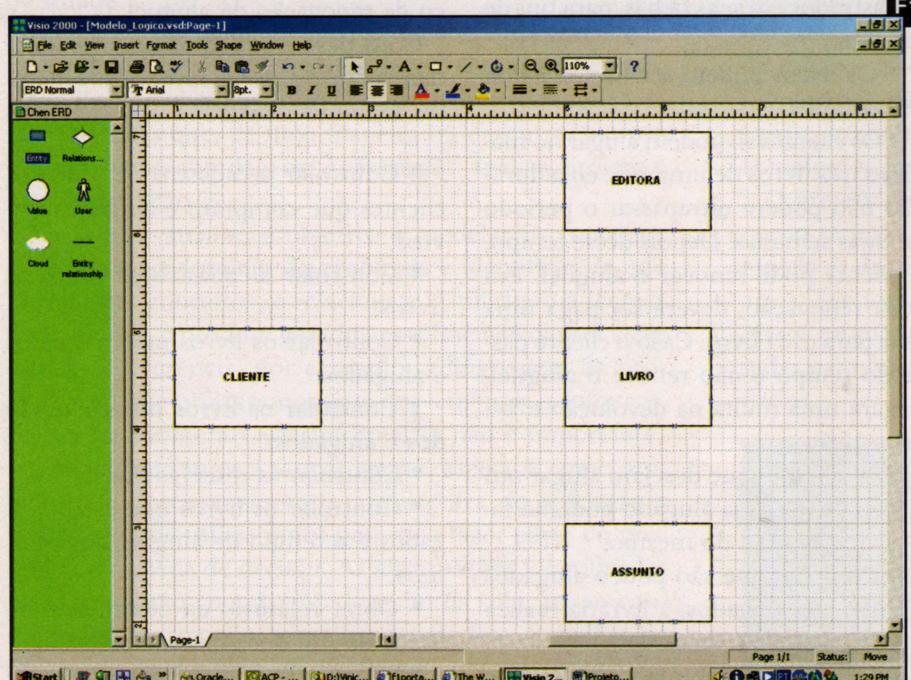
Neste item o modelo terá uma entidade *Cliente* para controlar o cadastro dos clientes. Todas as informações dos clientes estarão nesta entidade.

Manter Cadastro dos Livros

Neste item o modelo terá uma entidade *Livro* para controlar o cadastro dos livros. Todas as informações dos livros estarão nesta entidade.

Obter Relatório da Quantidade de Livros por Assunto

Neste item nossa entidade será *Assunto*, pois o cliente deseja obter um relatório com a quantidade de livros por um determinado assunto. Repare que vários livros podem falar sobre o mesmo assunto e a informação do assunto não necessariamente depende do livro. Podemos ter um assunto so-



Modelo de Entidade-Relacionamento parcial para Book.net

bre objetos distribuídos e não haver nenhum livro desse assunto na livraria, pois a mesma ainda não o adquiriu. É muito importante que o nosso modelo seja flexível para que a livraria possa ter um cadastro de assunto sem ter o livro para esse assunto.

Obter Relatório da Quantidade de Livros por Editora

Este item é idêntico ao anterior. A nossa entidade será *Editora*. O modelo pode ter um cadastro de editoras sem que existam livros dessa editora ou a livraria poderá, no futuro, deixar de adquirir livros dessa editora e querer manter o histórico.

Observando as entidades criadas na figura 1 podemos ver algumas regras para montar o nosso modelo lógico. Todas as entidades são representadas por um retângulo com o nome da entidade no singular e as letras maiúsculas e sem abreviação.

Identificando os Relacionamentos

Após a identificação das entidades, o próximo passo consiste em buscar a melhor organização estrutural para o modelo, que se dará através de relacionamentos entre as entidades. Para isso, analisaremos os seguintes requisitos:

- Controlar os livros mais vendidos e alugados;
- Controlar os livros menos vendidos e alugados;
- Controlar as reservas dos livros;
- Controlar os livros que estão alugados e o tempo de aluguel.

Nos itens acima, vemos que as regras de negócio são baseadas numa relação entre as informações. Para sabermos quais livros são mais vendidos ou menos alugados, o tempo de aluguel e os livros reservados, dependemos da relação entre as informações da entidade *Cliente* e da entidade *Livro*. Ou seja, quando a informação a ser obtida é dependente da relação entre as entidades damos o nome de relacionamento. Nesse caso podemos dizer que a entidade *Cliente* se relaciona com a en-

tidade *Livro*.

A entidade *Cliente* tem três tipos de relacionamento com a entidade *Livro*. Um relacionamento de aluguel, um de compra e um de reserva. Essas informações possuem semânticas diferentes e irão demandar operações distintas a serem realizadas sobre as entidades *Cliente* e *Livro* e, portanto, devem estar devidamente representadas no modelo, conforme pode ser observado na figura 2.

Podemos ver algumas regras para montar os relacionamentos no modelo lógico. Os relacionamentos são representados por um losango contendo o seu nome em maiúsculo, no singular e sem abreviação. Repare que o nome do relacionamento representa uma ação que o cliente está realizando em relação ao livro. A leitura é realizada da seguinte maneira:

Cliente **RESERVA** Livro, Cliente **COMPRA** Livro, Cliente **ALUGA** Livro.

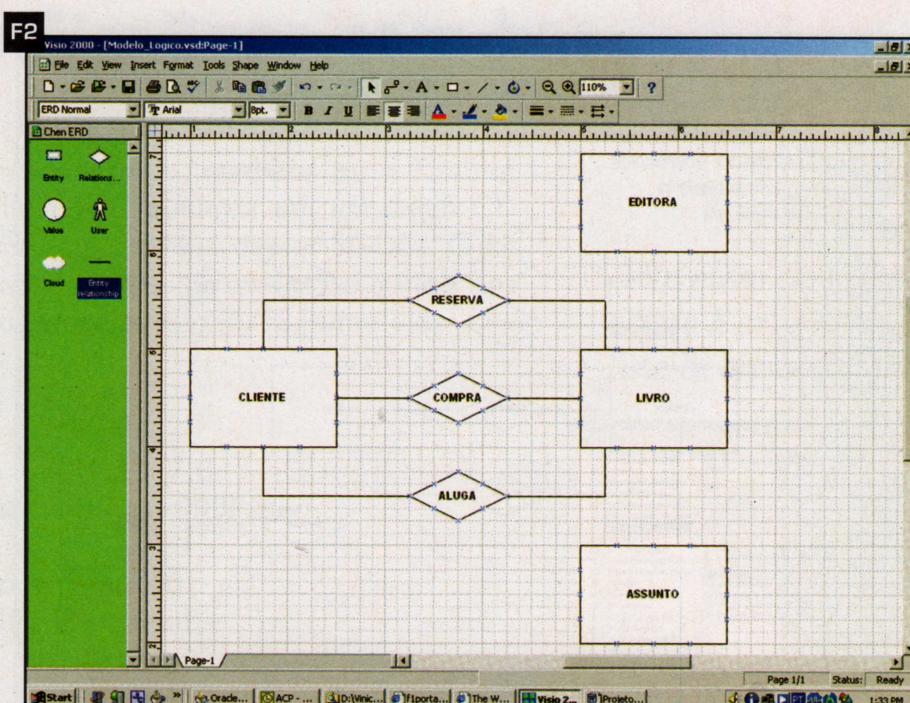
Existem outras formas de identificarmos os relacionamentos. Conforme dito anteriormente, relacionamentos são as informações que queremos ob-

ter, mas que dependem da relação entre as entidades. No exemplo acima vimos que os relacionamentos existem através de uma ação. Mas existem relacionamentos que não dependem de ação. Vendo como o nosso modelo está no momento, percebemos que duas entidades (*Assunto* e *Editora*) estão isoladas das demais.

No modelo lógico esta situação não pode ocorrer, pois *sempre* existirá relacionamento entre as entidades. Neste caso, podemos ver que as duas entidades têm um relacionamento natural com a entidade *Livro*. Partindo da regra de negócio do nosso cliente, vemos que um livro possui editora e assunto. Veja na figura 3 como ficou o modelo.

Observando a figura 3 vemos que os dois novos relacionamentos também têm nomes, mas, diferente dos outros relacionamentos, os nomes não representam uma ação. No modelo lógico mesmo quando o relacionamento não representa uma ação é importante nomeá-lo para melhorar a legibilidade.

Existem dois grandes grupos de relacionamentos:



Modelo de Entidade-Relacionamento parcial para Book.net

Relacionamentos Condicionais

Relacionamentos que possuem uma condição, uma qualificação para ocorrerem. Os relacionamentos condicionais são efetivamente aqueles em que nem todas as informações de uma entidade A estão ligadas com as informações da entidade B. Dizemos que este tipo de relacionamento possui opçãoalidade. No modelo podemos constatar que os relacionamentos entre a entidade *Cliente* e a entidade *Livro* são condicionais, pois o cliente pode ou não alugar, reservar ou comprar um livro.

Relacionamentos Incondicionais

Relacionamentos que não possuem condições, caracterizam-se por serem obrigatórios. Nos relacionamentos incondicionais, todas as informações de uma entidade estão obrigatoriamente relacionadas com uma informação, no mínimo, de outra entidade.

No modelo podemos dizer que o relacionamento entre a entidade *Livro* e a entidade *Assunto* é obrigatório, pois não pode existir um livro sem assunto. Poderíamos até dizer que o relacionamento entre a entidade *Livro* e a

entidade *Editora* também é obrigatório. Mas sabemos que pode existir um livro sem que tenha uma editora para lançá-lo. Neste caso, tudo dependerá da regra de negócio do cliente. No próximo artigo veremos como modelar esses dois grupos de relacionamentos.

Auto-Relacionamentos

Auto-Relacionamento é quando uma entidade possui um relacionamento com ela mesma. Observe o exemplo abaixo:

Todo cliente titular poderá possuir no máximo 3 clientes dependentes para aluguel de livros e todos os dependentes devem ser obrigatoriamente da família do titular.

Percebemos que existem dois tipos de clientes: titular e dependente. Esta modificação na regra de negócio implica em um auto-relacionamento para a entidade *Cliente*, conforme demonstra a figura 4.

Identificando os Atributos das Entidades

Agora que nosso modelo já possui as entidades e os relacionamentos, podemos identificar os atributos das enti-

dades. Para isso, devemos voltar à análise de requisitos, nos itens que falam sobre as informações dos clientes, livros, editora e assunto. Vamos ver os exemplos abaixo:

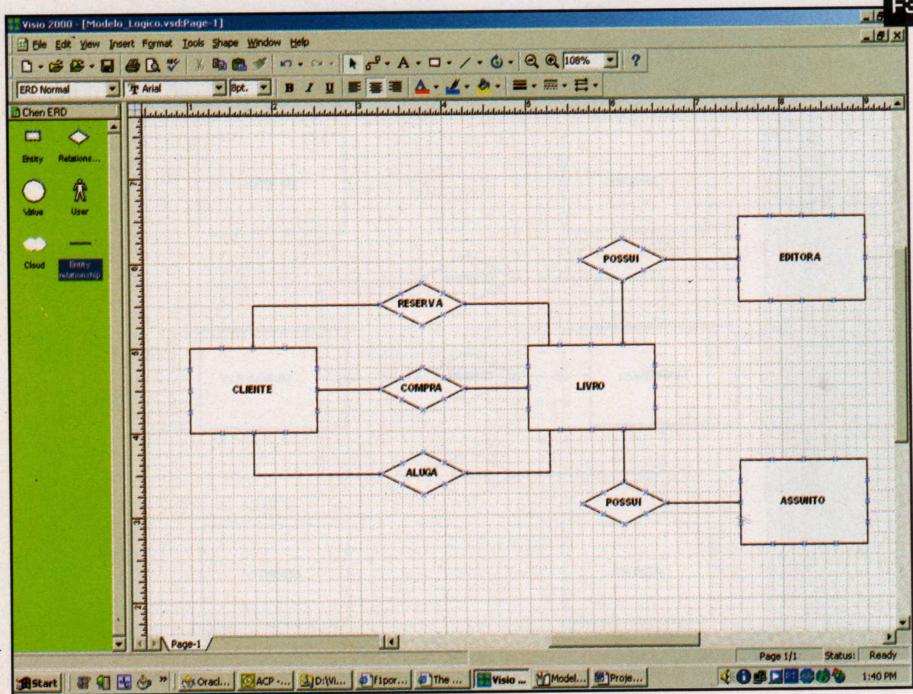
- O cliente pessoa física, ao se cadastrar, deve informar os seguintes dados obrigatórios: nome, identidade e / ou CPF, data de nascimento, endereço e pelo menos um telefone de contato. As informações opcionais são e-mail, home-page pessoal e outros tipos de telefones.

- O cliente pessoa jurídica ao se cadastrar deve informar os seguintes dados obrigatórios: CNPJ, razão social, endereço e pelo menos um telefone de contato. As informações de e-mail, home-page e outros telefones não são obrigatórias.

Repare que em nenhum momento nos preocupamos com informações como código do cliente e matrícula. Isso significa que na regra de negócio a informação código não é importante. Como estamos no modelo lógico, devemos seguir fielmente as informações passadas pelo nosso cliente, pois a existência do atributo código seria uma forma física de manter o controle dos dados.

Todos os atributos, com exceção de ENDEREÇO e TELEFONE, são atômicos, ou seja, são atributos que não podem ser divididos. Já os atributos ENDEREÇO e TELEFONE são chamados de atributos compostos, pois podemos dividí-los em atributos atômicos que façam sentido na regra de negócio do cliente. O atributo ENDEREÇO poderá ser dividido em RUA, BAIRRO, CIDADE e assim por diante. O mesmo vale para TELEFONE, que poderia ter DDD, NÚMERO e RAMAL. No entanto, se realizássemos a divisão nos atributos, prejudicaríamos a legibilidade do modelo.

- Ao cadastrar um livro as seguintes informações obrigatórias são preenchidas: ISBN, nome do livro, editora, au-

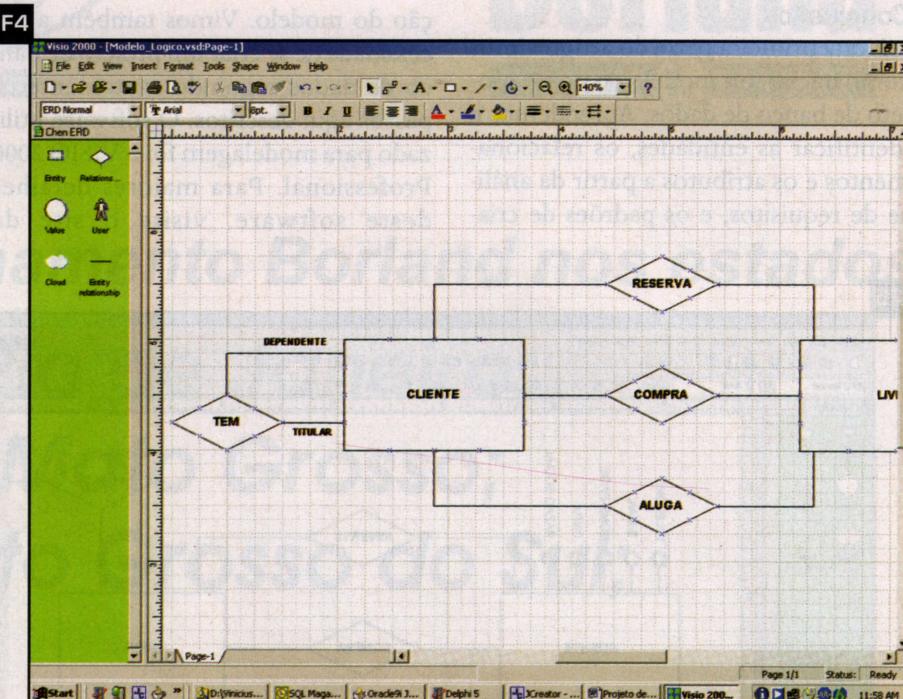


Modelo de Entidade-Relacionamento parcial para Book.net

tor, ano de publicação, assunto, se é para compra ou aluguel, se é nacional ou importado e quantidade que a livraria possui. As informações opcionais são preço de venda, preço de aluguel e preço de renovação de aluguel.

Os atributos ISBN, nome, autor, ano publicação, aluguel/compra, nacional/importado, quantidade, preço de venda, preço de aluguel e preço da renovação de aluguel são da entidade *Livro*. O atributo nome da editora pertence à entidade *Editora* e o atributo assunto pertence à entidade *Assunto*. Veja, nas **figuras 5 e 6**, como ficou o nosso modelo com os atributos em todas as entidades.

Os relacionamentos, de forma idêntica às entidades, possuem atributos. A identificação de atributos para relacionamentos será vista no próximo artigo.



Modelo de Entidade-Relacionamento parcial para Book.net

Kit Firebird & Interbase®

www.FireBase.com.br



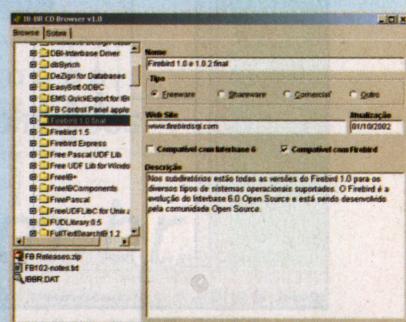
Para você que já utiliza ou pretende utilizar o Interbase ou o Firebird como banco de dados em suas aplicações, reunimos em um único CD tudo que você precisa para tirar o máximo proveito desses SGDBs. O kit contém softwares freewares, sharewares, versões de avaliação e trials dos melhores utilitários e ferramentas existentes para o IB/FB, entre elas :

- Firebird 1.0 final e 1.5beta (Open Source)
- Interbase 6.0.2 (Open Source)
- Versões de avaliação do Interbase 5.x , 6.5 e 7.0
- Ferramentas para manutenção e administração (IBExpert, QuickDesk, IB/FB Workbench, etc...)
- Drivers ODBC e OLEDB, UDFs, Documentação
- Componentes de acesso nativo (IBO, IBX, FreelB+...)
- Ferramentas Case e muito mais !

Faça seu pedido através do nosso site e receba o CD em sua casa com todo conforto e comodidade.

FirebirdSQL Foundation Sponsor
Estamos contribuindo para o desenvolvimento do Firebird

Visite nosso site e tenha acesso à artigos, dicas, lista de discussão, etc...



Navegue pelo conteúdo do CD através de um browser especialmente desenvolvido.

P rojeto de Banco de Dados: Modelo Lógico

Conclusão

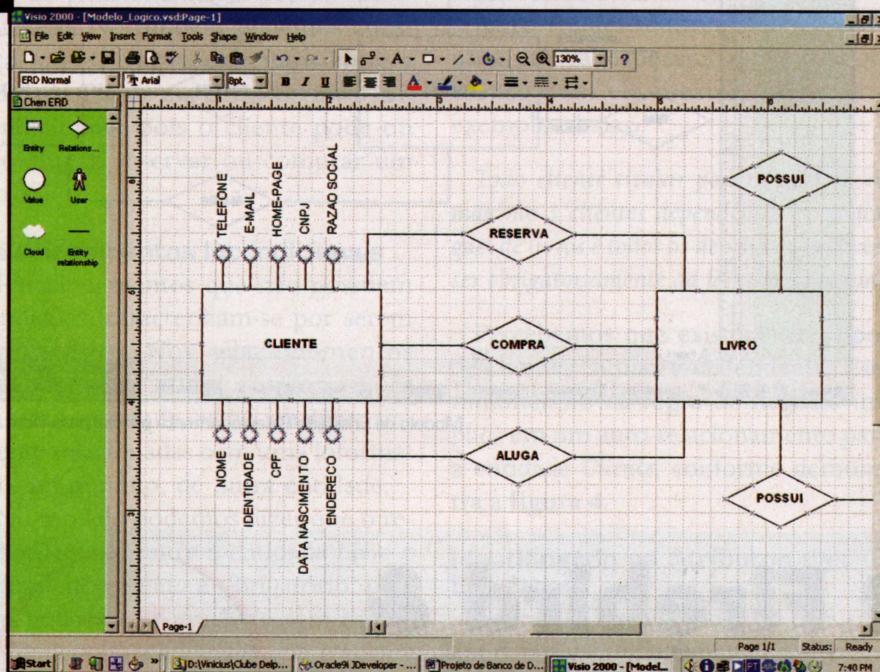
Nesta primeira parte do artigo vimos como iniciar um modelo do nosso projeto de banco de dados. Aprendemos a identificar as entidades, os relacionamentos e os atributos a partir da análise de requisitos, e os padrões de cria-

ção do modelo. Vimos também a necessidade de se criar um projeto de banco de dados seguindo exatamente os passos aqui descritos. O software utilizado para modelagem foi o VISIO 2000 Professional. Para maiores detalhes deste software, visite o site da

Microsoft.

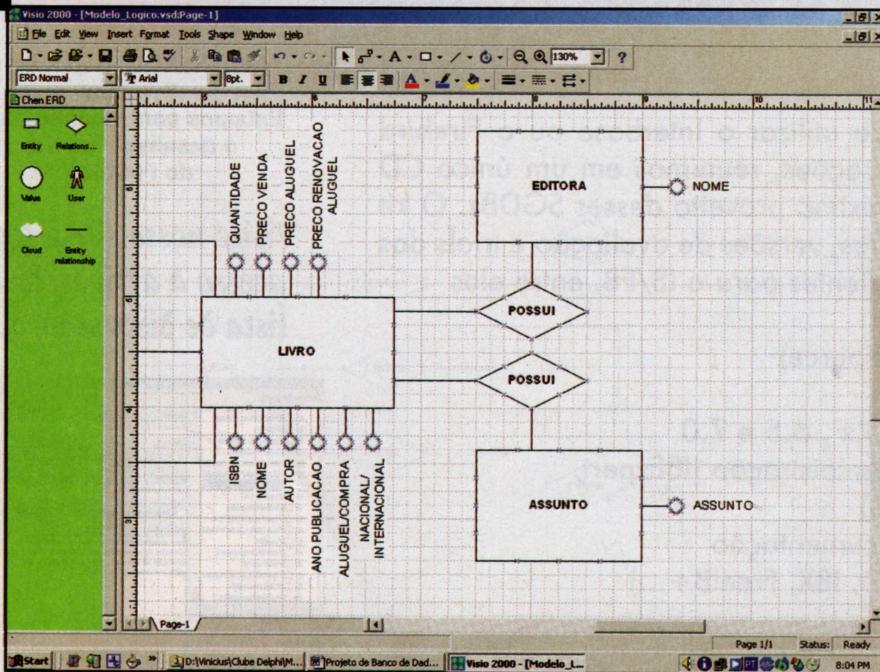
Na próxima edição continuaremos com a criação do nosso modelo de dados. Falarei sobre cardinalidades, atributos dos relacionamentos, tipos de atributos, especialização, generalização, entidades fracas e fortes.

F5



Modelo de Entidade-Relacionamento parcial para Book.net

F6



Modelo de Entidade-Relacionamento parcial para Book.net

A

Erica. Projeto de Banco de Dados – Uma Visão Prática. Felipe Nery R. Machado e Maurício Pereira Abreu. 7^a edição – 2001.

Campus. Modelagem Conceitual e Projeto de Banco de Dados. Paulo Sérgio Cougo. 1997.

Campus. Projetos de Bancos de Dados Relacionais. Jan L. Harrington. 2002

Campus. Introdução a Sistemas de Bancos de Dados. Christopher J. Date. 7^a edição – 2000.

A

Hamilton Oliveira é Analista de Sistemas, Desenvolvedor de Software e Mestrando em Engenharia de Sistemas e Computação. Pode ser contactado em:

hamilton@sqlmagazine.com.br

A

Vinicius Lourenço de Souza é analista de sistemas e desenvolvedor Delphi e Java em projetos Web e off-line, utilizando bancos de dados InterBase 6, Oracle 9 e DBA2/AS400 na DBA Engenharia de Sistemas. É pós-graduado em Análise, Projetos e Gerência de Sistemas na PUC-RJ e possui certificação BrainBench de Delphi e RDBMS. Pode ser contactado em vsouza@dba.com.br

P rojeto

de Banco de Dados

Parte 2: Modelo Lógico

Vinicius Lourenço de Sousa

Continuando a construção do projeto de banco de dados da livraria Book.Net, veremos mais alguns conceitos do modelo lógico, como atributos identificadores, modelagem de cardinalidades, atributos dos relacionamentos e especialização/generalização.

Atributos Identificadores

No modelo lógico, um atributo identificador serve para tornar uma informação exclusiva no contexto da entidade. Esse atributo tem preenchimento obrigatório e não pode conter valores repetidos. Observe que esse elemento não é o mesmo que uma chave primária, pois ainda estamos no modelo lógico. O atributo identificador é uma representação abstrata da regra de negócio, ou seja, do domínio do problema. A chave primária é uma representação física, pertencente ao domínio da solução. Nem sempre os dois serão representados pelo mesmo campo, da mesma forma que uma tabela nem sempre será representada por uma entidade equivalente.

Na entidade *Cliente* não temos um atributo que sirva como identificador único. O atributo nome não pode ser utilizado pois é obrigatório somente para clientes pessoa física. Dessa forma,

podemos criar um atributo identificador, o qual chamaremos de **Código**. Este elemento deve ser representado no diagrama por um círculo preenchido (**figura 1**).

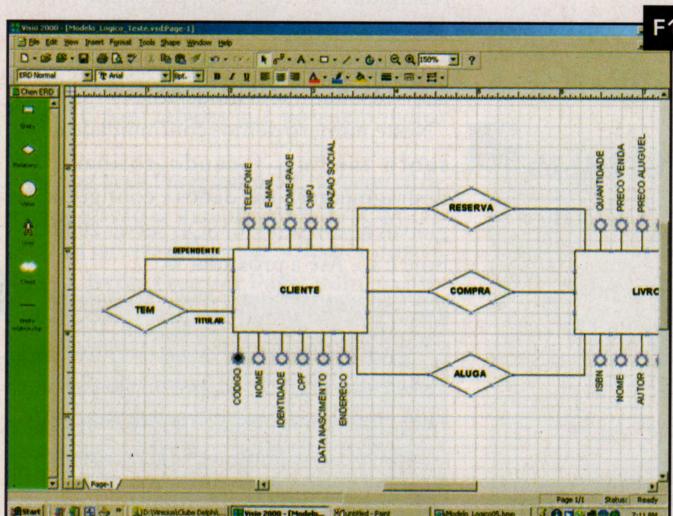
Na entidade *Livro*, o identificador será o atributo **ISBN**. O identificador da entidade *Editora* será **Nome** e a entidade *Assunto* será identificada pelo atributo **Assunto**.

Identificando Cardinalidades

A cardinalidade serve para detalhar o relacionamento, indicando a quantidade de ocorrências mínima e máxima de cada entidade. Por exemplo, no relacionamento entre *Livro* e *Editora*, cada livro possui no mínimo e no máximo uma editora e uma editora pode publicar no mínimo 0 (zero) e no máximo vários livros (de acordo com a regra de negócio uma editora pode estar cadastrada sem livros relacionados). Dessa forma, a relação da entidade *Livro* é de **um-para-um**, ou **1:1**, e a relação da entidade *Editora* é de **zero-para-vários**, ou **0:N**. A cardinalidade é definida como a ocorrência máxima de cada entidade no relacionamento. Na entidade *Livro*, a ocorrência máxima é 1 (um) e na entidade *Editora* a ocorrência máxima é N (vários), caracterizando uma cardinalidade de **um-para-vários**, ou **1:N**.

Para representar a cardinalidade no diagrama, deve-se colocar a relação da entidade A junto à entidade B, e vice-versa. Por exemplo, a relação 1:1, da entidade *Livro*, é inserida junto a entidade *Editora* e a relação 0..N é inserida junto a entidade *Livro*. Observe o diagrama alterado, com a inserção das relações entre a entidade *Livro* e *Assunto*, na **figura 3**.

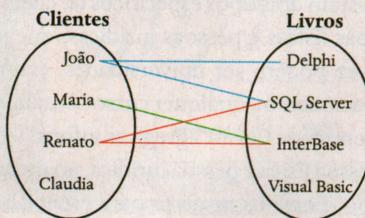
A identificação de cardinalidade nem sempre é simples. Vejamos outro exemplo: no relacionamento COMPRA, um cliente pode comprar nenhum, um ou vários livros. Continuando o raciocínio, um livro pode ser comprado por nenhum, um ou vários clientes (levando em consideração que a livraria tem várias cópias do mesmo livro, já que nos referenciamos apenas a “entidade” livro). Concluímos que a entidade *Cliente* tem um relacionamento de no mínimo 0 (zero) e no máximo N (vários) livros. A entidade



A cardinalidade também é conhecida como **Grau de Relacionamento**.

Livro tem no mínimo 0 (zero) e no máximo N (vários) clientes. Dessa forma, os clientes podem ter muitos livros e vice-versa, caracterizando uma cardinalidade de **muitos-para-muitos**, ou N:N.

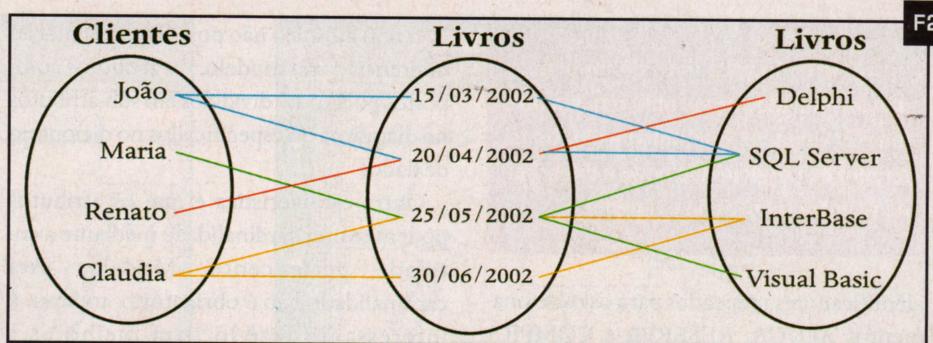
Uma forma mais simples de visualizar este grau de relacionamento é através do uso de conjuntos:



Vemos que a cliente **Claudia** está cadastrada, mas não comprou nenhum livro. Isso significa que a quantidade mínima de livros que podem ser comprados é 0 (zero). O livro **Visual Basic** não foi comprado, demonstrando que a quantidade máxima de clientes que podem comprar um livro é 0 (zero). Já o cliente **Renato** comprou dois livros e a cliente **Maria** comprou 1 livro, comprovando que a quantidade máxima de livros comprados durante o tempo pode variar ao infinito. Repare também que o mesmo título pode ser adquirido por vários clientes.

O diagrama contendo a representação de cardinalidade dos relacionamentos COMPRA, ALUGUEL e RESERVA pode ser visualizado na figura 4.

Um detalhe importante é que um cliente pode alugar, no máximo, três livros de uma vez. Não definimos a cardinalidade máxima



como "3", ao invés de "N", porque o valor "3" se refere a um aluguel específico. O cliente pode alugar mais de três livros ao longo do tempo.

Já no auto-relacionamento um cliente tem no máximo 3 dependentes. Essa restrição não é vinculada ao tempo, ou seja, é válida para sempre. Assim, a cardinalidade do auto-relacionamento da entidade Cliente é representada como 0..3 (figura 4).

Atributos dos Relacionamentos

Um relacionamento também pode conter atributos, com o intuito de representar requisitos da regra de negócio. Veja o exemplo:

Os clientes só podem alugar, no máximo, três livros de uma vez, cuja duração não poderá ultrapassar o período de duas semanas. Depois desse tempo, o cliente pode renovar o aluguel. Em toda renovação, devará ser paga uma taxa para cada livro. Caso o cliente desse do tempo e não renove o aluguel, pagará uma multa na devolução e/ou renovação.

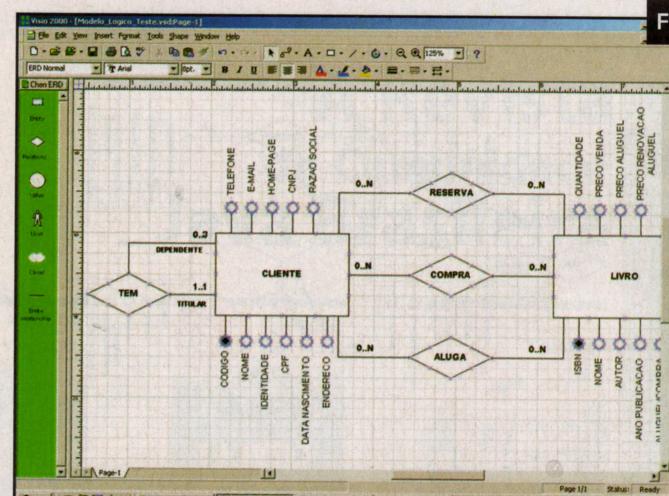
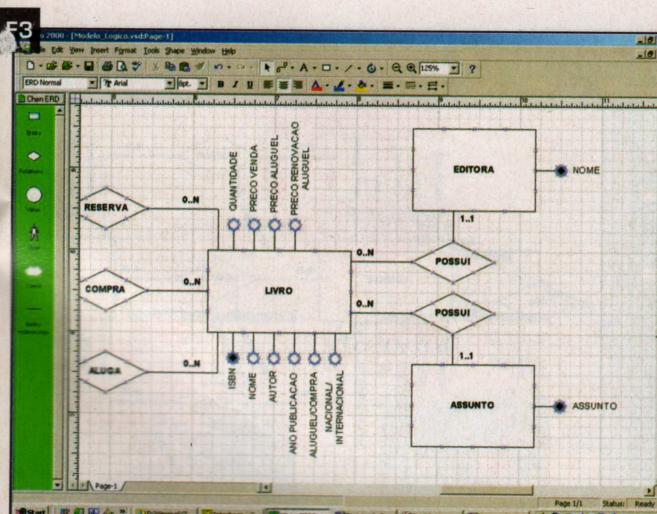
Os clientes que mais alugaram e mais compraram livros são notificados antes dos demais sobre as promoções, tanto para compra como para aluguel.

Na primeira regra de negócio percebemos que no aluguel de um livro algumas informações são necessárias, como período de aluguel, data de devolução e valor de multa. Na segunda regra vemos a necessidade de um atributo *Promoção*. Essas informações serão modeladas como atributos do relacionamento ALUGA (figura 5).

Assim como nas entidades, um relacionamento deve possuir um atributo identificador. Em cardinalidades **muitos-para-muitos** (N:N), que normalmente representam relacionamentos que acontecem ao longo do tempo, esse atributo será do tipo data/hora. Para facilitar a visualização, vamos utilizar novamente os conjuntos (figura 2).

Neste exemplo vemos que a data de aluguel é necessária para identificação exclusiva do relacionamento entre clientes e livros.

O diagrama com os atributos e



N

Um atributo para data de renovação não foi incluído porque uma renovação é um novo aluguel. Por motivos didáticos, a regra de negócio não pede a diferenciação entre renovação e aluguel.

identificadores mapeados para os relacionamentos ALUGA, RESERVA e COMPRA pode ser visualizado na figura 5.

Tipo e Cardinalidade de Atributos

Os atributos são classificados em três grupos, com duas opções cada. Veja as definições a seguir:

Atômico – Não pode ser dividido em sub-atributos. Exemplo: CPF e Identidade.

Composto – Pode ser dividido em atributos atômicos, para melhor compreensão do diagrama e implementação no banco de dados físico. Exemplo: Telefone e Endereço.

Mono-valorado – Possui apenas um domínio de valor dentro da informação. Exemplo: Identidade e Data Nascimento.

Multi-valorado – Possui mais de um domínio de valor. Exemplo: E-mail, Telefone e Home-Page.

Opcional – Não possui preenchimento obrigatório. Exemplo: E-mail e Home-Page.

Mandatório – Preenchimento obrigatório. Exemplo: Código, da entidade Cliente, e Assunto, da entidade Assunto.

O tipo atômico não possui representação diferenciada no modelo. Os atributos compostos podem ser divididos em sub-atributos no diagrama ou especificados no dicionário de dados.

Outra característica é que os atributos podem conter cardinalidade mediante a entidade pertencente. Modelar esta cardinalidade não é obrigatório, todavia, é interessante fazê-lo para melhorar a legibilidade, pois novas tabelas no modelo físico podem ser criadas a partir desta definição. Observe na figura 6 a representação de cardinalidade para os atributos Telefone, Email, Home-Page e Endereço. Na figura 8 podemos visualizar a cardinalidade dos demais atributos do modelo.

Especialização e Generalização

O modelo lógico oferece ainda o recurso de divisão ou agrupamento de entidades, de acordo com a necessidade de oferecer mais detalhes sobre os itens levantados na análise. Por exemplo, suponhamos que, em um projeto, foi levantada uma entidade de nome ANIMAL. Para cada tipo de animal, o sistema terá comportamentos e relacionamentos distintos. Para deixar o modelo mais próximo do domínio do problema, podemos desmembrar a entidade ANIMAL em cinco sub-entidades: Mamíferos, Répteis, Anfíbios, Aves e Peixes. A modelagem desse detalhamento é chamada de *Especialização*.

No caminho inverso, o agrupamento de entidades especialistas em uma entidade genérica é chamado de *Generalização*.

No modelo, podemos especializar a entidade *Cliente* de acordo com o requisito a seguir:

Os clientes podem ser pessoas físicas ou jurídicas. Os livros também são vendidos para universidades, centros educacionais e quaisquer cursos que desejam melhorar sua biblioteca.

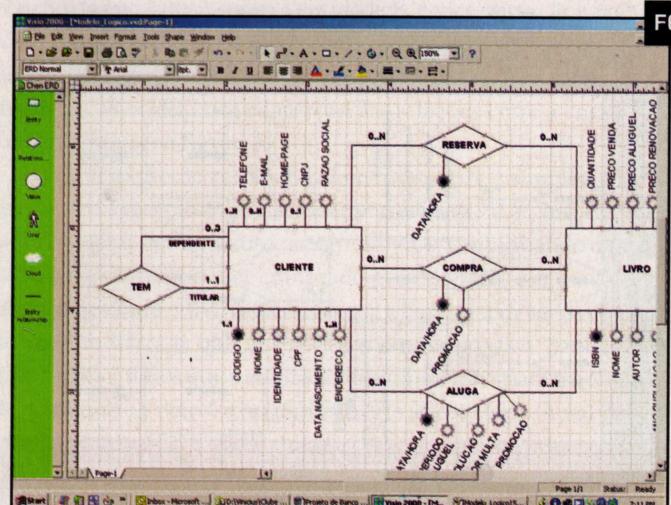
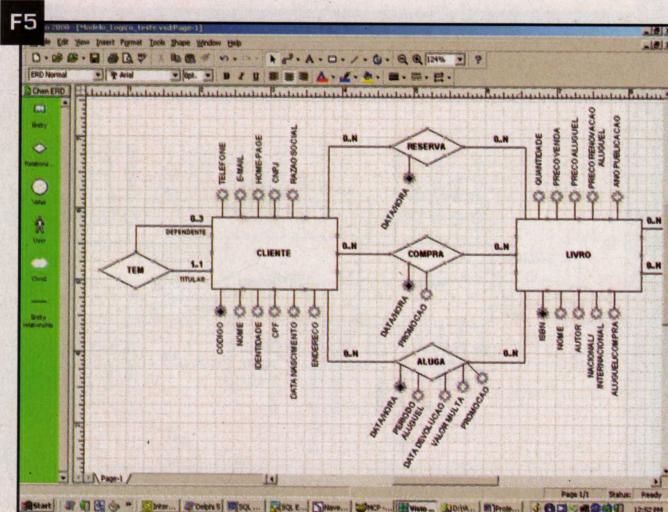
Existem dois tipos específicos de clientes: pessoas físicas e pessoas jurídicas, que por sua vez podem ser universidades, centros educacionais ou qualquer curso. Atualmente, a entidade *Cliente* agrupa as informações de pessoa física e pessoa jurídica, ao mesmo tempo. Entre os motivos para especializar essa entidade, podemos destacar:

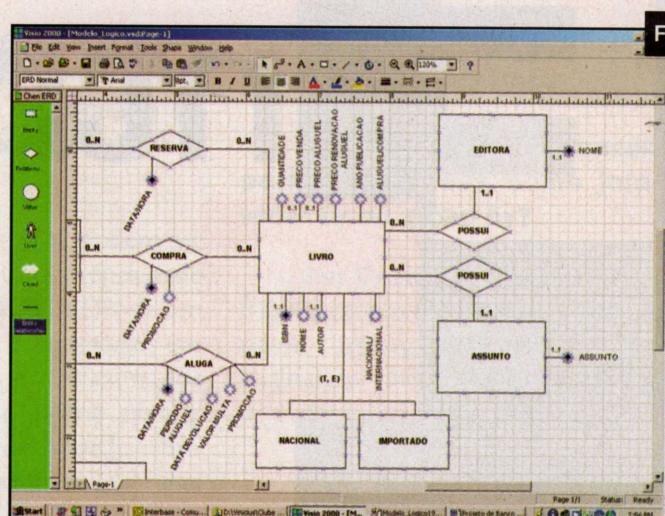
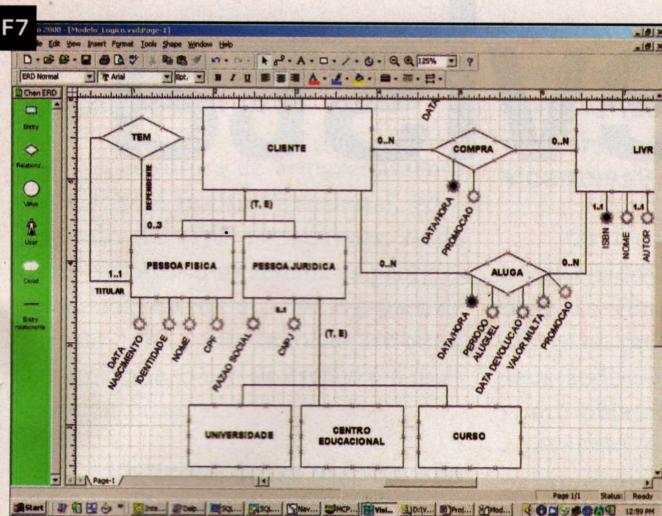
1- Se o cliente for pessoa jurídica, não terá dependentes. A especialização dá uma representação mais fiel do comportamento do sistema, já que o auto-relacionamento é exclusividade de um tipo de cliente;

2- Algumas propriedades são exclusivas de pessoa física enquanto outras são exclusivas de pessoa jurídica;

3- O modelo ficará mais legível para o resto da equipe e a quantidade de informação não documentada será menor.

Uma vez feita a especialização da entidade *Cliente*, podemos distribuir os atributos em suas respectivas sub-entidades. As figuras 7 e 8 contêm o diagrama com todas as especializações.





Tipos de Especialização

Existem três tipos de especialização no modelo lógico:

Total (T): É quando todas as possíveis sub-entidades estão modeladas. A especialização da entidade *Cliente* é total, pois todo cliente será pessoa física ou pessoa jurídica.

Parcial (P): É quando modelamos ape-

nas as principais sub-entidades, deixando as menos importantes de fora. Embora não apareçam no modelo, essas sub-entidades devem constar no dicionário de dados. Um exemplo seria a especialização de uma entidade *Funcionário* nas sub-entidades *Programador* e *Gerente*, já que nem todo funcionário é programador ou gerente.

damos a especialização/generalização, uma técnica poderosa para modelagem do comportamento do sistema.

No próximo número daremos continuidade ao modelo lógico, mostrando as restrições de integridade, regras de derivação, agregação, entidades fortes e fracas e dicionário de dados. ■

A especificação definida por Peter Chen (criador do Modelo Entidade-Relacionamento), indica que a especialização deve ser representada por um triângulo, posicionado na interseção das linhas que ligam a entidade com suas sub-entidades. Em algumas literaturas, o triângulo não é utilizado. Omitimos o símbolo neste artigo pois o software VISIO 2000 não possui essa representação.

Exclusiva (E): Indica que uma entidade possui uma especialização de cada vez. Por exemplo, se um cliente é pessoa física, não pode ser pessoa jurídica e vice-versa. Um livro não pode ser nacional e importado ao mesmo tempo.

As figuras 7 e 8 mostram os tipos de especialização modelados.

Conclusão

Nesta edição vimos a importância dos atributos identificadores e das cardinalidades. Conhecemos os tipos dos atributos e estu-

Vinicius Lourenço de Sousa é analista de sistemas/desenvolvedor em Delphi e Java em projetos WEB e Off-Line utilizando os banco de dados Interbase 6, Oracle 9 e DB2/AS400 na DBA Engenharia de Sistemas. É Pós-Graduado em Análise, Projetos e Gerência de Sistemas na PUC-RJ e possui certificação BrainBench de Delphi e RDBMS. Pode ser contatado no e-mail vsouza@dba.com.br.



A Revista da Comunidade **Java Brasileira**

www.javamagazine.com.br
info@javamagazine.com.br

projeto

de Banco de Dados

Parte 3 - Modelo Lógico

por Vinicius Lourenço de Sousa

Continuando a construção do projeto de banco de dados da livraria Book.NET, veremos mais alguns conceitos do modelo lógico, como restrições de integridade (RI), regras de derivação (RD), agregação, entidades fortes e fracas e dicionário de dados.

Restrições de Integridade

A princípio, qualquer regra do sistema pode ser qualificada como uma restrição de integridade. Por exemplo, o fato de um cliente não possuir mais de três dependentes ou um livro não possuir mais de um código ISBN são restrições de integridade. Essas duas regras já estão representadas no diagrama, através dos elementos básicos usados até aqui – a primeira está definida pela cardinalidade do auto-relacionamento de *Cliente* e a segunda é indicada pelo atributo identificador *ISBN*. No entanto, os elementos principais do diagrama não são suficientes para representar algumas restrições. Veja dois exemplos:

Os clientes só podem alugar no máximo três livros de uma vez, cuja duração não poderá ultrapassar o período de duas semanas.

Caso o cliente passe do tempo do aluguel e não o renove, pagará uma multa na devolução e/ou na renovação.

Para representar essas regras no diagrama podemos utilizar o símbolo RI, junto ao relacionamento *ALUGA*, conforme mostra a figura 1. Observe que a sigla possui um número seqüencial, indicando que o modelo pode conter várias restrições. No dicionário de dados devemos descrever cada RI, indicando seu número e a regra de negócio em questão. O ideal é que cada regra possua sua própria RI, facilitando a documentação no dicionário e a identificação da regra pelas pessoas envolvidas no projeto.

Representar a restrição de integridade no diagrama não é obrigatório, mas sua descrição no dicionário de dados é imprescindível.

Restrições de Integridade em Atributos

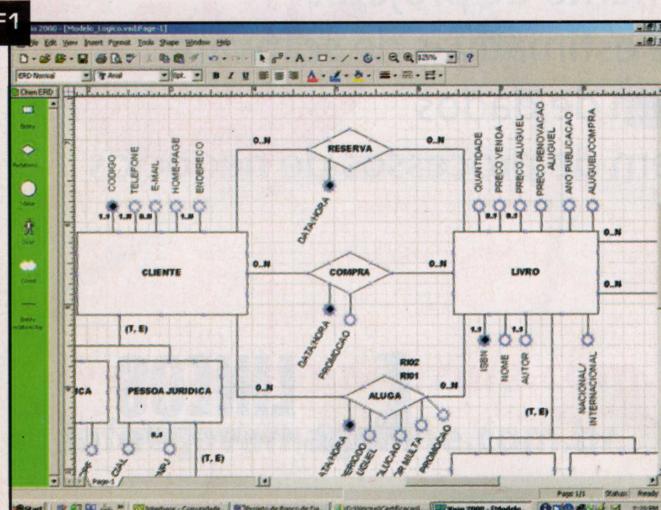
Embora sejam encontradas com mais freqüência em relacionamentos, as RIs também podem ser definidas para atributos. A idéia é a mesma: descrever uma regra de negócio que não seja possível representar com os elementos básicos do diagrama. Para exemplificar, usaremos o atributo *valor multa*, do relacionamento *ALUGA*. O cliente que ultrapassar duas semanas no aluguel de um ou mais livros deve pagar uma multa pelo atraso. Percebemos que o valor do atributo será preenchido somente se existir algum atraso na devolução do livro, representando uma restrição de integridade.

Outro atributo que também possui uma restrição é *período aluguel*, que não pode ser superior a duas semanas. Veja na figura 2 o modelo com as restrições de integridade nos atributos.

Regras de Derivação

Regras de derivação descrevem atributos cujo valor depende de outro atributo ou de um cálculo. No diagrama, o símbolo RD é utilizado para identificar esse tipo de regra. A descrição completa da fórmula utilizada para compor o valor do atributo deve ser documentada no dicionário de dados.

Por exemplo, para modelar o valor total de uma compra, vamos acrescentar dois atributos no relacionamento *COMPRA*: *quantidade comprada* e *total* (o atributo preço já está modelado na entidade *Livro*). Esse último contém uma regra de derivação: seu valor é calculado através da soma da quantidade de livros comprados multiplicado pelo preço de venda e subtraído pelo desconto de promoção, caso o cliente tenha direito. Na figura 3 vemos o símbolo RD



junto ao atributo *total*. Observe que, assim como na restrição de integridade, devemos numerar seqüencialmente cada regra de derivação do modelo. A descrição do cálculo do valor total será incluída no dicionário de dados.

Agregação

Até agora vimos relacionamentos apenas entre entidades, mas nada impede que haja relacionamentos entre relacionamentos. Por exemplo, imagine que o modelo inclua um controle de notas fiscais, que deverão ser emitidas para cada compra. A nota fiscal será uma nova entidade, possuindo as seguintes informações: uma numeração única, a data da compra, o cliente que comprou, os livros que foram comprados, a quantidade de cada livro, o valor de venda de cada livro e o valor total da compra. A nota será gerada apenas quando uma compra for efetuada, portanto, a entidade depende do relacionamento COMPRA. Veja a representação de conjuntos na figura 4.

Para modelar esse cenário utilizamos uma abstração, que trata um relacionamento como uma entidade de nível superior. Essa técnica, chamada de agregação, é representada no diagrama como um retângulo que envolve todos os componentes de um relacionamento, como podemos observar na figura 5. Em algumas literaturas, esse retângulo aparece pontilhado.

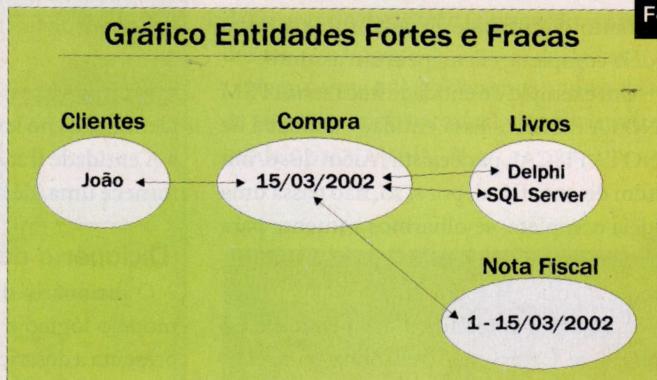
A linha que liga a entidade *Nota Fiscal* com a agregação vai até o retângulo, indicando que existe um relacionamento com COMPRA.

PRA. Assim como em relacionamentos entre entidades, também devemos representar as cardinalidades. Nesse caso, uma nota fiscal é gerada por uma compra e uma compra gera apenas uma nota fiscal (1:1).

O retângulo da agregação deve conter apenas um relacionamento. Se precisarmos modelar o relacionamento entre *Nota Fiscal* e ALUGA devemos criar outro retângulo de agregação para ALUGA. Essa é uma limitação do MER: o diagrama tende a ficar pesado e pouco legível, se possuir muitas agregações. Uma forma de diminuir a poluição visual é colocar as entidades e relacionamentos em posições que facilitem o desenho do retângulo, ou mesmo substituí-lo por um polígono.

Outro fato importante é que o relacionamento que será agregado (no caso, COMPRA), terá sempre cardinalidade N:N. O motivo é que apenas nesse caso encontramos uma relação que seja relevante ao ponto de permitir um relacionamento próprio, como COMPRA, ALUGA ou RESERVA. Nas cardinalidades 1:N ou 1:1, os relacionamentos não possuem a importância necessária para que possam, eles próprios, se relacionarem. Veja um exemplo na relação entre *Livro* e *Editora*: não faz sentido uma entidade se relacionar com POSSUI, neste contexto.

Gráfico Entidades Fortes e Fracas

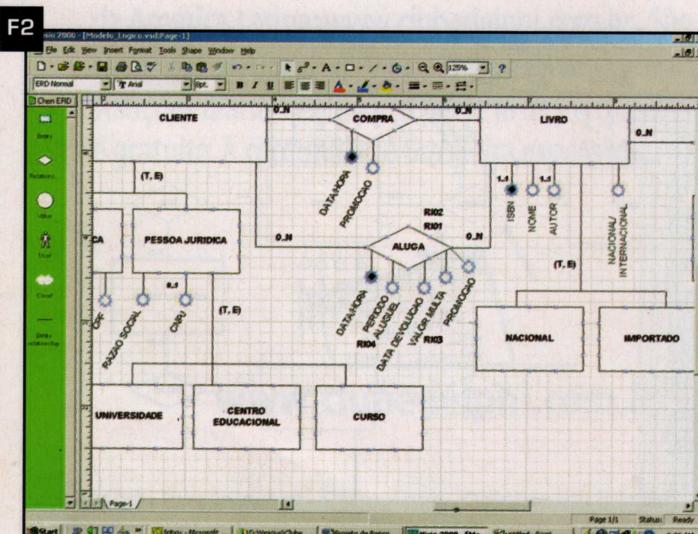


Outro detalhe são os atributos de *Nota Fiscal*: apenas *Número* foi modelado, pois as demais informações já estão no relacionamento COMPRA.

Entidades Fortes e Fracas

Em alguns casos, uma entidade depende de outra entidade para existir. Quando isso acontece, chamamos a entidade dependente de **entidade fraca** e a entidade principal de **entidade forte**. A relação entre essas entidades é conhecida como **dependência existencial**.

Também podemos dizer que uma entidade fraca é aquela que não apresenta uma visão íntegra, pois precisa da informação de outra entidade para completar a idéia. Por exemplo, a entidade *Cliente* é forte porque não precisamos de nenhuma outra entidade para completar a idéia de *cliente*. Esse conceito é fundamental para identificar corretamente as entidades fracas. Por exemplo, poderíamos dizer que a entidade *Assunto* depende da entidade *Livro* para existir. No



P rojeto de Banco de Dados - Parte 3: Modelo Lógico

entanto, a entidade *Assunto* fornece uma idéia completa: ela mapeia um assunto.

Um exemplo de entidade fraca seria *ITEM NOTA FISCAL*. Essa entidade depende de *NOTA FISCAL* para existir; Além disso, um ítem de nota fiscal, por si só, não passa uma idéia completa; se olharmos somente para essa entidade não temos uma visão consistente do que ela representa.

A figura 6 apresenta a nova entidade no diagrama. Observe que o relacionamento *POSSUI*, entre as entidades *Nota Fiscal* e *Item Nota Fiscal*, é representado com duas linhas, indicando uma dependência existencial. A entidade fraca também é representada com duas linhas, como podemos ver em *Item Nota Fiscal*. Como um item de nota possui um livro, essa entidade também se relaciona com *Livro*.

A figura 7 mostra o modelo completo.

N No Visio, para criar a representação da entidade fraca, basta clicar com o botão inverso do mouse sobre a entidade e selecionar a opção **Set as Dependent Entity**. Essa opção só está disponível para a entidade; a linha dupla do relacionamento deve ser feita manualmente.

Veja algumas dicas para identificar uma entidade forte/fraca:

- Toda entidade que estiver atrelada ao núcleo do negócio, será uma entidade forte. Na livraria, as entidades *Cliente* e *Livro* nunca serão fracas, pois pertencem a regra de negócio principal;
- Normalmente a entidade fraca é criada apenas para detalhar a entidade forte;

- A entidade fraca não existe sem a entidade forte;
- Muitas vezes, a entidade fraca não é identificada no levantamento de requisitos;
- A entidade fraca, vista isoladamente, não fornece uma idéia completa.

Dicionário de Dados

O dicionário de dados está presente no modelo lógico e no modelo físico. Ele representa a descrição detalhada do modelo e possui a mesma importância de qualquer documentação em um projeto: padronizar e facilitar a comunicação entre a equipe. Nele identificamos todas as informações relevantes, como entidades, atributos, relacionamentos, restrições de integridade, regras de derivação, obrigatoriedade do relacionamento, cardinalidades, tipos de atributos e qualquer outra informação que o analista julgue importante. Apesar de algumas literaturas especificarem padrões para a criação do dicionário, na prática, não há um modelo formal para sua construção. No final da série, veremos um dicionário de dados detalhado.

Sugestão para o Dicionário de Dados

Como em toda documentação, inicie descrevendo o objetivo do projeto, indicando que tipo de sistema está sendo modelado e quais regras de negócio ele visa atender. Em seguida, crie as seções abaixo:

Entidade – Descreva todas as entidades, informando seu objetivo, se é forte ou fra-

ca, com quais entidades ou relacionamentos (agregação) se relaciona e, se possui especialização ou generalização, detalhe seu tipo e as subentidades envolvidas. Para os relacionamentos descreva o nome do relacionamento e as cardinalidades (o relacionamento deve ser detalhado em outra seção).

Atributos - Para cada entidade, descreva todos os atributos, informando seu objetivo e seu tipo: obrigatório, multi-valorado, composto ou identificador. Defina sua cardinalidade e detalhe as regras de derivação e as restrições de integridade. Na regra de derivação informe a fórmula do cálculo e, caso seja necessário, quais atributos estão envolvidos. Na restrição de integridade descreva a regra em detalhes.

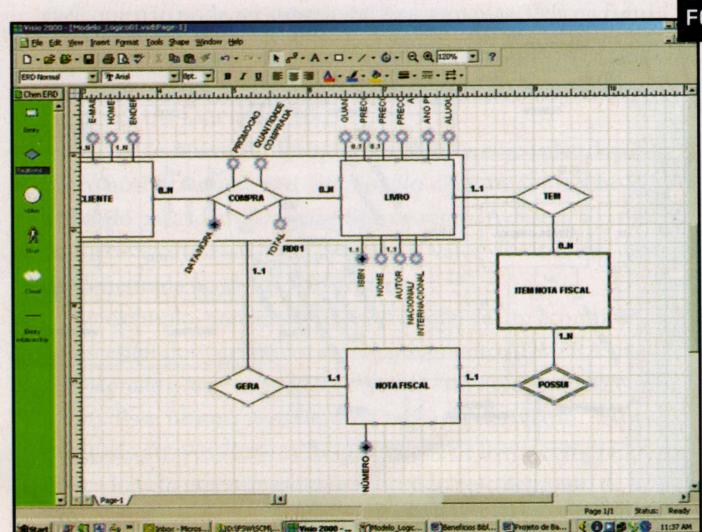
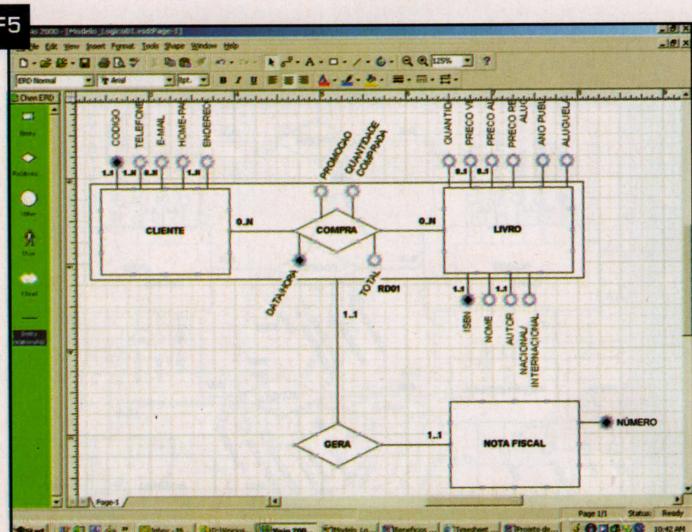
Relacionamentos – Para cada um, informe seu objetivo e descreva suas restrições de integridade, atributos, entidades envolvidas e se o relacionamento faz parte de uma agregação.

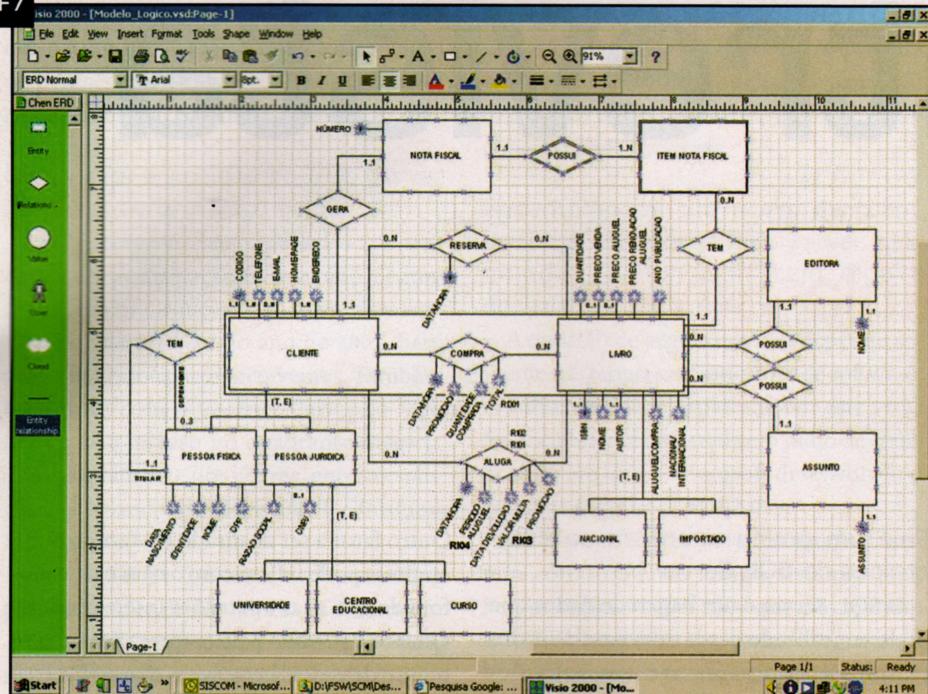
Ao final, adicione todos os comentários considerados importantes para o projeto ou para a comunicação entre a equipe.

N Como esse dicionário de dados pertence ao modelo lógico, não podemos usar termos como tabelas, campos ou tipos de campos.

Conclusão

Nesta edição chegamos ao final do modelo lógico. Vimos as restrições de integridade





de, regras de derivação, agregação, entidades fortes e fracas e dicionário de dados.

Na próxima edição daremos continuidade ao projeto de banco de dados iniciando o modelo físico. Até lá! ■

A

Vinicius Lourenço de Sousa é analista de sistemas/desenvolvedor em Delphi e Java em projetos WEB e off-line, utilizando os bancos de dados InterBase 6, Oracle 9i e DB2/AS400, na DBA Engenharia de Sistemas. É Pós-Graduado em Análise, Projetos e Gerência de Sistemas na PUC-RJ e possui certificação BrainBench em Delphi e RDBMS. Pode ser contatado no e-mail vsouza@dba.com.br.

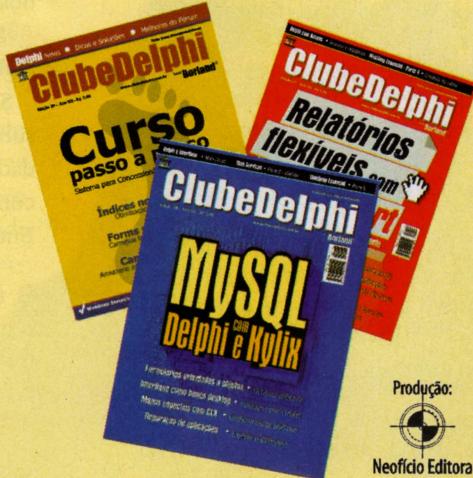
ClubeDelphi

Apoio **Borland®**

Conheça a maior comunidade de desenvolvedores Delphi da América Latina: www.clubedelphi.com.br. São mais de 30 mil cadastrados, trocando informações, dicas, componentes e soluções diariamente. Faça parte já deste clube. O cadastro é gratuito. A comunidade brasileira espera você!



www.clubedelphi.com.br



Produção:
Neofício Editora

Delphi Para Internet - Entrevistas - Delphi para Iniciantes - Kylix - Análise de Componentes - WebSnap - E muito mais!