MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-7524-TDI/732

ESTRATÉGIAS E PADRÕES PARA MODELAGEM DE BANCO DE DADOS PARA SISTEMAS BASEADOS NA ARQUITETURA SOFTBOARD

Letícia Teixeira Cottini

Dissertação de Mestrado em Computação Aplicada, orientada pelo Dr. João Bosco Schumann Cunha, aprovada em 25 de março de 1999.

INPE
São José dos Campos
2000

681.3.06

COTTINI, L. T.
Estratégias e padrões para modelagem de banco de dados para sistemas baseados na arquitetura softboard / L. T. Cottini – São José dos Campos: INPE, 1999.
154p. – (INPE-7524-TDI/732).

1.Banco de dados. 2.Orientação a objetos. 3.Softboard. 4.Padrões. 5.Estratégia. 6.Arquitetura de software. I.Título.

Aprovado pela Banca Examinadora em cumprimento a requisito exigido para a obtenção do Título de **Mestre** em **Computação Aplicada.**

Dr. Solon Venâncio de Carvalho — Presidente

Dr. João Bosco Schumann Cunha — Orientador

Dr. Tatuo Nakanishi — Membro da Banca

Drª Selma Shin Shimizu Melnikoff — Membro da Banca Convidada

Candidato (a) : Letícia Teixeira Cottini

São José dos Campos, 25 de março de 1999.

A Deus,
por ter me provido de toda
paciência e determinação necessárias,
dedico.

AGRADECIMENTOS

Agradeço ao Doutor João Bosco Schumann Cunha, pela orientação deste trabalho;

À Banca Examinadora que se propôs a analisar este trabalho e contribuir com suas sugestões;

Ao Instituto Nacional de Pesquisas Espaciais (INPE) pela oportunidade e apoio;

À Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo auxílio financeiro;

À minha família pelo constante incentivo durante toda a minha caminhada;

Ao pessoal do Laboratório de Computação da SPG pelo companheirismo e apoio;

E a todos que direta ou indiretamente contribuíram para a realização deste trabalho.

RESUMO

Visando a uma melhoria na qualidade dos sistemas de software e na produtividade dos seus desenvolvedores, este trabalho de pesquisa define um processo de desenvolvimento de um sistema de banco de dados. Tal processo se baseia em estratégias e padrões para a modelagem de um banco de dados relacional, que é usado por uma aplicação orientada a objetos baseada na SOFTBOARD. Este trabalho de pesquisa define também um modelo de metadados do catálogo, que o Componente Gerenciador de Configuração de Sistema Orientado a Objetos (CMOOS) da SOFTBOARD, utiliza para que seja feita a interface entre o Componente Domínio do Problema (CDP) da aplicação, o Componente Gerenciador de Dados (CGD) e o banco de dados, bem como um conjunto de estratégias para alimentar este catálogo.

STRATEGIES AND PATTERNS FOR SOFTBOARD-BASED SYSTEM DATABASE MODELING

ABSTRACT

Looking for more software systems quality and the developers productivity, this research define a database system development process based on strategies and patterns to model a relational database used by a SOFTBOARD-based object oriented application. This research also defines a metadata catalog model which is used by the SOFTBOARD CMOOS to interface the application CDP, the CGD and the database, as well as a set of strategies to mantain this catalog.

SUMÁRIO

Pág.

LISTA DE FIGURAS

LISTA DE TABELAS

Pág.

LISTA DE SIGLAS E ABREVIAÇÕES

ANSI	-	*American National Standard Institute* (Instituto Nacional Americano de Padrões)
BCNF	-	*Boice Codd Normal Form* (Forma Normal de Boyce Codd)
BDEC	-	Banco de Dados de Elementos Controlados
BDOO	-	Banco de Dados Orientado a Objetos
BDR	-	Banco de Dados Relacional
CDP	-	Componente Domínio do Problema
CGC	-	Componente Gerenciador de Cenários
CGD	-	Componente Gerenciador de Dados
CIS	-	Componente de Interface entre Sistemas
CMOOS	-	Componente Gerenciador de Configuração de Sistema Orientado a Objetos
GoF	-	*The Gang of Four*
GoV	-	*Siemens Gang of Five*
OID	-	*Object Identifier* (Identificador de Objeto)
OSQL	-	*Object Structured Query Language* (Liguagem Estruturada de Consulta Orientada a Objetos)
SGBD	-	Sistema Gerenciador de Banco de Dados
SOFTBOARD	-	Arquitetura Configurável para Sistema de Software Orientado a Objetos
UML	-	*Unified Modeling Language* (Linguagem de Modelagem Unificada)
SQL	-	*Structured Query Language* (Linguagem Estruturada de Consulta)

CAPÍTULO 1

INTRODUÇÃO

Organizações que desenvolvem software ainda se deparam com os seguintes problemas:

- as estimativas de prazo e de custo frequentemente são imprecisas;
- a produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços;
- a qualidade de software, às vezes é inadequada.

A orientação a objetos oferece um novo paradigma de desenvolvimento de aplicações, prometendo melhorias de qualidade e produtividade, principalmente através da reutilização.

Na busca de um incremento na reutilização, Coad e Yourdon (1992) propuseram uma arquitetura para sistemas de software composta de cinco componentes, com responsabilidades bem definidas. Tais componentes são: Componente Interface Humana (CIH), Componente Interface com outros Sistemas (CIS), Componente Gerenciador de Cenários (CGC), Componente Gerenciador de Dados (CGD) e Componente Domínio do Problema (CDP).

Dando sequência nesta busca por um incremento na reutilização, percebe-se que o ganho poderia ser maior se houvesse uma arquitetura configurável para o sistema de software.

Desta forma, aproveitando a idéia de Coad e Yourdon, Cunha (1997) propôs um passo mais à frente na direção da reutilização, definindo uma arquitetura configurável para sistema de software orientado a objetos, denominada de SOFTBOARD.

Nesta arquitetura, os componentes CIH, CIS, CGC e CGD são configuráveis e se adaptam ao CDP da aplicação que estiver sendo utilizada naquele momento. Para que os componentes sejam configuráveis, Cunha propôs a inclusão de um novo componente na arquitetura proposta por Coad e Yourdon, o Componente Gerenciador de Configuração de Sistema Orientado a Objetos (CMOOS).

O CMOOS exerce uma importante função na SOFTBOARD, pois disponibiliza todas as informações de configuração necessárias para que os componentes CIH, CIS, CGC e CGD possam ser totalmente reutilizados pelo CDP em questão.

Detalhando a arquitetura configurável de sistema de software orientado a objetos, proposta por Cunha, foram desenvolvidos os seguintes trabalhos de pesquisa: Componente Gerenciador de Dados Configurável (Ferreira, 1996) e Interface Configurável (Souza, 1997).

No trabalho de pesquisa, Componente Gerenciador de Dados Configurável (Ferreira, 1996), o componente CGD foi especificado e projetado. E ainda, foi apresentado um protótipo inicial do CMOOS.

Mesmo com a SOFTBOARD reutilizável, onde o CGD é o responsável pela gestão do banco de dados da aplicação, encontra-se a necessidade de se desenvolver o projeto desse banco de dados e alimentar o catálogo a ser utilizado pelo CMOOS com as informações de configuração, para que o CGD possa gerenciar esse banco de dados. Estes são os pontos tratados neste trabalho de pesquisa:

- Como projetar o banco de dados utilizado pelo sistema de software baseado em SOFTBOARD?
- Que informações manter no catálogo a ser utilizado pelo CMOOS e como alimentá-lo?

Buscando efetivar e definir um processo para o projeto do banco de dados a ser utilizado por uma aplicação baseada em SOFTBOARD, é proposto um conjunto de estratégias e padrões a serem seguidos. Portanto, uma meta desta pesquisa consistiu na descrição dessas estratégias e padrões.

Buscando definir que informações manter no catálogo a ser utilizado pelo CMOOS, para atender as necessidades do CGD, um modelo dessas informações é detalhado e um conjunto de estratégias é apresentado. Portanto, outra meta desta pesquisa consistiu em modelar as informações de configuração necessárias para o CGD e descrever as estratégias de como alimentá-las.

Este trabalho de pesquisa, através das metas descritas anteriormente, teve como objetivo mais geral dar continuidade à proposta de Cunha (1997) de uma arquitetura configurável de sistema de software orientado a objetos – SOFTBOARD e detalhar o trabalho de pesquisa: Componente Gerenciador de Dados configurável (Ferreira, 1996).

Os objetivos apresentados neste capítulo, podem ser identificadas na Figura 1.1.

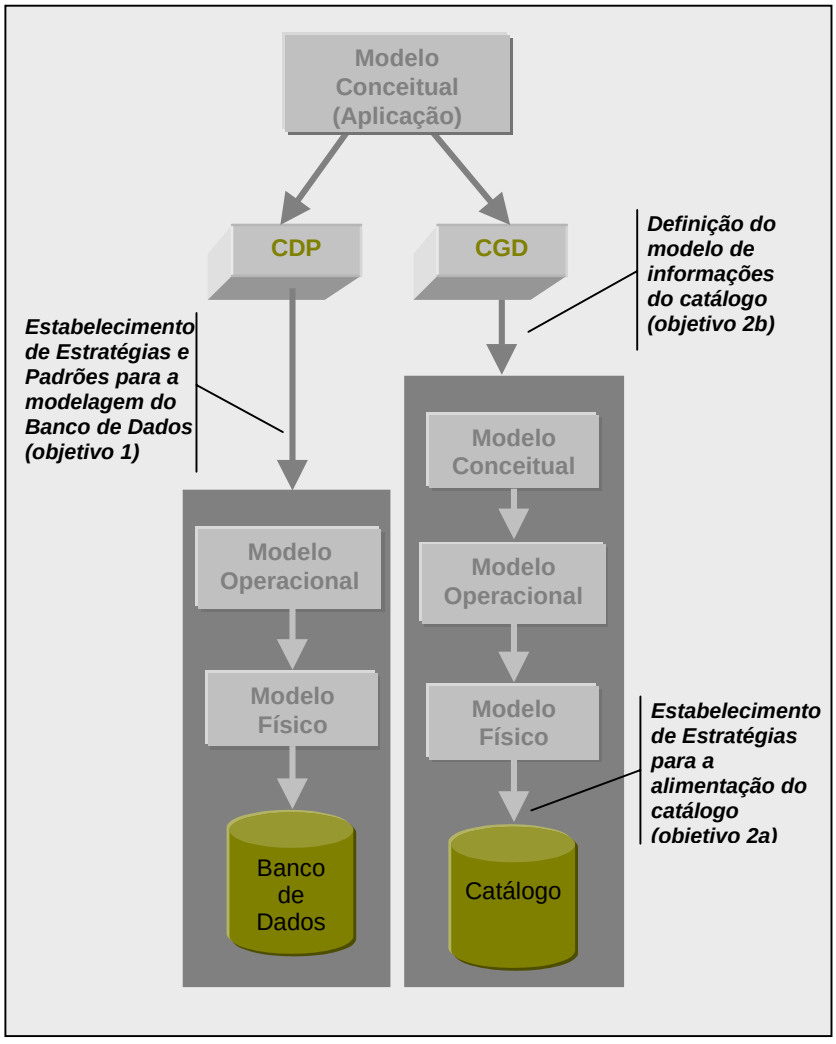


Fig. 1.1 - Identificação dos objetivos propostos neste trabalho de pesquisa.

Este trabalho de pesquisa encontra-se organizado do seguinte modo:

- No capítulo 2, descreve-se a fundamentação teórica utilizada como base para este trabalho de pesquisa.

- No capítulo 3, são descritas estratégias e padrões para a modelagem do banco de dados para uma aplicação baseada em SOFTBOARD.
- No capítulo 4, encontra-se a definição de um modelo de informações do catálogo, que atende às necessidades do CGD, e a descrição das estratégias para a alimentação desse catálogo.
- Na capítulo 5, encontram-se as principais conclusões deste trabalho, ressaltando-se suas contribuições.
- No apêndice A, encontra-se a bibliografia complementar utilizada durante a pesquisa. Nos apêndices B e C, encontram-se as notações utilizadas neste trabalho de pesquisa. No apêndice D, encontra-se o estudo de caso realizado com o intuito de teste e validação das estratégias e dos padrões para a modelagem do banco de dados, propostos no capítulo 3 e das estratégias de alimentação do catálogo, propostas no capítulo 4.

CAPÍTULO 2

FUNDAMENTAÇÃO

Neste capítulo, encontra-se a fundamentação teórica utilizada como base para este trabalho de pesquisa.

Primeiramente, descreve-se sobre o paradigma de orientação a objetos e a proposta de Cunha (1997) para uma arquitetura configurável de sistema de software orientado a Objetos - SOFTBOARD, enfatizando os componentes com maior relevância neste trabalho de pesquisa - CGD e CMOOS.

Em seguida, são descritos alguns tópicos relevantes a respeito de banco de dados. Dentre os quais: uma apresentação dos níveis de abstração envolvidos em um possível processo de modelagem de sistema de software, baseado em SOFTBOARD, levando à criação de uma base de dados.

Logo após, são descritas as principais diferenças entre a tecnologia de orientação a objetos e o modelo de dados relacional. E ainda, são apresentados alguns tópicos a respeito do mapeamento objeto-relacional.

Finalizando, descreve-se, brevemente, uma abordagem sobre a importância de padrões e estratégias atualmente na engenharia de software.

2.1 Arquitetura Configurável de Sistemas de Software Orientado a Objetos - SOFTBOARD

2.1.1 Paradigma de Orientação a Objeto

No contexto da Engenharia de Software, o termo Paradigma significa modelo para estruturar e representar problemas cuja solução se deseja obter através do computador. Os paradigmas de programação começaram a ser identificados e chamados como tais em meados da década de 60. Nesta época, reconhecia-se a existência do paradigma procedimental que encontrava expressão natural nas linguagens de programação e do paradigma funcional, onde soluções são expressas através de composições de funções. Na década de 80, surge um novo paradigma: o orientado a objetos.

Os benefícios advindos do emprego do paradigma de objetos são vários, como abstração de dados/encapsulamento, modularidade, hierarquia, reutilização de software, facilidade maior de extensão e manutenção (Buzato e Rubira, 1998).

A expressão orientação a objetos significa, de maneira geral, que o software é organizado como uma coleção de objetos distintos, que incorporam tanto o comportamento quanto a estrutura de dados (Rumbaugh, 1991).

2.1.2 Reutilização

O uso crescente de recursos computacionais para solução de problemas criou uma forte demanda na área de projeto e análise de sistemas, com as seguintes características: alta qualidade, elevado desempenho, alta confiabilidade, baixo custo de desenvolvimento, mecanismos eficazes de desenvolvimento cooperativo, elevada modularidade e portabilidade, alta taxa de reutilização de componentes, interfaces externas adequadas ao usuário e interfaces de dados independentes de plataformas.

A reutilização de software vem sendo apresentada como um possível caminho para a redução dos custos de produção e tempo de desenvolvimento, e para a melhoria da qualidade e confiabilidade do software produzido. Dentre as tecnologias, atualmente, disponíveis para o suporte ao desenvolvimento, buscando a reutilização de software, temos: geradores de aplicação, linguagens de quarta geração, ambientes de suporte a bibliotecas de software, sistemas orientados a objetos, etc.

O termo *reutilização de software* caracteriza-se pela utilização de produtos de software desenvolvidos ao longo do ciclo de vida em uma situação diferente daquela para a qual foram originalmente produzidos. Neste sentido, considera-se não apenas a reutilização de produtos executáveis (módulos, rotinas, fragmentos de código, etc.) mas, também, produtos não executáveis (definição de requisitos, especificação, projeto, etc.).

A adoção de uma estratégia de reutilização no desenvolvimento de software pode trazer muitas vantagens, tais como:

- melhores índices de produtividade no desenvolvimento de software;
- produtos de melhor qualidade, mais confiáveis, consistentes e padronizados;

- redução dos custos e tempo envolvidos no desenvolvimento de software, e maior flexibilidade na estrutura do software produzido, facilitando assim, sua manutenção e/ou evolução.

2.1.3 Proposta de Cunha para um Ambiente Configurável

Cunha (1997) propõe que, para que se tenha uma melhoria sensível na qualidade dos sistemas de software produzidos, e na produtividade dos desenvolvedores, faz-se necessário adotar uma abordagem de qualidade e produtividade para o desenvolvimento de sistemas de software grandes e complexos que:

- utilize o paradigma de desenvolvimento de software orientado a objetos;
- estabeleça uma arquitetura para o sistema de software a ser desenvolvido, propiciando o máximo de reutilização;
- estabeleça um ambiente e processo de desenvolvimento, de acordo com a cultura da organização de desenvolvimento e voltado para a arquitetura do sistema de software estabelecido;
- controle, de forma integrada, os aspectos técnicos, gerenciais, organizacionais e humanos do desenvolvimento dos sistemas de software, utilizando a gestão pela qualidade total.

Sendo assim, Cunha começou a explorar a reutilização de código, evoluindo mais tarde para códigos configuráveis e, finalmente, para a adoção de uma arquitetura configurável de sistema de software orientado a objetos, denominada de SOFTBOARD (Cunha, 1997), que consiste em uma extensão da proposta de Coad e Yourdon, de um modelo multicomponentes (Coad e Yourdon, 1992; 1993).

A arquitetura SOFTBOARD consiste, na divisão do sistema de software em seis componentes, cada um com responsabilidade bem definida dentro do sistema, os quais consistem em: Componente Interface Humana (CIH), Componente Interface com outros Sistemas (CIS), Componente Gerenciador de Cenários (CGC), Componente Gerenciador de Dados (CGD), Componente Domínio do Problema (CDP) e Componente Gerenciador de Configuração de Sistema Orientado a objetos (CMOOS),

como mostra a Figura 2.1.

Na arquitetura configurável, o CDP pode ser trocado de tal forma, que os outros componentes se adequam a esta mudança. Desta forma, o sistema de software configurável seria similar a uma “motherboard”, onde se teria o CDP que seria um “chip” que poderia ser trocado, enquanto todos os outros componentes se adaptariam a este novo CDP (Cunha, 1997).

Esta SOFTBOARD é capaz de ser utilizada por apenas uma aplicação de cada vez. Para que cada componente se adapte, é preciso ficar registrado em algum lugar que, quando a aplicação for trocada, uma nova interface humana terá que ser mostrada, novos cenários serão gerenciados e novos dados serão mantidos. O responsável por disponibilizar estas informações de configuração é o CMOOS.

Como os componentes CIH, CIS, CGC e CGD podem ser configuráveis para atender o CDP de qualquer aplicação, evita-se a necessidade de se gerar um novo código para estes componentes, conseguindo-se desta forma, reutilização total dos mesmos.

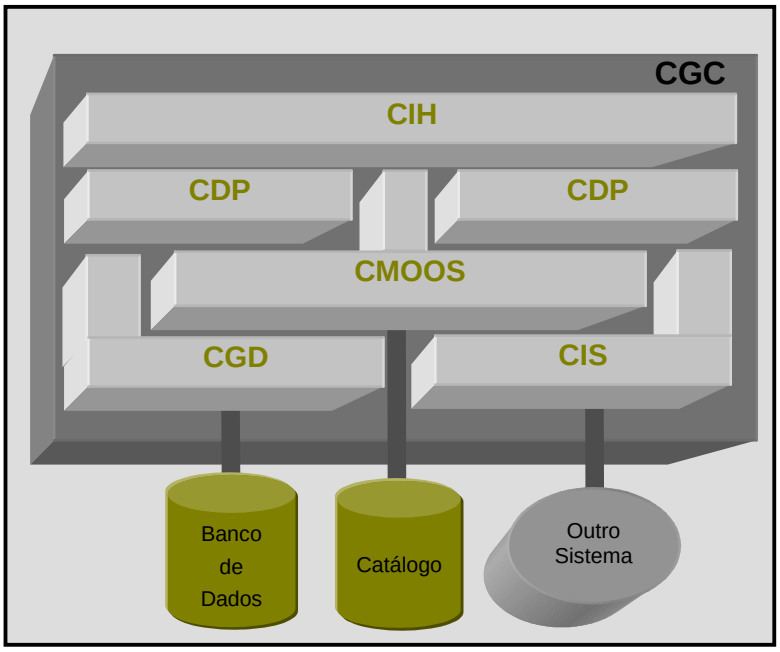


Fig. 2.1 - SOFTBOARD -Aperfeiçoamento da proposta de Coad e Yourdon, por Cunha. Adaptada de Cunha (1997).

2.1.3.1 Componente Domínio do Problema (CDP)

No CDP ficam os objetos da essência do sistema de software, aqueles que têm as responsabilidades fim da aplicação. Neste componente estão as características da aplicação.

No CDP estão descritas as principais classes-&-objetos[1] do sistema. Para cada uma delas estão descritos seus atributos, serviços/operações e conexões com outras classes-&-objetos.

2.1.3.2 Componente Interface Humana (CIH)

O CIH inclui os formatos de telas de vídeo e entradas necessárias para a efetiva interação humana com o computador. Nele ficam as classes-&-objetos que possibilitam a interação entre os objetos da essência do sistema de software e os usuários.

O CDP reutiliza as classes-&-objetos de um CIH para realizar sua interface com os usuários.

2.1.3.3 Componente de Interface entre Sistemas (CIS)

O CIS contém classes-&-objetos responsáveis pela comunicação entre os objetos do CDP de diferentes sistemas, no mesmo ou em diferentes processadores. As classes-&-objetos do CDP utilizam as classes e objetos do CIS para realizar esta comunicação.

2.1.3.4 Componente Gerenciador de Cenários (CGC)

O CGC compreende a definição, comunicação e coordenação (criação, suspensão, ativação, destruição, etc.) de cenários.

Um cenário é um conjunto de classes-&-objetos que expressam um comportamento para responder a um subconjunto de eventos, estímulos originados no ambiente externo da aplicação.

2.1.3.5 Componente Gerenciador de Dados Configurável (CGD)

O CGD fornece a infra-estrutura para o armazenamento e a recuperação de objetos, sendo a interface entre as classes-&-objetos persistentes do CDP e o banco de dados que

as armazena.

A Importância de um CGD configurável (Ferreira, 1996):

- Não há necessidade de implementar o código de acesso ao BD:

  Os objetos que estariam no CDP, não precisam se preocupar em desenvolver códigos para se salvar. Esta preocupação é passada para o CGD, que se torna então o responsável por resgatar o estado deste objeto e armazená-lo no banco de dados.

- Reutilização de Software:

  É o objetivo mais forte e o que justifica o projeto de um CGD configurável, pois ao se mudar a aplicação, basta apenas carregar outro ambiente para atender esta nova aplicação. Assim, não há mais necessidade de desenvolver uma nova interface com o banco de dados, portanto, tem-se um aumento de produtividade.

- Guardar o ambiente de um objeto:

  Isto também é um motivo forte para o projeto de um CGD configurável, pois mesmo os bancos de dados totalmente orientados a objetos não permitem este tipo de recurso. Ao se armazenar um objeto no banco de dados, suas conexões também seriam armazenadas. Ao se recuperar este objeto, o CGD acessa o catálogo através do CMOOS e disponibiliza estas conexões.

- Transparência de utilização de ambientes de armazenamento:

  O sistema de software irá trabalhar com objetos e o CGD irá então criar uma camada sobre o banco de dados, ficando transparente para o sistema como os objetos estão armazenados internamente.

Responsabilidades do CGD configurável:

O CGD configurável se comporta como uma camada entre o banco de dados e a aplicação e desta forma, o CGD tem as seguintes responsabilidades:

[1] Classe-&-objetos: termo utilizado para referir a uma classe e os objetos que a mesma contém.

- possibilitar mais de uma forma de gerenciar os objetos armazenados: arquivo simples, banco de dados relacional e banco de dados orientados a objetos;
- Definir uma interface com a aplicação (CDP), disponibilizando para a mesma os serviços para atender às solicitações comuns para acesso a um banco de dados, tais como: armazenar, alterar, excluir, etc., vindas do usuário, de um outro objeto, internamente do objeto, etc. Assim, retira-se a responsabilidade da aplicação de conter código para acessar a base de dados;

  O CGD disponibiliza uma Classe-&-objetos CDP-CGD que atende a esta responsabilidade, porque ela fornece às classes-&-objetos do domínio do problema, como herança, os serviços para atender à solicitação de armazenar, recuperar, excluir, dentre outras. Além disso, como todo objeto do CDP deve ter uma identificação única na base de dados, esta identificação, representada pelo atributo ID, é também fornecida pela Classe-&-objetos CDP-CGD.
- Definir uma interface com a aplicação, para buscar na memória o objeto que está sendo armazenado, ou seja, quando for solicitado o armazenamento de um objeto, o CGD terá que descobrir qual é a configuração do objeto (quais são seus atributos, os serviços de aceso dos atributos, as conexões, etc.), para depois buscá-lo na memória. O mesmo deve acontecer com a recuperação, em que se deve ter uma interface para colocar na memória o objeto que está sendo recuperado;
- Definir uma interface com o banco de dados, para armazenar os objetos da aplicação;
- Definir uma interface com o CMOOS, para recuperar a configuração do objeto a ser armazenado, recuperado ou excluído.

A Figura 2.2 representa uma modelagem das classes-&-objetos do CGD da SOFTBOARD.

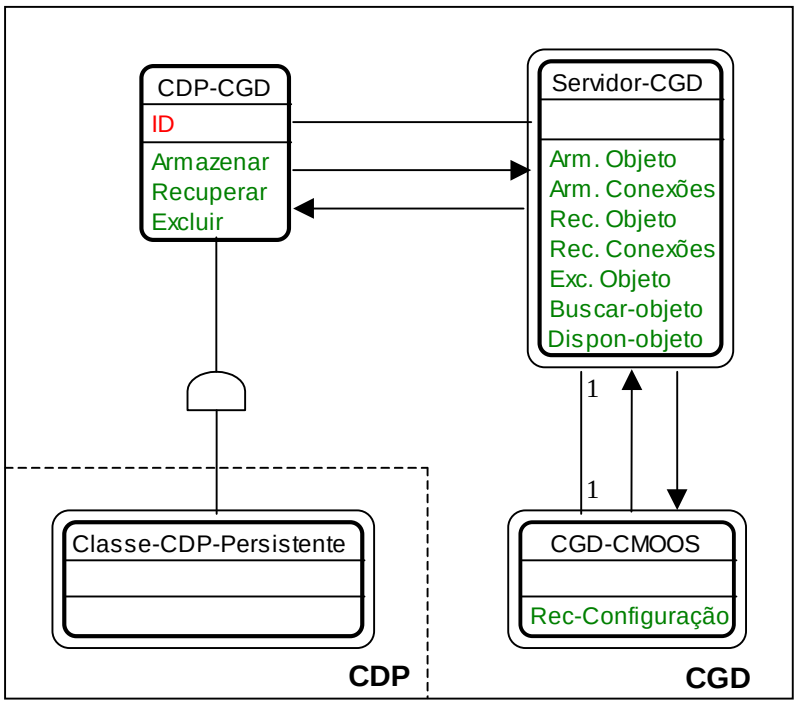


Fig. 2.2 - Uma modelagem das classes-&-objetos do CGD da SOFTBOARD.

FONTE: Cunha (1997).

2.1.3.6 Componente Gerenciador de Configuração de Sistema Orientado a Objetos (CMOOS)

O CMOOS constitui-se numas das principais contribuições de Cunha para que o sistema de software possa ser configurável.

O CMOOS faz a interface entre o CDP e os demais componentes do sistema, utilizando para isto o catálogo, um repositório de descritores (metadados), que consistem nas informações que possibilitam que o CIH, o CGC, o CIS e o CGD se adaptem, quando um CDP diferente é colocado no sistema de software.

Além das informações de configuração, o CMOOS possui uma classe-&-objeto SERVIDOR-CMOOS, que disponibiliza os serviços para montar, alterar em tempo de execução e recuperar as configurações de um dado objeto, seja ele do CDP, CGD, CIH e CIS (Cunha, 1997).

O objetivo de se ter um sistema de software configurável para várias aplicações, só tem

êxito se, na troca de uma aplicação por outra, todos os metadados da nova aplicação estiverem armazenados e mantidos no catálogo. Isto porque na carga para a execução da nova aplicação, os cenários que a compõem devem ser ativados, as interfaces, que esta aplicação mantém com o usuário, devem ser carregadas, e os dados permanentes, os objetos que esta aplicação manipula, devem estar disponíveis (Ferreira, 1996).

Neste trabalho, a preocupação será somente com os metadados que o CMOOS irá gerenciar, para permitir que seja feita a interface entre o CGD e o Banco de Dados, para diferentes aplicações.

2.1.3.7 Banco de Dados de Elementos Controlados (BDEC)

O catálogo, usado pelo CMOOS, é um subconjunto de informações mantidas no Banco de Dados de Elementos Controlados – BDEC.

O cadastro de todos os elementos (produtos, itens e relacionamentos) devem ser mantidos em um banco de dados, denominado de Banco de Dados de Elementos controlados – BDEC (cunha, 1997).

O BDEC, no desenvolvimento de software, baseado em SOFTBOARD, serve para dois propósitos:

- Manter informações para o controle de configuração e versões do sistema de software;
- Manter as informações necessárias, sobre a SOFTBOARD e a aplicação desenvolvida, para gerar o catálogo utilizado pelo CMOOS.

A equipe de controle de configuração e versões tem a responsabilidade de controlar o BDEC e gerar as aplicações que utilizam a SOFTBOARD e o respectivo catálogo para o CMOOS. Maiores detalhes sobre o esquema de controle e o BDEC são apresentados em Cunha (1997).

2.2 Banco de Dados

Um banco de dados pode ser definido como uma coleção de dados inter-relacionados, que são acessados por um ou mais programas de aplicação (Date, 1990).

Os sistemas de bancos de dados (um exemplo é apresentado na Figura 2.3) são projetados para administrar grandes volumes de informações. O gerenciamento de dados envolve a definição de estruturas para o armazenamento da informação, e a provisão de mecanismos para a manipulação destas informações.

O objetivo principal de um sistema de banco de dados é prover um ambiente que seja adequado e eficiente, para uso na recuperação e no armazenamento dos dados persistentes.

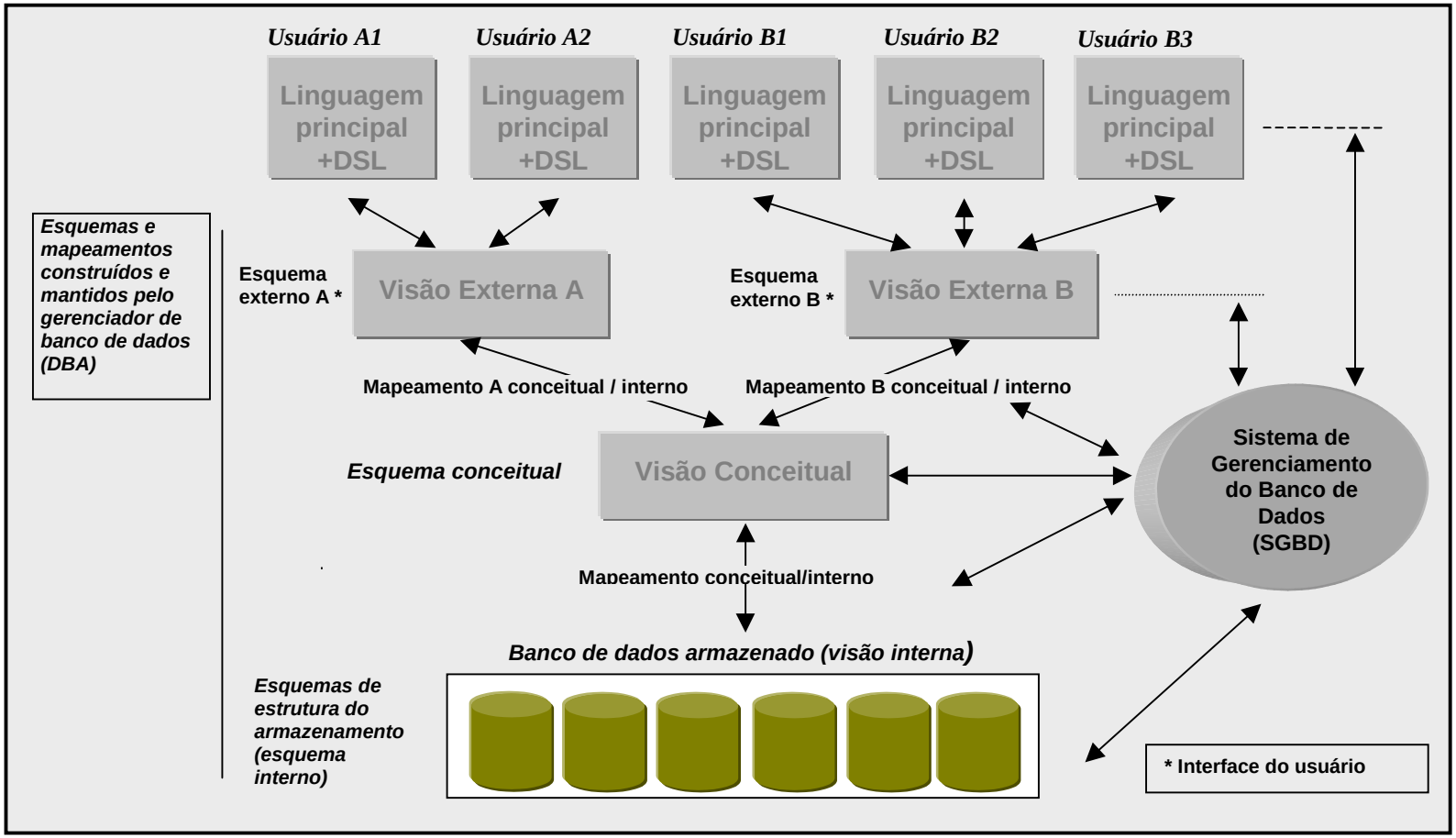


Fig. 2.3 – Arquitetura de um sistema de banco de dados.

FONTE: Date (1990).

2.2.1 Abstração de Informação e Dados

Abstração é a examinação seletiva de certos aspectos do problema. A meta da abstração é isolar aqueles aspectos que são importantes para alguns propósitos e suprimir os que não têm relevância. A abstração deverá sempre ter algum propósito, que determine o que importante ou não. São possíveis várias abstrações diferentes de uma mesma coisa, dependendo do propósito para o qual elas foram feitas.

Todas as palavras e linguagens humanas são abstrações – descrições do mundo real. O propósito de uma abstração é limitar o universo do que é possível de se fazer.

Na Figura 2.3, apresenta-se um esquema que contém os vários níveis de abstração, envolvidos em um possível processo de modelagem de sistema de software, baseado em SOFTBOARD, levando à criação de uma base de dados.

O nível mais alto é o mundo real que, do ponto de vista formal, é ainda muito nebuloso. Os objetos do mundo real são os seres, os fatos, as coisas e os organismos sociais. Fica a cargo do projetista da aplicação delimitar o que lhe interessa como mundo real, para fins de tratamento de informações (Setzer, 1995).

O segundo nível é o dos objetos informais e é caracterizado por relatórios escritos em uma linguagem natural. Nesse nível, denominado de Nível Descritivo, a descrição de um universo (ou de suas partes) deve ser totalmente inteligível para as pessoas que interagem normalmente com ele (ou partes dele), sem se exigir um conhecimento adicional ao que normalmente empregam nessa interação. Essa descrição deve organizada da melhor forma possível e constitui um modelo da realidade.

O terceiro nível é dos objetos formais em que o modelo desenvolvido deve ser estritamente formal. Esse modelo é baseado em símbolos para os quais deve haver uma conceituação rigorosa. Tal nível é denominado Nível Conceitual. Neste nível, aparecem dois aspectos distintos, em geral misturados nos modelos descritivos: tratam-se das estruturas de informações (metadados que descrevem as informações propriamente ditas) e das manipulações das informações (que se referem ao tratamento das informações), sendo conveniente definir-se uma linguagem formal para especificá-los.

Para a SOFTBOARD, os níveis descritos acima, fazem parte da modelagem da aplicação. Como resultado desta modelagem, tem-se o Componente Domínio do Problema (CDP). É dele que serão extraídas as informações necessárias à modelagem operacional do Banco de Dados. Informações mais detalhadas sobre a abstração do processo de modelagem do CDP podem ser encontrados no trabalho Estratégias e Padrões para o Desenvolvimento de Sistemas de Software Baseados na Arquitetura SOFTBOARD (Dias).

O quarto nível é o dos dados, que são os símbolos a serem inseridos no computador, tanto na descrição de estruturas (metadados) como aqueles que constituem os dados a serem propriamente processados pela máquina. Tal nível é denominado de Nível Operacional.

O quinto e último nível é denominado de Nível Físico, que consiste no nível mais baixo de abstração e descreve como os dados estão realmente armazenados. Neste nível, complexas estruturas de dados de baixo nível são descritas em detalhes.

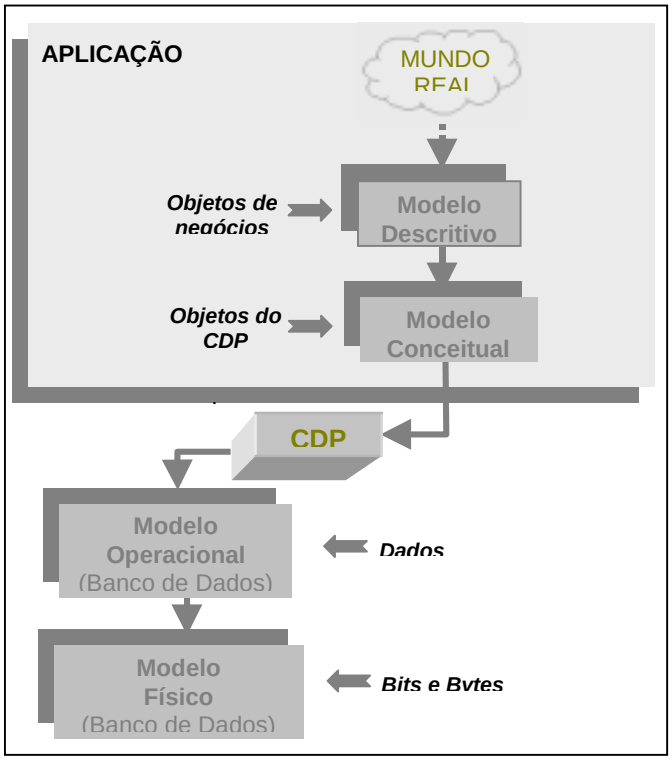


Fig. 2.4 - Níveis de abstração de um processo de desenvolvimento de sistemas de banco de dados baseado em SOFTBOARD.

2.2.2 Modelagem de Dados

Um modelo é uma representação abstrata e simplificada de um sistema real, com a qual pode-se explicar ou testar o seu comportamento, em seu todo ou em partes (Cougo, 1997).

A modelagem de dados tem sido, basicamente, aplicada como meio para obtenção de estruturas de dados que nos levem ao banco de dados.

Os modelos operacionais de dados podem ser definidos como uma coleção de dados, relacionamentos de dados, semântica de dados e restrições de consistência. Tal conceito é fundamental à estrutura de um banco de dados.

Os modelos físicos de dados captam aspectos da implementação de sistemas de bancos de dados e não serão abordados neste trabalho. Maiores detalhes sobre este nível, em Korth e Silberschatz (1994).

2.2.3 Banco de Dados Relacionais (BDR)

Este sistema de armazenamento de dados prevalece na maioria das instituições e tem resolvido muito bem o armazenamento de tipos primitivos de dados, como caracteres, inteiros, etc. Para os usuários do banco de dados relacional, as informações são armazenadas em forma de tabelas, que também são denominadas de relações, onde se têm linhas e colunas. Cada linha desta tabela é denominada de tupla e pode armazenar dados de um objeto, de tal forma que, nas colunas desta linha, estão os atributos deste objeto.

As principais características do Banco de Dados Relacional são (Date, 1990):

- Independência de Dados: a representação interna dos dados é independente da interface do aplicativo, ou seja, o banco de dados pode ser totalmente reorganizado a nível físico, sem a recompilação dos programas que o utiliza.
- Manipulação Declarativa: uma linguagem padrão internacional, a Linguagem Estruturada de Consulta (“Structured Query Language” - SQL) é utilizada para tratar o modelo relacional. A SQL é uma forma mais simples para manipular um banco de dados, do que o uso de uma linguagem de programação tradicional,

como Pascal ou C. Através dela o usuário pode, dentre outras coisas, criar tabelas, fazer consultas e armazenar informações em um BDR. A SQL foi uma linguagem criada, especificamente, para manipular os dados armazenados em um BDR. Ela se tornou um padrão em 1986, pelo Instituto Nacional Americano de Padrões (ANSI - “American National Standards Institute”).

- Simplicidade: o modelo relacional é considerado simples, porque é baseado em tabelas (relações) que é uma estrutura de dados simples. As pessoas programadoras ou não, normalmente, já estão familiarizadas com os conceitos básicos do modelo, que tem uma posição consolidada no mercado de banco de dados.

O modelo relacional é baseado numa coleção de relações. O usuário do banco de dados pode consultar essas relações, inserir novas tuplas, removê-las e atualizá-las.

2.2.3.1. Normalização

Normalização é um processo através do qual os dados são organizados no banco de dados relacional, de modo a reduzir e até eliminar a redundância dos mesmos.

Acima de tudo, a normalização propõe uma série de passos a serem seguidos para que tenhamos um processo de verificação, validação e ajuste das estruturas de dados que tenham sido observadas e modeladas, mas que ainda apresentem distorções.

A normalização pode ocorrer em três instantes distintos:

- durante o processo de concepção do modelo conceitual;
- durante a derivação do modelo operacional;
- após a derivação do modelo operacional.

Existe um grande número de formas normais, porém, neste trabalho, tratar-se-á somente da Forma Normal de Boyce Codd (“Boyce Codd Normal Form” - BCNF), pois ela engloba a primeira, segunda e terceira formas normais de Codd, tratando satisfatoriamente todas as relações usuais, que são representadas no modelo relacional.

Vale ressaltar que as várias formas de normalização e idéias associadas não são

consideradas partes do modelo relacional em si, mas constituem uma teoria separada, construída sobre este modelo (Date, 1990).

Forma Normal de Boyce-Codd

Segundo Korth e Silberschatz (1994), um banco de dados de boa qualidade é aquele que contém relações que satisfazem as restrições estabelecidas na BCNF.

Uma relação *R* está em BCNF, se somente se, todo determinante for chave candidata.

Neste contexto, um determinante consiste em uma coluna ou um conjunto de colunas *B* de *R*, que determina funcionalmente uma outra coluna *A* de *R*, ou seja, os valores de A de R, estarão sempre em função dos valores de *B* de *R*.

2.2.3.2 Restrições de Integridade no Modelo Operacional

O termo integridade é utilizado em contextos de bancos de dados com o significado de precisão, correção ou validade. O problema da integridade é o de assegurar que os dados no banco de dados sejam precisos – isto é, o problema de preservar o banco de dados contra atualizações inválidas. Tais imprecisões podem ser causadas por erros na entrada de dados, por erros da parte do operador e do programador da aplicação ou por falhas do sistema (Date, 1988).

Segundo Cougo (1997), as restrições no modelo operacional, podem ser tratadas sob três aspectos bastante distintos, que serão descritos a seguir.

Restrições de Domínio:

Dizem respeito às considerações de implementação a serem avaliadas para cada uma das colunas, em função dos possíveis valores assumidos em função de características do tipo de dado, e do próprio SGBD onde essa coluna venha a ser alocada.

O que determinará a restritividade (ou permissividade) na atribuição de valores para uma coluna será, em princípio, um dos itens a seguir:

- tipo de dado da coluna;
- tipo de representação interna escolhida para o dado;

- As características conceituais da presença ou não desse dado;
- A especificação explícita de intervalos para domínios contínuos;
- A especificação explícita de um conjunto de valores para domínios discretos;
- A definição de alguns tipos especiais de dados.

Restrições de Integridade:

Dizem respeito a questão tanto de implementação (característica do SGBD) como de negócio (de preservação de integridade das informações).

Em alguns casos, terá a função de estabelecer (implementar) no modelo operacional, algumas das restrições tecnológicas do modelo relacional e, outras vezes, simplesmente terá a função de manter a integridade entre os dados armazenados no banco de dados.

Dentre as restrições de integridade estão:

- As restrições do Modelo Operacional Relacional
    - Unicidade de valores dos identificadores das tuplas (chave primária);
    - Integridade Referencial: o valor dos elos explícitos entre tuplas de relações, que se relacionam, devem ser iguais.
- As restrições de negócio
    - Conectividade;
    - Valor mínimo e máximo para atributos, etc.

Restrições de Implementação:

Têm um caráter puramente restritivo de implementação sendo, portanto, dependente fortemente do SGBD escolhido.

Este tipo de restrição não será abordado neste trabalho. Maiores detalhes em Cougo (1997).

2.2.4 Banco de Dados Orientado a Objetos (BDOO)

Bancos de dados orientados a objeto têm sido caracterizados como a próxima geração de

sistemas gerenciadores de bancos de dados, para aplicações avançadas (McKeown e Saiedian, 1997).

Os BDOOs surgiram da necessidade de armazenamento persistente de objetos complexos, criados por linguagens de programação orientadas a objeto; da necessidade de novos tipos de dados, para o armazenamento de imagens ou grandes itens de texto, e da necessidade de definição de operações de aplicações específicas, não padronizadas.

A orientação a objetos oferece flexibilidade para manipular esses requerimentos sem se limitar pelos tipos de dados e linguagens de consulta de SGBDs tradicionais.

A principal característica dos bancos de dados orientados a objeto, consiste na habilidade de definir objetos complexos e as operações que podem ser aplicadas a esses objetos.

Os bancos de dados orientados a objeto integram o conceito da orientação a objeto com aptidões de banco de dados. Através de construções orientadas a objeto, os usuários podem esconder os detalhes da implementação de seus módulos, compartilhar a referência a objetos e expandir seus sistemas através de módulos existentes. A funcionalidade do banco de dados é necessária para assegurar o compartilhamento das informações por diferentes aplicações.

Através dos bancos de dados orientados a objetos, os usuários podem obter o estado em que os objetos se encontram e, estar atualizados entre as várias solicitações de programas. E ainda, vários usuários podem ao mesmo tempo compartilhar a mesma informação. Os bancos de dados orientados a objeto combinam os benefícios e conceitos da orientação a objeto com a funcionalidade dos bancos de dados (Koshafian, 1994).

Considerando os princípios da orientação a objetos (tipagem de dados, herança e identidade de objeto) e as aptidões dos bancos de dados (continuidade, concomitância, transações, recuperação, filtragem, atualização, integridade, segurança e desempenho), o potencial dos bancos de dados orientados a objeto reside na íntima integração destas duas tecnologias.

A vantagem de um banco de dados orientado a objetos é que, como todas as construções e relações são suportadas diretamente, praticamente nenhuma transformação é necessária para mapear um diagrama entidade-relacionamento ou um diagrama de projeto orientado a objetos, no esquema do banco de dados. As linguagens ou mecanismos de definição de dados, da maioria dos sistemas de gerenciamento de banco de dados orientado a objetos, fornecem construções que capturam diretamente as relações em modelos semânticos ou de relação de entidades. Assim, no que diz respeito à *definição* do banco de dados, pelo menos estruturalmente, bancos de dados orientados a objeto são muito ricos.

Uma de suas desvantagens é a falta de uma linguagem padrão para a manipulação dos dados. A maioria dos sistemas de banco de dados orientado a objeto fornece para este fim, linguagens de consulta sem um modelo rigoroso. Em muitos casos, essas linguagens de consulta são extensões SQL, orientadas a objeto.

2.2.5 Bancos de Dados Relacionais Estendidos

Sendo o banco de dados relacional largamente usado para aplicações tradicionais, alguns fornecedores de banco de dados relacionais estão incorporando extensões orientadas a objetos a seus produtos, que são denominados de bancos de dados estendidos ou *objeto-relacionais*. As extensões incluem gerenciamento de uma larga variedade de tipos de dados, armazenamento de relacionamentos complexos, dentre outras características do modelo de objetos. A cada produto são inseridas determinadas extensões, portanto a escolha do produto depende da aplicação para a qual ele será utilizado.

Os bancos de dados estendidos disponibilizam algumas capacidades de objetos, mas eles não oferecem ainda o mesmo nível de suporte dos bancos de dados orientados a objeto, para características como encapsulamento e herança. Uma questão a esse tipo de banco de dados que continua em aberto é se o cliente tem que escolher entre o suporte a dados complexos e a habilidade de usar outras características como processamento paralelo, replicação de dados, ou banco de dados distribuídos (Davis, 1998).

2.2.6 Evolução dos Bancos de Dados

Conclui-se então, que os anos 90 provavelmente serão conhecidos como a década que lançou a era da orientação a objetos na computação, ou pelo menos, como a década em que o paradigma ganhou mais força.

Para os bancos de dados orientados a objetos as coisas ainda não estão bem claras. Existem correntes que preferem estender seus produtos, os bancos de dados relacionais, para uma orientação a objetos, gerando novas versões da SQL, como a SQL2 e a SQL3, as quais serão abordadas posteriormente. Existem outras correntes que preferem construir um banco de dados orientado a objetos totalmente novo, inclusive com uma nova linguagem de consulta, a Linguagem de Consulta Estruturada a Orientada a Objetos (“Object Structured Query Language” - OSQL).

O fato é se está em um período de transição, de um lado as aplicações orientadas a objeto exigindo recursos de banco de dados com extensão a orientação a objetos, de outro lado ainda não estão definidas as padronizações para as linguagens com extensão a objetos. Esta é a razão pela qual muitos autores acreditam que os bancos de dados relacionais existirão por um longo tempo, e mesmo que definam as padronizações para os bancos de dados orientados a objetos, muitas bases instaladas estarão ainda no modelo relacional. E esta foi a razão da escolha do modelo relacional para o banco de dados a ser utilizado pela SOFTBOARD.

Linguagem de Consulta Estruturada (SQL)

Desde que a linguagem SQL é um padrão internacional para manipular dados em um banco de dados relacional, nada mais razoável do que estender essa linguagem, com construções orientadas a objetos. Esta abordagem está sendo perseguida por fabricantes de bancos de dados relacionais, evidentemente para preservar a base instalada e pensando no futuro. Os vendedores de BDRs, tais como Oracle, Ask, Informix e Borland, estão incorporando padrões de orientação a objeto em seus produtos. A SQL está sendo modificada, portanto, estão sendo geradas as versões SQL2 e a SQL3 (Roddick, 1992).

Linguagem de Consulta Estruturada 2 (SQL2)

Após o antigo padrão SQL ANSI/ISO tornar-se amplamente aceito, os esforços de padronização se concentraram em um aprimoramento compatível, chamado SQL2. Esse padrão foi confirmado em 1992 e inclui algumas alterações do tipo:

- Fornece um sistema de tipos mais rico, incluindo data (DATE), hora (TIME), registro de hora (TIMESTAMP) e intervalo (INTERVAL) para as transações que estão sendo feitas no Banco de Dados.
- Oferece melhor suporte à manipulação de "string" de caracteres;
- Permite diferentes níveis de isolamento. O isolamento é um conceito utilizado nos sistemas de gerenciamento de banco de dados, para poder permitir que as transações possam realmente ser isoladas da interferência de outras transações que estão sendo realizadas no banco de dados, fornecendo assim maior segurança.

As extensões da SQL2 são principalmente incrementais. Juntamente com o esforço de padronização, há um esforço para definir um padrão SQL3 que contenha recursos de banco de dados orientado a objetos (Melton e Simon, 1993).

Linguagem Estruturada de Consulta 3 (SQL3)

Em processo de padronização, cobre os seguintes requisitos:

- Tipificação de dados definidos pelo usuário*:*

  A maioria dos sistemas SQL vem com um grupo embutido de tipos de dados como: inteiros, números de pontos flutuantes, "strings" de caracteres, datas, registro de hora, etc. Para ser mais flexível e satisfazer às necessidades do usuário quanto à extensibilidade, uma tendência é valorizar esses tipos de dados e permitir tipos extras. Uma alternativa mais realista é permitir que os usuários desenvolvam seu próprio tipo de dado (Ferreira, 1996).

- Herança:

  A SQL3 inclui uma proposta para herança de tabelas.

2.3 Mapeamento Objeto-Relacional

O desenvolvimento de software orientado a objetos está rapidamente se tornando a principal abordagem para se construir sistemas flexíveis, em ambientes cliente/servidor. Adicionalmente, desde a década passada, a tecnologia relacional tem maturidade e vem sendo largamente adotada para se gerenciar dados corporativos. Estas duas tendências estão motivando a necessidade de construção de aplicações orientadas a objeto que acessam bancos de dados relacionais.

O mapeamento objeto-relacional é uma nova técnica, que ainda está em fase de estudos, utilizada para a interação de aplicações orientadas a objetos com bancos de dados relacionais (Agarwal et al., 1998; Brown e Whitenack, 1998; Rothwell, 1998). O mapeamento entre os dois modelos requer a decisão de como os dois mundos podem se referir um ao outro (Fussell, 1998).

Um dos segredos do sucesso do mapeamento objeto-relacional é o entendimento de ambos os paradigmas, e suas diferenças (Ambler, 1998).

2.3.1 Tecnologia de Orientação a Objetos x Banco de Dados Relacionais

Enquanto as linguagens de programação orientadas a objetos estão revolucionando o desenvolvimento de software desde 1990, os sistemas de bancos de dados relacionais revolucionaram os sistemas de gerenciamento de bancos de dados na década de 80.

A técnica orientada a objetos enfatiza a independência de objetos através de encapsulamento de dados e operações. Assim, objetos são independentes uns dos outros, comunicando-se entre si por mensagens. Bancos de dados tradicionais enfatizam a independência de dados, separando a abordagem em duas partes: dados e operações.

As principais características da tecnologia de orientação a objetos e dos bancos de dados relacionais, são apresentadas na Tabela 2.1.

TABELA 2.1 – TECNOLOGIA DE ORIENTAÇÃO A OBJETO X BANCOS DE DADOS RELACIONAIS

Tecnologia de Orientação a objetos	Bancos de dados relacionais
Boa para representar relacionamentos complexos entre objetos.	Bons para gerenciar grandes quantidades de dados .
Excelentes para manipulação dos dados, mas provêm pouco ou nenhum suporte para recuperação e persistência dos dados.	Bons para recuperação dos dados mas provêm pouco suporte para manipulação dos dados.
Objetos são independentes uns dos outros, comunicando-se entre si por mensagens.	Enfatizam a independência dos dados, separando o mundo em dados e operações.
O foco da modelagem de objetos está na identidade e no comportamento.	O foco da modelagem relacional é a informação.

2.3.2 Modelagem de Objetos e Modelagem Relacional

A modelagem de informações tem evoluído no decorrer dos anos. A ferramenta principal para a modelagem de informações, o diagrama de entidade-relacionamento, evoluiu para modelos semânticos de dados. Modelar o mundo em dados ajudou a capturar o conteúdo do domínio dos problemas (Coad e Yourdon, 1992).

Os *modelos de objetos* são diferentes de outras técnicas de modelagem porque eles têm agrupados o conceito de variáveis e tipos de dados abstratos, em um tipo variável abstrato: um objeto. Objetos têm identidade, estado e comportamento. Para facilitar a modelagem de objetos, existem conceitos de tipos, herança, associação e classe.

A *modelagem relacional* trabalha em termos de predicado, axiomas verdadeiros, e declarações verdadeiras deriváveis. Relações definem a possível declaração verdadeira; tuplas descrevem o conhecimento verdadeiro corrente; os valores da relação agrupam declarações verdadeiras; variáveis da relação lembram valores; e as expressões da relação podem derivar novos valores.

Felizmente, a modelagem objeto e a modelagem relacional possuem diferentes interesses, que são atualmente extremamente compatíveis.

2.3.3 Problemas do Uso de um Banco de Dados Relacional com Sistemas Orientados a Objetos:

Em um banco de dados relacional, a informação é armazenada em tabelas. Os tipos de dados a serem utilizados nas tabelas são, em sua maioria, tipos primitivos como inteiros e caracteres. Isto traz alguns problemas ao se querer armazenar os objetos. O primeiro deles é que somente dados podem ser armazenados e não comportamento. O segundo é que somente tipos de dados primitivos podem ser armazenados e não estruturas complexas, como objetos.

Quando uma linguagem de programação orientada a objeto é conectada a um SGBD Relacional surgem alguns problemas. O primeiro deles é que toda informação está armazenada em objetos. Precisa-se transformar os objetos em tabelas. Este problema é, às vezes, chamado de *problema de impedância*. Isso é bastante frequente, visto que os programas possuem um conjunto de tipos muito amplo, incluindo aqueles criados pelo usuário. Todos estes tipos precisam ser convertidos para os tipos existentes de Sistemas Gerenciadores de Banco de Dados Relacional.

O *problema de impedância* vem à tona quando observamos a abordagem referente ao acesso: com o paradigma de objeto percorre-se objetos via seus relacionamentos enquanto que com o paradigma relacional duplicam-se os dados para associar linhas em tabelas. Outro problema é o tratamento de herança de objetos no banco de dados. Como expressar no banco de dados a capacidade de um objeto poder ser herdado por outro.

2.4 Padrões e Estratégias

Padrões e estratégias expressam exemplos de boa prática, aqueles que podem ser usados para ajudar modeladores no desenvolvimento de resultados mais efetivos. Ambos ajudam desenvolvedores a ganhar um entendimento intuitivo, uma maior *sensibilidade* para a construção de modelos (Coad et al., 1995).

2.4.1 Padrões

Um dos últimos tópicos em emergência nas comunidades de orientação a objeto, consiste na abordagem de padrões para o desenvolvimento de softwares.

Uma das primeiras coisas que toda ciência tem que ter é um vocabulário para expressar seus conceitos e uma linguagem para relacioná-los juntos. O objetivo dos padrões com a comunidade de software é a criação de um corpo de literatura, para ajudar desenvolvedores de software a resolverem problemas de dificuldade comum, encontrados através da engenharia e desenvolvimento de software. Padrões ajudam a criar uma linguagem compartilhada para comunicação de percepção e experiência sobre estes problemas e suas soluções.

2.4.1.1 Origem dos Padrões

Padrões de software iniciaram a se tornar popular, com a larga aceitação do livro: “Design Patterns: Elements of Reusable Object Oriented Software” de Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides (frequentemente referidos como “The Gang of four” ou somente “GoF”). Atualmente, a comunidade de software está usando padrões largamente para desenvolver a arquitetura e projeto de software, e mais recentemente, processo e organização de desenvolvimento de software. Outros livros recentes, que têm ajudado a popularizar os padrões são: “Pattern-Oriented Software Architecture: A System of Patterns” de Frank Buschman, Regine Meunier, Hans Rohnert, Peter Sommerlad e Michael Stal (as vezes chamados de “Siemens Gang of Five” ou somente “GoV”); os livros “Pattern Languages of Program Design” e “Pattern Languages of Program Design 2” e o livro “Object Models: Strategies, Patterns, & Applications” de Peter Coad, David North e Mark Mayfield..

O uso corrente do termo *padrão* é derivado das descrições do arquiteto Christopher Alexander, que escreveu vários livros no tópico, onde o relaciona com planejamento e arquitetura de construção.

Alexander (1979) mostra que, embora toda construção seja única, cada qual pode ser criada, seguindo-se uma coleção de padrões geral, ou seja, um padrão é uma solução

geral para um problema ou questão comum, de onde uma solução específica pode ser derivada.

2.4.1.2 Definição de Padrões

A noção de padrões é especialmente significante para o uso do paradigma de orientação a objeto, pois eles representam um grande passo para o reuso. A busca por padrões tem como foco a identificação do comportamento comum e das iterações que transcendem objetos individuais (Booch, 1998).

Para Coplien (1998a), o termo *padrão*, como adotado por desenvolvedores de software contemporâneo, é tanto parte de um sistema quanto uma descrição de como se construir tal parte do sistema, que neste último caso, nesta pesquisa é denominado de estratégia. Padrões usualmente descrevem abstrações de software usadas por projetistas e programadores avançados, em seus softwares.

Um bom padrão deve ajudar, guiar ou instruir um profissional inexperiente através de um processo de solução de problema. Alexander criou uma forma efetiva de padrão, a qual provê a solução do problema com: uma definição do problema, um contexto, uma descrição de forças atuantes no problema, a solução do problema, uma explicação racional da solução e como ela soluciona as forças. Contudo, não há um único meio de se escrever padrões (Coplien, 1998b; Meszaros e Doble, 1998).

Pode-se ainda, unir padrões em uma linguagem de padrões, famílias de padrões relacionados, que juntos produzem um conjunto solução, para problemas de construção de sistemas em um domínio individual (Coplien, 1998b).

Segundo Martin (1998), padrões consistem em um meio para capturar bons princípios de projeto. Eles podem ser úteis para projetistas que procuram soluções potenciais para problemas particulares. Eles consistem em um meio efetivo para comunicar soluções para problemas.

2.4.1.3 Importância dos Padrões

Há algum tempo, padrões têm sido vistos somente como parte importante de como as pessoas entendem e modelam o mundo ao seu redor. Auxiliando projetistas de software

por toda parte, a indústria tem reconhecido, recentemente, o valor dos padrões também no projeto de softwares (Coplien, 1998b).

O uso de padrões para projeto não é muito similar a nenhum outro método. Padrões de projeto provêm redução de complexidade, através do fornecimento de um vocabulário ao processo de projeto, assim como auxiliando projetistas, fornecendo elementos apropriados para um determinado contexto (Palepu, 1998).

Tanto padrões como linguagens de padrões auxiliam os desenvolvedores a comunicarem um maior conhecimento sobre determinada arquitetura. Auxiliam ainda, pessoas a aprenderem um novo paradigma de projeto ou estilo de arquitetura. Eles ajudam principalmente, novos desenvolvedores a ignorarem armadilhas e ciladas que têm, tradicionalmente, sido aprendidas somente através de muita experiência.

Contudo, é muito importante que se tenha qualidade na descrição de padrões, pois a utilidade de um padrão é proporcional a sua percepção por seus usuários (Meszaros e Doble, 1998).

2.4.2 Estratégias

As estratégias têm como maior finalidade guiar o modelador no decorrer do desenvolvimento do projeto, indicando *o quê*, *como* e *quando* executar determinadas ações.

2.4.2.1 Definição de Estratégias

Uma estratégia consiste em um conjunto de ações, destinado a alcançar determinado objetivo (Coad et al., 1995).

Existem duas fases distintas associadas com o uso bem sucedido de estratégias: o descobrimento e as generalizações de estratégias. A primeira, se refere ao uso inicial de estratégias e a segunda, em todos os limites de situações em que uma estratégia pode ser aplicável.

Estratégias podem ser desenvolvidas por várias razões. A mais óbvia é uma falha de abordagens existentes. Contudo, elegância, inovação e eficiência, também motivam a descoberta de novas estratégias (Siegler e Jenkins, 1989).

2.4.2.2 Importância das Estratégias

As estratégias consistem em um importante meio, através do qual a realização de uma tarefa se torna mais eficiente e confiável, pois elas fornecem um conjunto de ações que deve ser usado como guia para a realização de determinada tarefa.

A questão da eficiência sempre surge, desde que haja mais de um modo para se realizar uma tarefa e, particularmente, quando se tem uma tarefa complexa. O conhecimento destes modos alternativos e da maneira como escolher dentre eles, pode ser referido como conhecimento estratégico.

O conhecimento estratégico tem sido estudado em uma variedade de domínios, como aprendizado e solução de problemas. Por exemplo, Anderson (1995) descreve o aprendizado de estratégia, como o melhoramento que sucede quando pessoas aprendem o meio ótimo de organizar a solução de seus problemas, para um domínio particular.

CAPITULO 3

MODELAGEM DO BANCO DE DADOS PARA A SOFTBOARD

Este capítulo irá tratar da descrição das estratégias e dos padrões para a modelagem do banco de dados relacional a ser utilizado pelo sistema de software, baseado em SOFTBOARD, conforme o objetivo 1 desta pesquisa, apresentado na Figura 3.1.

O Apêndice E apresenta a aplicação e validação das estratégias e dos padrões descritos neste capítulo.

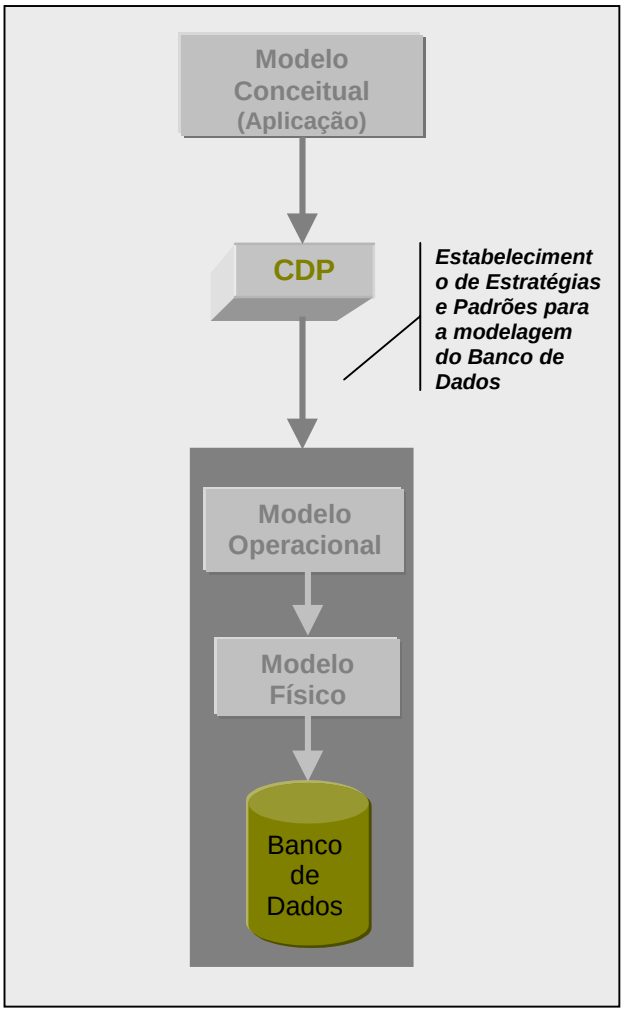


Fig. 3.1 - Identificação do objetivo 1: Estratégias e padrões para modelagem do banco de dados para a SOFTBOARD.

As estratégias elaboradas, para a modelagem do banco de dados relacional de uma aplicação baseada em SOFTBOARD, estão divididas em quatro categorias:

- Estratégias de identificação;
- Estratégias de representação;
- Estratégia para normalização do modelo operacional;

As estratégias são apresentadas da seguinte forma:

- um identificador da estratégia: uma letra de identificação e uma numeração (por exemplo: #M1, #M2 – onde M identifica que é uma estratégia de modelagem);
- um título: descreve o propósito da estratégia;
- um tópico de descrição: representa a estratégia propriamente dita, apresentando uma declaração textual dos passos a serem seguidos para se atingir um objetivo específico;
- um tópico de comentário: quando necessário, apresenta algumas informações adicionais referentes as estratégias e;
- um tópico de estratégia para cadastro no BDEC: são apresentadas as estratégias que devem ser utilizadas para a alimentação do BDEC, as quais estão descritas no capítulo 4 deste trabalho de pesquisa.

3.1 Estratégias de Identificação

É necessário que sejam designados identificadores únicos aos objetos para que se possa identificá-los, permitindo que um objeto seja armazenado e recuperado de forma única no banco de dados.

Na terminologia relacional, um identificador único é denominado de chave e na terminologia de objeto, ele é denominado de identificador de objeto (“Object identifier” - OID), que será tratado, neste trabalho, também como identificador (ID).

Na SOFTBOARD, como citado no capítulo anterior no item 2.1.3.5, o Componente Gerenciador de dados é o componente responsável pela identificação dos objetos. Ele

possui uma Classe-&-Objetos denominada de “CDP-CGD”, que fornece, através de herança, um identificador (ID) para todo objeto persistente do Componente Domínio do Problema (Cunha, 1997).

- ✓ Primeiramente, devem ser utilizadas as estratégias de identificação, que consistem em: identificação dos objetos persistentes, identificação dos atributos persistente e identificação dos padrões em que os objetos do Componente Domínio do Problema se apresentam.

Estratégia #M1 // Identificar as Classes-&-Objetos Persistentes

Descrição:

Selecionar as classes-&-objetos do Domínio do Problema da aplicação, que necessitam de se armazenarem no banco de dados.

Estratégia para cadastro no BDEC: #A2.

Estratégia #M2 // Identificar Atributos Persistentes

Descrição:

Selecionar os atributos de cada classe-&-objetos, que necessitam de se armazenarem no banco de dados.

Os atributos selecionados devem ser classificados em: atributos simples monovalorados ou multivalorados e atributos compostos monovalorados ou multivalorados.

Comentário:

Um atributo pode conter valores primitivos ou complexos, que deverão ser mapeados para relações.

Devem ser considerados como atributos com valores primitivos: atributos simples monovalorados.

Devem ser considerados como atributos compostos: atributos simples multivalorados e atributos compostos monovalorados ou multivalorados.

Estratégia para cadastro no BDEC: #A2.

Estratégia #M3 // Identificar os Padrões em que os Objetos do Componente Domínio do Problema se Apresentam

Descrição:

Identificar os objetos do CDP do sistema e suas associações. Em seguida, devem ser identificados os padrões em que eles se encontram, de acordo com as estruturas que estes objetos possuem.

Identificados os padrões, as classes-&-objetos do CDP deverão ser mapeadas para o modelo relacional de acordo com as estratégias escolhidas ou determinadas, relacionadas a cada padrão identificado.

Comentários:

Elementos participantes dos padrões de transação: ator, participante, transação, lugar, item específico, item de linha de transação, transação subsequente, item de linha de transação subsequente, item, item de linha, generalizações/especializações.

Elementos participantes dos padrões de agregação: recipiente, conteúdo, item de linha de recipiente, grupo, membro, montagem, parte, parte composta, pacote, componente do pacote.

Padrões de Transação: #P1 // Ator-Participante, #P2 // Participante-Transação, #P3 // Lugar-Transação, #P4 // Item Específico-Transação, #P5 // Transação-Item de Linha de Transação, #P6 // Transação-Transação Subsequente, #P7 // Item de Linha de Transação-Item de Linha de Transação Subsequente, #P8 // Item-Item de Linha, #P9 // Item Específico-Item de Linha, #P10 // Item-Item Específico, #P11 // Generalização-Especialização.

Padrões de Agregação: #P12 // Recipiente-Conteúdo, #P13 // Grupo-Membro, #P14 // Montagem-Parte.

3.2 Padrões

Pode-se observar que o Mundo Real é formado por um conjunto de “objetos”, os quais podem ser classificados em:

- atores: pessoas e organizações;
- participantes: correspondem aos diferentes papéis executados pelos atores. Por exemplo, em um Sistema de Biblioteca, uma pessoa pode executar os papéis de funcionário ou usuário da biblioteca;
- lugares: locais onde outros "objetos" se situam. Por exemplo, a própria biblioteca pode ser considerada como um lugar que contém livros e periódicos;
- coisas tangíveis: "objetos" concretos pertencentes ao Mundo Real. Por exemplo, os exemplares de livros e periódicos da biblioteca;
- coisas descritivas: "objetos" abstratos pertencentes ao Mundo Real. Por exemplo, o acervo da biblioteca, representado por seus livros e periódicos;
- transações: representam fatos do Mundo Real que consistem nos eventos a serem lembrados. Por exemplo, empréstimo, devolução e reserva de um livro.

Ao se observar a interação entre esses "objetos", pode-se notar o surgimento de diversos padrões de relacionamento, os quais têm o intuito de simplificar o entendimento, e a reprresentação do Mundo Real. Estes padrões podem ser geralmente divididos em padrões de transação e de agregação.

Nos padrões de transação, os relacionamentos ocorrem todos em torno de um "objeto transação". Por exemplo, toda *transação* acontece em um *lugar/organização* específico, sendo geralmente associada a um *participante* (por exemplo, um *usuário da biblioteca* associa-se a um *empréstimo* de um livro) ou associada a um *item específico* (uma *coisa tangível* associada a uma *transação*. Por exemplo, um carro (*item específico*) associa-se a uma colisão (*transação*) à qual ele se envolve). Cada *transação* contém um ou mais *itens de transação*, que representam as coisas (tangíveis ou descritivas) envolvidas na transação (por exemplo, os *produtos* (coisa tangível) envolvidos em uma *venda* são representados por um *item de venda* (*item de transação*), o qual está associado a *venda*). Pode acontecer também que uma transação esteja associada a uma outra transação – uma *transação subseqüente* (por exemplo, após uma transação *venda*, uma transação de *pagamento* é registrada) – e essa nova transação pode conter um ou mais *itens de*

transação subseqüente.

No Mundo Real, existem ainda, “objetos” que se subdividem em categorias com características parcialmente distintas. É o caso de em uma organização um *funcionário* poder ser classificado por categorias funcionais, tais como: *motoristas, secretárias* e *engenheiros*. Neste caso, existem características em comum entre esses “objetos”, tais como: nome, RG, CPF e endereço. No entanto, existem características próprias de cada categoria. Assim, não interessa para a organização o fato de uma secretária ter ou não carteira de habilitação para dirigir veículos, mas para os motoristas é necessário ter conhecimento sobre o número dessa carteira; por outro lado, é essencial saber-se quais são as línguas estrangeiras dominadas pelas secretárias, que cursos forma feitos pelos engenheiro etc. Isto é representado através do padrão de generalização-epecialização.

Os padrões de agregação surgem do princípio de que, no Mundo Real, existem “objetos” que são agregadores de outros objetos. Através deste padrão, são representados os “objetos recipiente” que possuem “objetos conteúdo” (por exemplo, o recipiente *biblioteca* e os seus conteúdos *acervos*), os “objetos grupo” formados por um conjunto de “objetos membro” (por exemplo, o grupo *biblioteca* e os seus membros *usuários*) e os “objetos montagem”, compostos por seus “objetos parte” (por exemplo, uma montagem *avião* e suas partes *asas*, *motores*, etc.).

Partindo-se do princípio de que os “objetos” que compõem o Mundo Real possuem padrões de relacionamento, Coad et al. (1995) apresentaram vários padrões exibindo a interação entre as diversas classes de objetos que constituem uma aplicação. Essas representações consistem em abstrações dos vários tipos de relacionamento entre os diversos “objetos” que fazem parte do Mundo Real.

Os padrões para a modelagem do banco de dados para a SOFTBOARD, que serão apresentados a seguir, foram gerados tendo como base os padrões apresentados em Coad el al. (1995).

3.2.1 Padrões de Transação

Os padrões de transação ocorrem todos em torno de um objeto transação do Mundo

Real, conforme ilustra a Figura 3.2, utilizando-se a notação da UML.

A notação simplificada da UML utilizada para o digrama de classes é apresentada no Apêndice B.

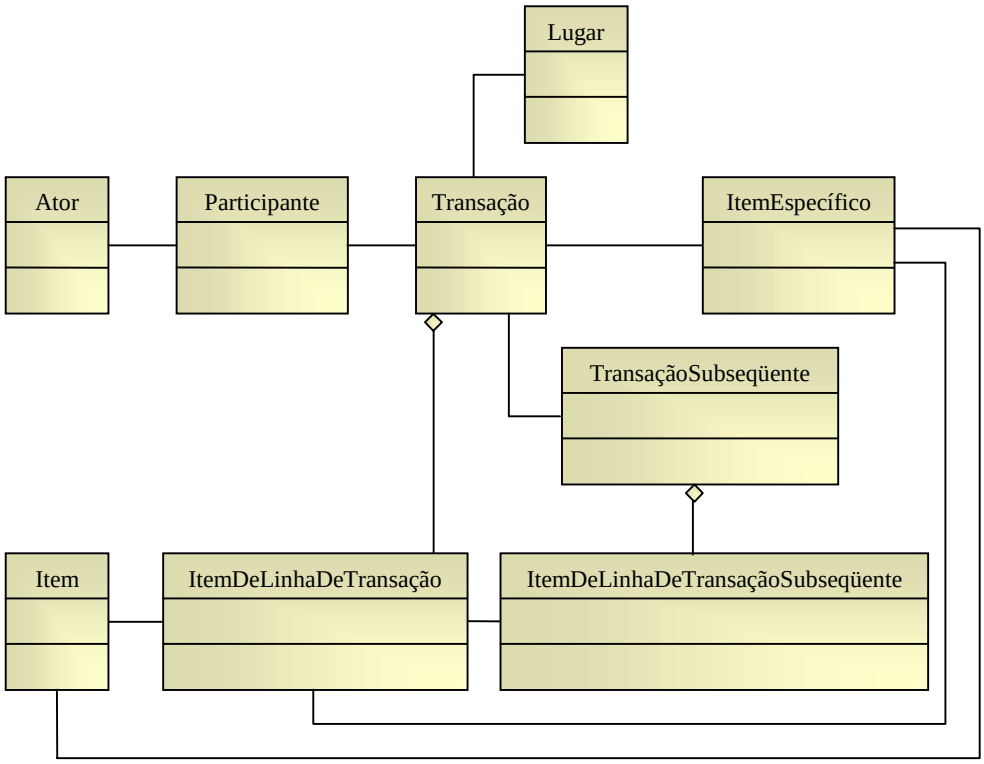


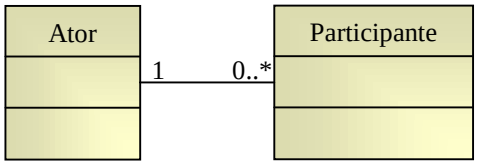
Fig. 3.2 – Modelo de padrões de transação.

Fonte: Coad et. al. (1995).

Padrão #1 // Ator-Participante

Este padrão representa a associação entre atores (que consistem nas pessoas e organizações) e participantes (papéis específicos executados pelos atores) pertencentes ao Mundo Real. Desta maneira, um participante sempre será um ator, e um ator poderá ser um determinado participante. Este padrão é utilizado, ao invés de se utilizar o padrão de generalização-especialização, para representar este relacionamento, pelo fato de um ator poder executar mais de um papel específico; ou seja, ser mais de um participante (por exemplo, em um Sistema Hospitalar, uma mesma pessoa pode executar os papéis de médico e paciente dentro do Sistema.).

Modelo:



Tipos de associações possíveis:

Associações 1:1 ou N:1. Não há casos de associações N:N.

Exemplos:

Ator: pessoas, organizações (agência, corporação, organização).

Participante: agente, comprador, caixa, cliente, distribuidor, trabalhador, investidor, estudante, professor.

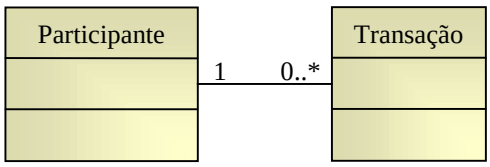
Estratégias de representação relacionadas:

#M10 // Avaliação da abordagem de representação de associação, #M11 // Representação de associações 1:1, #M12 // Representação de associações N:1.

Padrão #2 // Participante-Transação

Toda transação (evento a ser lembrado) que ocorre no Mundo Real está associada a um participante ou a um item específico (coisa tangível responsável pela transação). Este padrão representa a associação entre uma transação e o seu participante.

Modelo:



Tipos de associações possíveis:

Associações 1:1 ou N:1. Não há casos de associações N:N.

Exemplos:

Participante: agente, comprador, caixa, cliente, distribuidor, trabalhador, investidor, estudante, professor.

Transação: autorização, contrato, compra, depósito, pagamento.

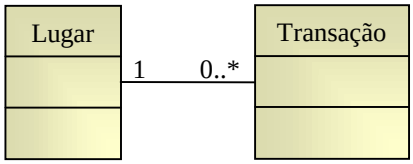
Estratégias de representação relacionadas:

#M10 // Avaliação da abordagem de representação de associação, #M11 // Representação de associações 1:1, #M12 // Representação de associações N:1.

Padrão #3 // Lugar-Transação

Toda transação que acontece no Mundo Real ocorre em um lugar específico. Este padrão representa a associação entre uma transação e o lugar onde esta transação ocorre.

Modelo:



Tipos de associações possíveis:

Associações 1:1 ou N:1. Não há casos de associações N:N.

Exemplos:

Lugar: aeroporto, banco, clínica, hospital, loja, região, entidade geográfica.

Transação: autorização, contrato, compra, depósito, pagamento.

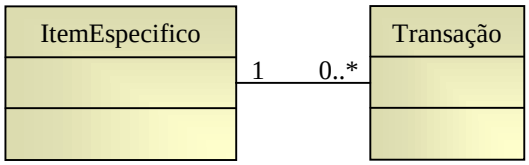
Estratégias de representação relacionadas:

#M10 // Avaliação da abordagem de representação de associação, #M11 // Representação de associações 1:1, #M12 // Representação de associações N:1.

Padrão #4 // Item Específico-Transação

Conforme foi descrito pelo Padrão #2, toda transação que acontece no Mundo Real está associada a um participante ou a um item específico (coisa tangível responsável pela transação). Este padrão representa a associação entre uma transação e um item específico responsável pela transação.

Modelo:



Tipos de associações possíveis:

Associações 1:1 ou N:1. Não há casos de associações N:N.

Exemplos:

Item Específico: veículo específico, registradora específica, barco específico.

Transação: autorização, contrato, compra, depósito, pagamento.

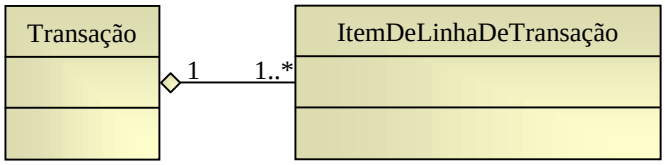
Estratégias de representação relacionadas:

#M10 // Avaliação da abordagem de representação de associação, #M11 // Representação de associações 1:1, #M12 // Representação de associações N:1.

Padrão #5 // Transação-Item de Linha de Transação

Toda transação que ocorre no Mundo Real possui um ou mais itens de transação, que são as coisas descritivas que representam as coisas tangíveis relacionadas a esta transação. Este padrão representa a conexão entre esses dois "objetos" – item de linha de transação e transação.

Modelo:



Tipos de associações possíveis:

Associações 1:1 ou N:1. Não há casos de associações N:N.

Exemplos:

Transação – Item de Linha de Transação: depósito- item de linha de depósito, pagamento-item de linha de pagamento, venda-item de linha de venda.

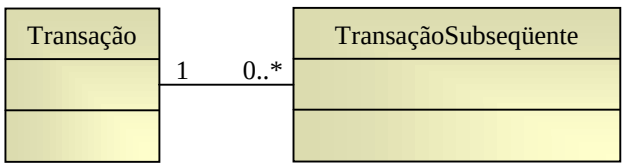
Estratégias de representação relacionadas:

#M13 // Representação de associações todo-parte.

Padrão #6 // Transação-Transação Subsequente

No Mundo Real, existem casos em que uma transação se associa a uma outra transação – uma transação subseqüente. Este padrão representa a ocorrência deste fato.

Modelo:



Tipos de associações possíveis:

Associações 1:1 ou N:1. Não há casos de associações N:N.

Exemplos:

Transação - Transação Subsequente: resultado intermediário-resultado final, empréstimo-devolução.

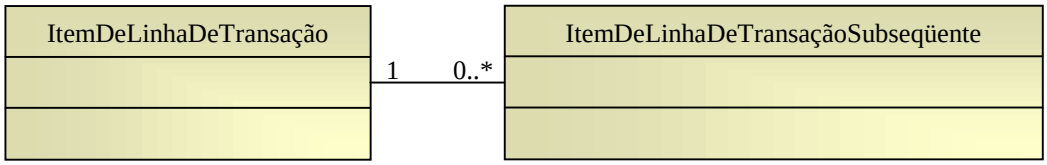
Estratégias de representação relacionadas:

#M10 // Avaliação da abordagem de representação de associação, #M11 // Representação de associações 1:1, #M12 // Representação de associações N:1.

Padrão #7 // Item de Linha Transação-Item de Linha de Transação Subsequente

Conforme foi descrito pelo *Padrão #6*, no Mundo Real pode acontecer que uma transação esteja associada a uma outra transação. A cada uma dessas transações existem itens relacionados, que são as coisas descritivas envolvidas nessas transações. Este padrão representa a associação entre esses "objetos itens de linha de transação".

Modelo:



Tipos de associações possíveis:

Associações 1:1 ou N:1. Não há casos de associações N:N.

Exemplos:

Item de Linha de Transação - Item de Linha de Transação Subsequente: item empréstimo-exemplar, item reserva-acervo.

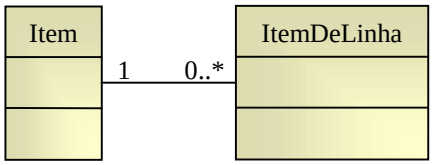
Estratégias de representação relacionadas:

#M10 // Avaliação da abordagem de representação de associação, #M11 // Representação de associações 1:1, #M12 // Representação de associações N:1.

Padrão #8 // Item-Item de Linha

No Mundo Real, a uma dada transação existem vários "objetos" associados, que são as coisas (descritivas ou tangíveis) envolvidas nesta transação. Esses "objetos" podem ser chamados de itens de transação, os quais descrevem os itens (coisas tangíveis). Este padrão representa o relacionamento entre os objetos itens e os objetos que os descrevem na transação, os itens de linha.

Modelo:



Tipos de associações possíveis:

Associações 1:1 ou N:1. Não há casos de associações N:N.

Exemplos:

Item – Item de Linha: item-item de linha de pagamento, item-item de linha de venda.

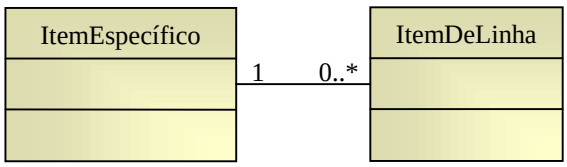
Estratégias de representação relacionadas:

#M10 // Avaliação da abordagem de representação de associação, #M11 // Representação de associações 1:1, #M12 // Representação de associações N:1.

Padrão #9 // Item Específico-Item de Linha

Os itens de transação são utilizados para representar itens específicos que estão associados a uma determinada transação. Este padrão representa o relacionamento entre esses dois tipos de objetos.

Modelo:



Tipos de associações possíveis:

Associações 1:1 ou N:1. Não há casos de associações N:N.

Exemplos:

Item Específico – Item de Linha: veículo específico-item de linha, registradora específica-item de linha.

Estratégias de representação relacionadas:

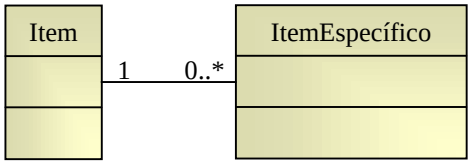
#M10 // Avaliação da abordagem de representação de associação, #M11 // Representação de associações 1:1, #M12 // Representação de associações N:1.

Padrão #10 // Item-Item Específico

No Mundo Real, existem “objetos” de conotação mais geral que podem ser

representados por “objetos” que possuem uma conotação mais específica. É o caso de um exemplar, em uma biblioteca, representar um item específico do acervo da Biblioteca. Este padrão representa a associação entre os objetos de conotação mais geral (itens) e objetos que os descrevem especificamente (itens específicos).

Modelo:



Tipos de associações possíveis:

Associações 1:1 ou N:1. Não há casos de associações N:N.

Exemplos:

Item - Item Específico: acervo-exemplar, avião-avião específico.

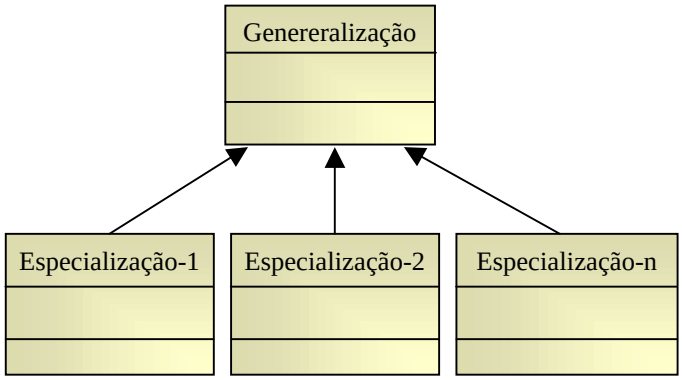
Estratégias de representação relacionadas:

#M10 // Avaliação da abordagem de representação de associação, #M11 // Representação de associações 1:1, #M12 // Representação de associações N:1.

Padrão #11 // Generalização-Especialização

O padrão de Generalização-Especialização surge devido à existência no Mundo Real de “objetos” que se subdividem em categorias com características parcialmente distintas. Este padrão exibe explicitamente as diferenças e as semelhanças entre essas categorias de “objetos”.

Modelo:



Exemplos:

Generalização/Especialização: departamento (pessoal / financeiro), veículo (civil / militar).

Estratégias de representação relacionadas:

#M7 // Representação de herança usando uma relação para uma hierarquia inteira de classe-&-objetos, #M8 // Representação de herança usando uma relação por classe-&-objetos concreta da hierarquia, #M9 // Representação de herança usando uma relação por classe-&-objetos da hierarquia.

3.2.2 Padrões de Agregação

Os padrões de agregação surgem do fato de que, no Mundo Real, existem "objetos agregadores" que são formados por diversos "objetos agregados", conforme ilustra a Figura 3.3, utilizando-se a notação da UML.

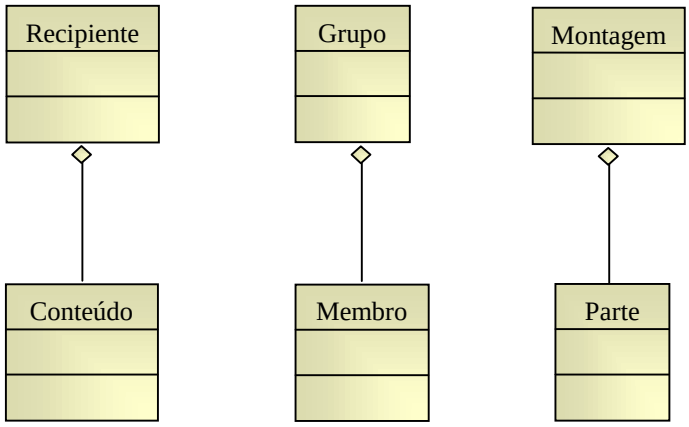


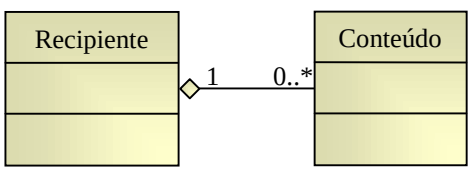
Fig. 3.3 – Modelo de padrões de agregação.

FONTE: Coad et al. (1995).

Padrão #12 // Recipiente-Conteúdo

Este padrão representa os "objetos recipiente" que contêm vários outros objetos, os "objetos conteúdo".

Modelo:



Tipos de associações possíveis:

Associações 1:1 ou N:1. Não há casos de associações N:N.

Exemplos:

Recipiente – Conteúdo: avião-carga, avião-passageiro, loja-item, catálogo-item do catálogo.

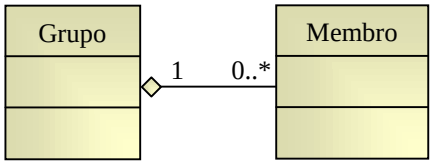
Estratégias de representação relacionadas:

#M13 // Representação de associações todo-parte.

Padrão #13 // Grupo-Membro

Este padrão representa a formação de um "objeto grupo" do Mundo Real, composto por seus vários membros.

Modelo:



Tipos de associações possíveis:

Associações 1:1, N:1 ou N:N.

Exemplos:

Grupo – Membro: Companhia-empregado; time-membro do time.

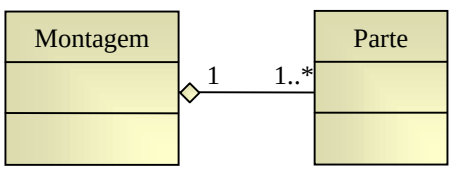
Estratégias de representação relacionadas:

#M13 // Representação de associações todo-parte. #M15 // Representação de associações N:N.

Padrão #14 // Montagem-Parte

Neste padrão, é representada a composição de "objetos montagem" do Mundo Real por suas diversas partes.

Modelo:



Tipos de associações possíveis:

Associações 1:1 ou N:1. Não há casos de associações N:N.

Exemplos:

Montagem - Parte: avião-motor, avião-parte do motor.

Estratégias de representação relacionadas:

#M13 // Representação de associações todo-parte.

3.3 Estratégias de Representação

- ✔ Estando identificados os padrões em que as classes-&-objetos do Componente Domínio do Problema se encontram, devem ser utilizadas as estratégias de representação, que consistem basicamente em: representação das classes-&-objetos persistentes em relações, representação dos atributos persistentes, representação do identificador de objeto, representação de herança e representação de associações.
- ✔ Para as classes-&-objetos que não se adequam a nenhum dos padrões estabelecidos, devem ser utilizadas as seguintes estratégias de representação: #M4, #M5 e #M6.
- ✔ Para as classes-&-objetos que se adequam aos padrões estabelecidos, deve-se escolher uma associação de cada vez e modelá-la, aplicando-se as estratégias de representação relacionadas ao padrão com que cada uma delas se identifica. Tais estratégias de representação são: estratégias de representação de herança (#M7 a #M9) e estratégias de representação de conexões (#M10 a #M15).
- ✔ A estratégia #M5 deve ser utilizada, com exceção a estratégia de representação dos identificadores dos objetos, juntamente com todas as demais estratégias de representação.

Estratégia #M4 // Representação das Classes-&-Objetos Persistentes em Relações

Descrição:

Deve-se criar uma relação para a representação de cada classe-&-objetos persistente.

Cada objeto da classe será representado por uma tupla na relação. Para isto, deve-se utilizar a estratégia #M6 de representação dos atributos persistentes em uma relação.

Comentários:

Esta estratégia deve ser utilizada somente para classes-&-objetos do sistema, que não se adequam aos padrões estabelecidos.

Estratégia para cadastro no BDEC: #A3, #A4, #A5, #A9, #A10.

Estratégia #M5 // Representação dos atributos persistentes

Descrição:

Cada atributo de tipo primitivo (caracter, "string", inteiro, "float", etc) será representado através de uma coluna da relação.

Para cada atributo multivalorado (simples ou composto), deverá ser criada uma nova relação, na qual ele deverá ser representado.

Cada atributo de tipo composto monovalorado poderá ser mapeado para a relação que representa a classe-&-objetos que ele pertence, através da divisão do atributo em várias colunas ou como único atributo. A decisão de como representá-lo, dependerá do modo de como as informações contidas no atributo forem utilizadas dentro do sistema em questão.

Cada atributo de tipo composto multivalorado deverá ser representado do mesmo modo que o composto monovalorado, porém na nova relação criada para representá-lo.

Dependendo do modo de como as informações contidas no atributo multivalorado forem utilizadas dentro do sistema em questão, a nova relação criada para representá-lo poderá ser representada de duas maneiras:

- deverá conter uma coluna de chave estrangeira, com a representação do identificador da classe-&-objetos de origem do atributo e outra coluna com o próprio atributo, os quais deverão compor a chave primária da relação;
- deverá conter o próprio atributo e uma identificação para ele, que deverá compor a chave primária da relação. Além disso, deverá ser criada uma outra relação, a qual

representará uma associação entre a relação que representa o atributo multivalorado e a relação que representa a sua classe-&-objetos de origem, sendo composta pelos atributos que representam as suas chaves primárias. Estes atributos deverão compor a chave primária da relação em questão.

Comentários:

Se houver restrição de unicidade do atributo multivalorado, a chave primária da nova relação criada para representá-lo, no caso em que a chave primária é representada pelo próprio atributo e uma identificação criada para ele, poderá também ser representada apenas pela coluna do próprio atributo.

Esta estratégia deve ser utilizada, com exceção a estratégia de representação dos identificadores dos objetos, juntamente com todas as demais estratégias de representação.

Estratégias para cadastro no BDEC: #A4, #A5 e #A9.

Estratégia #M6 // Representação do identificador de objeto

Descrição:

Todo objeto persistente tem um identificador, que deve ser mapeado para uma coluna de identificação da relação, a qual é denominada de chave primária da relação.

Comentários:

O valor de uma coluna ID deve ser único dentro da relação. Isto deve ser garantido, definindo-o sempre como chave primária da relação que representa sua classe-&-objetos de origem (Object Matter, 1998).

Estratégias para cadastro no BDEC: #A10.

3.3.1 Estratégias de representação de herança

Herança é um mecanismo para expressar a similaridade entre Classes, simplificando a definição de classes iguais a outras que já foram definidas. Ela representa generalização e especialização, tornando explícitos os atributos comuns em uma hierarquia de classes.

Uma hierarquia possui uma classe abstrata (classe ancestral da hierarquia) e classes-&-objetos concretas (classes-&-objetos que herdam as características da classe ancestral).

Existem basicamente três tipos de representações para herança, os quais serão descritos a seguir. A escolha do tipo de representação, que deverá ser usada, dependerá das necessidades da aplicação em relação ao banco de dados (Brown e Whitenack, 1998).

Os tipos de representações existentes e os fatores a considerar para a escolha do tipo ideal para uma aplicação em particular, são apresentados na Tabela 3.1.

TABELA 3.1 – FATORES A CONSIDERAR PARA A REPRESENTAÇÃO DE HERANÇA.

	Facilidade de implementação	**Facilidade de acesso aos dados**	**Junção**	**Velocidade de acesso aos dados**
Uma relação por hierarquia	Simples	Simples	Muito alto	Rápido
Uma relação por classe concreta	Médio	Simples	Alto	Rápido
Uma relação por classe	Difícil	Médio	Baixo	Médio/Rápido

Adaptada de Ambler (1998).

Para caracterizar os fatores *facilidade de implementação* e *facilidade de acesso aos dados* foram usados os parâmetros *simples*, *médio* e *difícil* para indicar o grau de facilidade de implementação.

Para caracterizar o fator *junção* foram usados os parâmetros *baixo*, *alto* e *muito alto* para indicar a quantidade de junções necessárias para o acesso a uma determinada informação.

Para caracterizar o fator *velocidade de acesso aos dados* foram usados os parâmetros *médio* e *rápido* para quantificar a velocidade em que os dados podem ser acessados.

A Figura 3.4 exemplifica os três tipos de representação de herança:

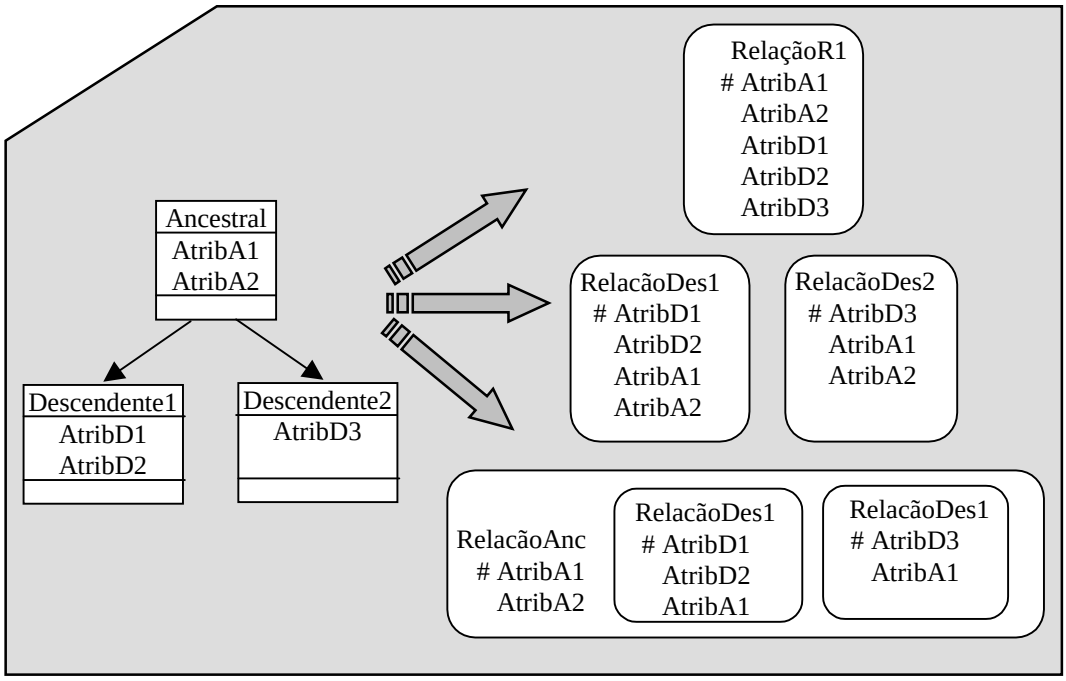


Fig. 3.4 – Tipos de representação de herança.

Estratégia #M7 // Representação de herança usando uma relação para uma hierarquia inteira de classe-&-objetos

Descrição:

Uma hierarquia de classes-&-objetos inteira deverá ser representada através da criação de uma única relação, onde se deve criar colunas para representar todos os atributos de todas as classes-&-objetos da hierarquia. A chave primária da relação deverá ser a representação do identificador da classe ancestral da hierarquia.

Comentários:

Dentre algumas vantagens desta abordagem, encontra-se a simplicidade de representação da hierarquia.

As desvantagens desta abordagem são as seguintes: toda vez que um novo atributo é adicionado em algum lugar na hierarquia das classes-&-objetos, uma nova coluna precisa ser adicionada na relação. Então, se um erro é cometido ao se adicionar um único atributo, todas as classes dentro da hierarquia podem ser afetadas e não somente as subclasses, para as quais os atributos se destinam (Ambler, 1998).

Estratégias para cadastro no BDEC: #A3, #A4, #A5, #A9, #A10.

Estratégia #M8 // Representação de herança usando uma relação por classe-&-objetos concreta da hierarquia

Descrição:

Deve-se criar uma relação para cada classe-&-objetos concreta da hierarquia. Cada relação deverá conter colunas que representam os próprios atributos de cada classe-&-objetos concreta, e colunas que representam os atributos herdados da classe ancestral da hierarquia. A chave primária de cada relação deverá ser a representação do próprio identificador das classes-&-objetos que elas representam.

Comentários:

A principal vantagem desta abordagem é que todo dado, que se necessita sobre uma classe-&-objetos individual, encontra-se armazenado somente em uma relação.

As desvantagens desta abordagem são as seguintes: quando se modifica uma classe ancestral da hierarquia, torna-se necessário modificar as relações que representam todas as suas subclasses (Ambler, 1998).

Estratégias para cadastro no BDEC: #A3, #A4, #A5 e #A6, #A9, A#10.

Estratégia #M9 // Representação de herança usando uma relação por classe-&-objetos da hierarquia

Descrição:

Deve-se criar uma relação para cada classe-&-objetos da hierarquia. A relação que representa a classe abstrata da hierarquia, deverá conter colunas somente com seus próprios atributos, enquanto que relações que representam as classes-&-objetos concretas deverão conter, além de colunas representantes de seus próprios atributos, uma coluna de chave estrangeira com a representação do identificador herdado da classe ancestral. A chave primária de cada relação deverá ser a representação do próprio identificador das classes-&-objetos que elas representam.

Comentários:

A principal vantagem desta abordagem é que ela se adequa melhor aos conceitos de orientação a objetos. Há também muita facilidade para a modificação de superclasses, e a adição de novas subclasses, como somente se faz necessário modificar ou adicionar uma única relação.

As desvantagens desta abordagem são as seguintes: A existência de muitas relações no banco de dados (uma para cada classe-&-objetos) e o aumento no tempo de leitura e escrita de dados, causado pela necessidade de se acessar múltiplas relações (Ambler, 1998; Agarwal et al., 1998).

Estratégias para cadastro no BDEC: #A3, #A4, #A5 e #A6, #A9, A#10.

3.3.2 Estratégias de representação de conexões

- ✓ Ao aplicar as estratégias de representação de conexões, deve-se sempre escolher anteriormente qual a melhor estratégias a ser utilizada. Isto pode ser feito através da estratégia de avaliação de conexões.

Existem basicamente três tipos de representações de conexões, que consistem em (Agarwal et al., 1998):

- Relação distinta: Nesta abordagem, a associação é representada como uma relação distinta no banco de dados.
- Chave embutida: Nesta abordagem, a representação do identificador de uma das classes-&-objetos envolvidas na associação é embutida na relação que representa a outra classe-&-objetos a ela associada.
- Classe embutida: Nesta abordagem, duas classes-&-objetos são mescladas em uma única relação.

A escolha do tipo de representação a ser usada dependerá das necessidades da aplicação em particular, tais como flexibilidade e eficiência.

Os tipos de representações existentes e os fatores a considerar para a escolha do melhor tipo de representação para a conexão em questão, são apresentados na Tabela 3.2.

TABELA 3.2 – FATORES A CONSIDERAR PARA A REPRESENTAÇÃO DE CONEXÕES

	1:1	**1:N**	**N:N**
Relação Distinta	Ineficiente	Flexível	Eficiente
Chave Embutida	Flexível	Eficiente	Não utilizada
Classe Embutida	Eficiente	Não utilizada	Não utilizada

Adaptada de Agarwal et al. (1998).

O parâmetro *ineficiente* significa que a abordagem pode ser utilizada para o grau de conectividade especificado, porém somente quando não se tiver outra alternativa.

O parâmetro *flexível* indica que, com o uso da abordagem especificada, fica fácil de se manter o modelo, mesmo se futuramente o grau de conectividade das associações necessitar de de ser alterado.

O parâmetro *eficiente* indica a melhor abordagem para o grau de conectividade especifico, desde que o modelo não necessite de flexibilidade.

A Figura 3.5 exemplifica os três tipos de representação de conexões.

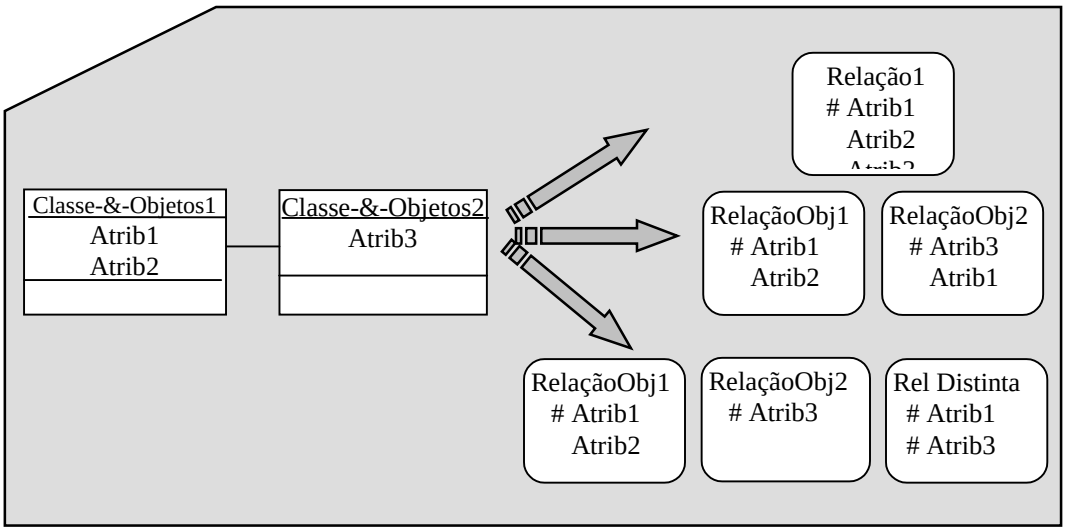


Fig. 3.5 – Tipos de representação de conexão.

Estratégia #M10 // Avaliação das abordagens de representação de associações

Descrição:

Levando em consideração as características das três diferentes abordagens de representação de associações, deve-se escolher a que melhor se adequa às necessidades do modelo em questão, considerando-se a princípio o grau de conectividade de cada associação.

Comentários:

Relação distinta: Esta abordagem pode ser utilizada para associações 1:1, N:1 e N:N. Ela promove maior flexibilidade de modelagem, pois faz com que a remoção e a adição de associações sejam transparentes a outras relações. Contudo, esta abordagem pode ser cara se a associação for frequentemente percorrida.

Chave embutida: Esta abordagem pode ser utilizada para associações 1:1 e N:1. Ela resulta numa característica de performance melhor que a da abordagem da "relação distinta", porém pior que a da abordagem da "classe embutida".

Classe embutida: Esta abordagem pode ser utilizada somente para associações 1:1.Ela melhora o tempo de consulta no banco de dados, já que a necessidade de junções neste caso se torna bem menor.

Estratégia #M11 // Representação de associações 1:1

Descrição:

Relação distinta: Deve-se criar uma relação para representar cada classe-&-objetos que participa da associação, cada qual com seus respectivos atributos, e ainda, uma outra relação para representar a associação, a qual deverá conter as representações dos identificadores das classes-&-objetos envolvidas na associação. A chave primária das relações que representam as classes-&-objetos envolvidas na associação, deverá ser a representação dos identificadores das próprias classes-&-objetos representadas. A chave primária da relação, criada para representar a associação, deverá ser composta pela representação dos identificadores das classes-&-objetos envolvidas na associação. (Brown e Whitenack, 1998; Object Matter, 1998).

Chave embutida: Deve-se criar uma relação para representar cada classe-&-objetos que participam da associação, cada qual com seus respectivos atributos. Cria-se uma coluna em uma das relações que representam as classes-&-objetos envolvidas na associação, para se incluir a representação do identificador da outra classe-&-objetos (Ambler, 1998). A chave primária de cada relação deverá ser a representação dos identificadores das classes-&-objetos que elas representam.

Classe embutida: Para este tipo de associação, esta abordagem é indicada quando a associação não tiver nenhum significado individual dentro do sistema a ser modelado. Deve-se mesclar os atributos das classes-&-objetos envolvidas na associação, representando-os através de colunas, em uma única relação (Brown e Whitenack, 1998). A chave primária da relação deverá ser a representação de qualquer um dos identificadores das classes-&-objetos que participam da associação.

Comentários:

Caso a conectividade tenha a opção de 0-1 de um dos dois lados, deve-se utilizar as abordagens de relação distinta ou chave embutida. Para a utilização da abordagem de chave embutida, a coluna a ser criada para a representação do identificador da classe-&-objetos deverá ser inserida na relação que representa a classe-&-objetos com conectividade 1. Para este caso, o uso da abordagem de classe embutida não é aconselhável por gerar lacunas na relação.

Caso a conectividade tenha a opção de 0-1 dos dois lados, deve-se utilizar a abordagem de relação distinta, pois a utilização das outras abordagens não é aconselhável por gerar lacunas na relação.

Estratégias para cadastro no BDEC: #A3, #A4, #A5, #A7, #A8, A#9, A#10.

Estratégia #M12 // Representação de associações N:1

Descrição:

Relação distinta: Deve-se criar uma relação para representar cada classe-&-objetos que participam da associação, cada qual com seus respectivos atributos, e ainda, uma outra relação para representar a associação, a qual deverá conter as representações dos

identificadores das classes-&-objetos envolvidas na associação. A chave primária das relações que representam as classes-&-objetos envolvidas na associação, deverá ser a representação dos identificadores das próprias classes-&-objetos representadas. A chave primária da relação criada para representar a associação, deverá ser composta pela representação dos identificadores das classes-&-objetos envolvidas na associação. (Brown e Whitenack, 1998; Object Matter, 1998).

Chave embutida: Deve-se criar uma relação para representar cada classe-&-objetos que participam da associação, cada qual com seus respectivos atributos. Cria-se uma coluna na relação que representa a classe-&-objetos com a conectividade “1”, para se incluir a representação do identificador da outra classe-&-objetos envolvida na associação (Ambler, 1998). A chave primária de cada relação deverá ser a representação dos identificadores das classes e objetos representadas.

Comentários:

Caso a conectividade tenha a opção N:0-1ou 1-N:0-1, não é aconselhável que se utilize a abordagem de chave embutida, por ela gerar lacunas na relação.

Caso a conectividade tenha opção 1-N:1, podem ser utilizadas as duas abordagens apresentadas anteriormente, sendo que a mais indicada é a abordagem de chave embutida.

Estratégias para cadastro no BDEC: #A3, #A4, #A5, #A7, #A8, #A9, #A10.

Estratégia #M13 // Representação de estrutura todo-parte

Descrição:

Uma estrutura todo-parte pode ser representada no modelo relacional através das abordagens de chave embutida e relação distinta.

Relação distinta: Se a associação possuir conectividade N:N, deve-se criar uma relação para representar cada classe-&-objetos que participam da associação, cada qual com seus respectivos atributos, e ainda, uma outra relação para representar a associação, a qual deverá conter a representação dos identificadores das classes-&-objetos envolvidas na associação. A chave primária das relações que representam as classes-&-objetos

envolvidas na associação deverá ser a representação dos identificadores das próprias classes-&-objetos representadas. A chave primária da relação criada para representar a associação, deverá ser composta pela representação dos identificadores das classes-&-objetos envolvidas na associação.

Chave embutida: deve-se criar uma relação para cada classe-&-objetos que participam da associação, cada qual com seus respectivos atributos. Deve ser criada uma coluna de chave estrangeira embutida, na relação que representa a "classe-&-objetos proprietária" da associação, a qual deverá conter a representação do identificador da "classe-&-objetos possuída". A chave primária de cada relação deverá ser a representação dos identificadores das classes-&-objetos que elas representam.

Comentários:

Caso a conectividade tenha a opção N:0-1ou 1-N:0-1, deve-se utilizar a abordagem de relação distinta. Não é aconselhável que se utilize a abordagem de chave embutida, pois ela gera lacunas na relação.

Caso a conectividade tenha opção 1-N:1, podem ser utilizadas as duas abordagens apresentadas anteriormente, sendo que a mais indicada é a abordagem de chave embutida.

Estratégias para cadastro no BDEC: #A3, #A4, #A5, #A7, #A8, #A9, #A10.

Estratégia #M14 // Representação de associações N:N

Descrição:

Relação distinta: Deve-se criar uma relação para representar cada classe-&-objetos que participam da associação, cada qual com seus respectivos atributos, e ainda, uma outra relação para representar a associação, a qual deverá conter a representação dos identificadores das classes-&-objetos envolvidas na associação.

A chave primária das relações que representam as classes-&-objetos envolvidas na associação, deverá ser a representação dos identificadores das próprias classes-&-objetos representadas. A chave primária da relação criada para representar a associação, deverá ser a representação dos identificadores das classes-&-objetos envolvidas na

associação. (Brown e Whitenack, 1998).

Comentários:

Caso a conectividade tenha a opção N:1-N ou 1-N:1-N, deverá ser aplicada a descrição apresentada anteriormente para o caso N:N.

Estratégias para cadastro no BDEC: #A3, #A4, #A5, #A7, #A9, #A10.

Estratégia #M15 // Representação de auto-associações

Descrição:

As auto-associações são representadas no modelo relacional, utilizando-se as mesmas abordagens que as associações binárias com grau de conectividade correspondente, considerando-se que as 2 classes-&-objetos participantes da associação binária são representadas, neste caso, por uma única classe-&-objetos.

Comentários:

As auto-associações podem possuir conectividade 1:1, N:1, N:N e as demais variações referentes a estes casos.

Estratégias para cadastro no BDEC: #A3, #A4, #A5, #A7, #A8, #A9, #A10.

3.4 Estratégia para Normalização do Modelo Operacional

- ✓ Estando modeladas todas as classes-&-objetos do sistema e suas associações, deverá ser aplicada a estratégia de normalização do modelo operacional.

Estratégia #M16 // Normalizar de acordo com a Forma Normal de Boyce-Codd ("Boyce-Codd Normal Form" - BCNF)

Descrição:

Em uma relação *R*, verificar se todo determinante é chave candidata.

Caso haja algum determinante que não seja chave candidata em uma relação *R*, deverá ser criada uma nova relação, contendo este atributo determinante e o atributo dependente dele. A chave primária desta relação deverá ser representada pelo atributo determinante.

E ainda, deverá ser criada uma nova relação, contendo as chaves primárias da relação *R* original e da nova relação criada para representar o determinante e seu dependente, as quais deverão compor a chave primária desta relação.

Em alguns casos, a criação desta última relação poderá ser substituída pela migração da chave primária da relação *R*, para a relação criada para representar o atributo determinante (não chave candidata em *R*) e seu dependente.

Comentários:

Denomina-se de determinante funcional um atributo sobre o qual outro atributo seja dependente (totalmente dependente).

Uma Relação R está em BCNF se, e apenas se, todo determinante for chave candidata.

Quando se modificam as relações geradas pela modelagem, através desta estratégia, é necessário que os metadados do BDEC sejam atualizados.

CAPÍTULO 4

MODELAGEM E ALIMENTAÇÃO DOS METADADOS DO BDEC

Este capítulo irá tratar, conforme o objetivo apresentado na Figura 4.1, da descrição das estratégias para a alimentação dos metadados usados pelo Componente Gerenciador de Configuração de Sistema Orientado a Objetos (CMOOS), para que seja feita a interface entre o Componente Gerenciador de Dados (CGD) e o Banco de Dados. Para isto, se faz necessário primeiramente, que se defina o modelo de metadados.

O Apêndice E apresenta a aplicação e validação das estratégias descritas neste capítulo.

A relação entre as estratégias de modelagem do banco de dados e as estratégias de alimentação do BDEC sào apresentadas no Apêndice D.

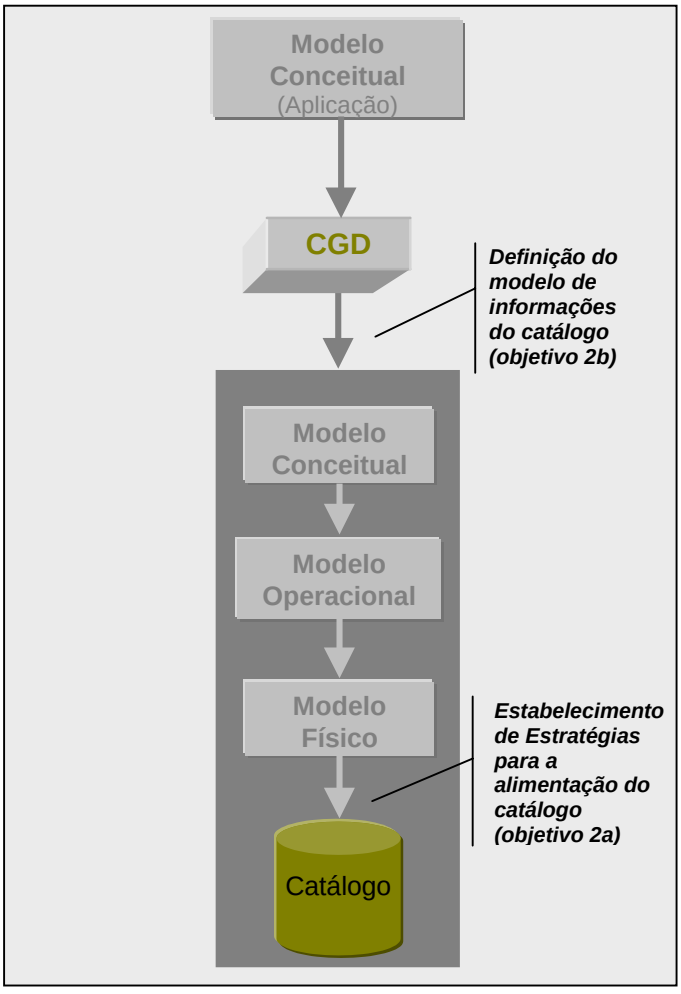


Fig. 4.1-Identificação do objetivo 2: Definição de modelo e alimentação dos metadados.

4.1 Modelagem dos Metadados Necessários para o CGD

Durante o Processo de Desenvolvimento de Software, vários modelos do sistema são construídos em níveis de abstração diferentes, como apresentado na Figura 2.4.

Para o processo de modelagem dos metadados necessários para o CGD, serão apresentados os seguintes níveis de abstração:

- Modelo Descritivo: descreve a percepção do mundo, a abstração em termos do domínio do problema e os requisitos para que o sistema possa ser desenvolvido;
- Modelo Conceitual: descreve o que o sistema deve fazer para satisfazer aos requisitos, dentro do domínio do problema. Este modelo é dividido em duas etapas: modelo ambiental e modelo comportamental, os quais serão apresentadas neste trabalho;
- Modelo Operacional: indica como o sistema será implementado, a fim de satisfazer aos requisitos, dentro do domínio do problema. Este modelo é dividido em duas etapas: modelo implementacional e modelo de programação. Neste trabalho será apresentado somente o modelo implementacional, que formaliza o modelo conceitual, adicionando estruturas e comportamentos necessários para a implementação do sistema.

4.1.1 Modelo Descritivo

A seguir, é apresentada uma breve descrição sobre como se relacionam os metadados necessários para que seja feita a interface entre os objetos do CDP da aplicação, o CGD e Banco de Dados Relacional.

Cada elemento persistente que participa do modelo orientado a objetos da aplicação, deve ser mapeado para o modelo de dados relacional.

O modelo de objetos de uma aplicação baseada em SOFTBOARD possui classes-&-objetos que constituem os Componentes do Sistema, que são classificados em: Componente Domínio do Problema, Componente Interface Humana, Componente Gerenciador de Cenários, Componente Gerenciador de Configuração de Sistema

Orientado a Objetos, Componente Gerenciador de Dados e Componente Interface com Sistemas (sendo que estes dois últimos podem não estar presentes no sistema). Cada componente pode possuir várias classes-&-objetos persistentes.

As classes-&-objetos do modelo deve possuem um identificador único, o qual é designado, na SOFTBOARD, pelo Componente Gerenciador de Dados.

Além de um identificador, as classes-&-objetos possuem atributos, os quais podem ser classificados em atributos simples (monovalorados ou multivalorados) ou atributos compostos (monovalorados ou multivalorados).

As classes-&-objetos podem possuir conexões (associações e estruturas todo-parte) ou pertencerem a uma estrutura de generalização/especialização.

Um modelo relacional consiste em uma coleção de relações, as quais são compostas por colunas e linhas, que representam respectivamente, atributos e tuplas.

As classes-&-objetos persistentes são mapeadas para relações e seus atributos persistentes são mapeados para as colunas destas relações.

Os atributos compostos podem ser mapeados para um ou vários atributos da relação, dependendo do grau de relevância dos seus dados, dentro do sistema.

Os atributos multivalorados, são mapeados para uma nova relação e, dependendo do seu grau de relevância dentro do sistema, deverá ser designado um identificador único para ele.

Uma conexão pode ser mapeada para uma relação ou ser representada através de uma chave estrangeira.

Toda relação deve conter uma chave primária, a qual pode ser representada pelo identificador de um atributo multivalorado, criado para este fim; pelo próprio atributo multivalorado em conjunto com a representação do identificador de sua classe-&-objetos de origem; pela representação do identificador da classe-&-objetos que uma relação representa; pelas representações dos identificadores das classes-&-objetos que, através de conexões, criaram uma nova relação.

4.1.2 Modelo Conceitual

4.1.2.1 Modelo Ambiental

O objetivo de se modelar os metadados necessários para o CGD consiste na necessidade de manter informações sobre os objetos do sistema, para que o CGD da SOFTBOARD possa ser configurável e interagir com o banco de dados da aplicação.

Os objetos do sistema solicitam o seu armazenamento e recuperação no banco de dados. Quem atende a este evento é o CGD. Porém, para que ele possa atender estas solicitações, o CMOOS tem que disponibilizar as informações sobre os objetos do sistema.

4.1.2.2 Modelo Comportamental

Facilidades do sistema:

Para manter informações para atender às solicitações de manipulação dos objetos do sistema, no banco de dados, deve-se:

- Registrar informações sobre a localização, no banco de dados, das classes-&-objetos persistentes, seus atributos persistentes e suas conexões, as quais são mapeadas do modelo orientado a objetos para o modelo relacional.
- Registrar ainda, todas as relações existentes entre as classes-&-objetos dos próprios modelos, orientado a objetos e relacional.

Modelo do componente domínio do problema:

A Figura 4.1, apresenta o modelo de objetos do domínio do problema, que é composto de classes-&-objetos do modelo orientado a objetos, de classes e objetos que representam o modelo relacional e de classes-&-objetos que representam o mapeamento entre estes dois modelos.

Neste caso, são atribuídos apenas os atributos de importância para os metadados.

Para a representação das Classes-&-Objetos do modelo do componente domínio do problema, é utilizada a notação da UML, que é descrita com maiores detalhes no Apêndice B.

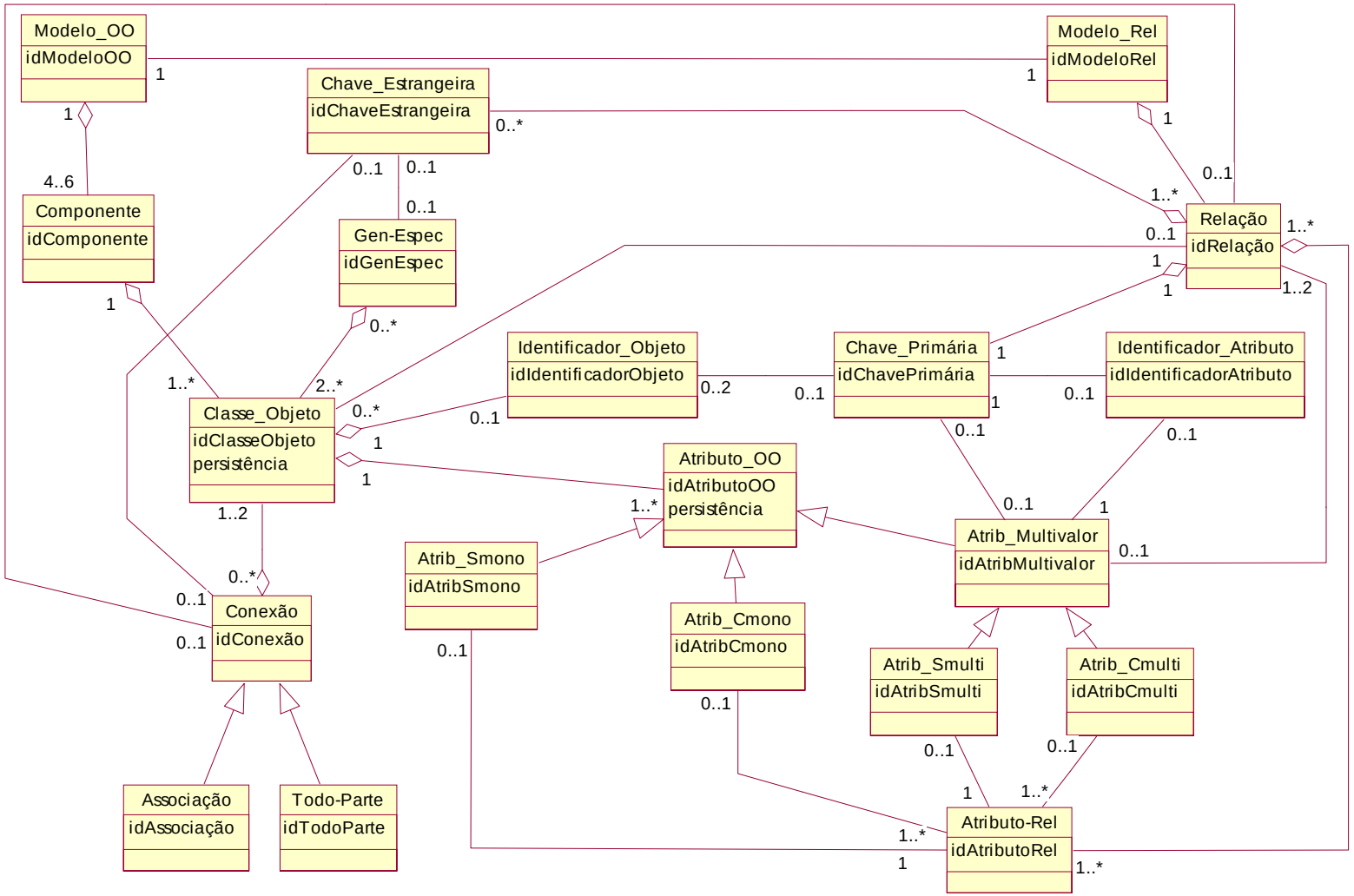


Fig. 4.2 - Modelo do domínio do problema dos metadados necessários para o CGD.

4.1.3 Modelo Operacional

O modelo operacional utilizado para a definição do modelo dos metadados foi o modelo de dados relacional.

O modelo relacional dos metadados foi construído através da utilização das estratégias criadas para a modelagem do banco de dados das aplicações baseadas na SOFTBOARD, apresentadas no capítulo 3.

Para a representação das estruturas (relações) do modelo operacional, é utilizada a notação Case Method, que é descrita com maiores detalhes no Apêndice C.

Aplicação das estratégias para a geração do modelo relacional de dados

A seguir serão apresentados os passos utilizados durante a construção do modelo relacional dos metadados.

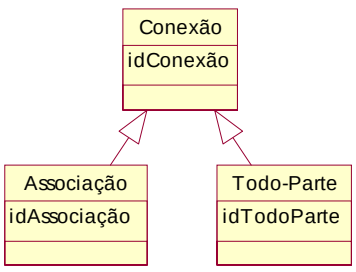
Neste caso, utilizam-se estratégias de representação que tratam de suporte a herança, e suporte a conexões.

Para a aplicação das estratégias, as classe-&-objetos do modelo do Domínio do Problema dos metadados necessários para o CGD, foram agrupadas de acordo com seus relacionamentos, em quatro casos: estruturas de generalização/especialização, estruturas todo-parte, associações 1:1 e associações N:1 com suas respectivas variações.

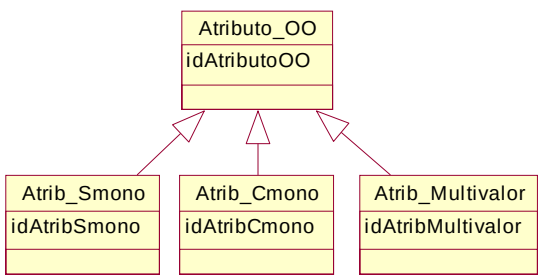
Caso 1: Estruturas de Generalização/Especialização

As estruturas de Generalização/Especialização encontradas no modelo apresentado na Figura 4.2 são as seguintes:

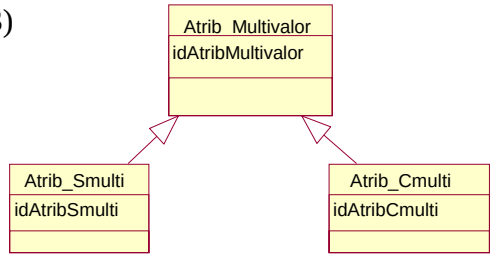
1)



2)



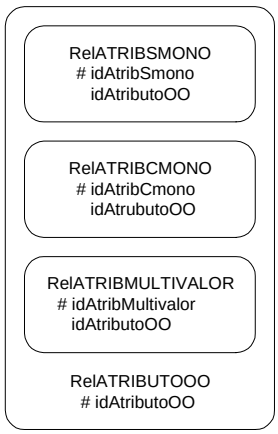
3)



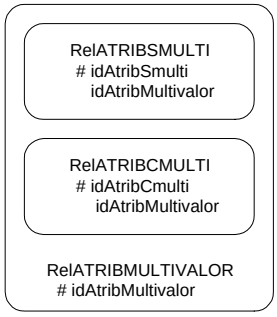
Para a modelagem das estruturas ①, ② e ③, utilizou-se a Estratégia #M9 e decidiu-se utilizar a abordagem de representação que consiste na criação de uma relação por classe-&-objetos da hierarquia, pois assim são minimizados os relacionamentos entre as relações. As relações geradas para estas estruturas são apresentadas a seguir:

1)

2)



3)

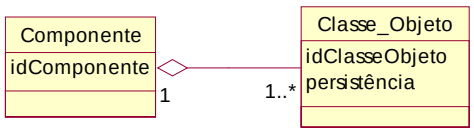


Caso 2: Estruturas Todo-Parte

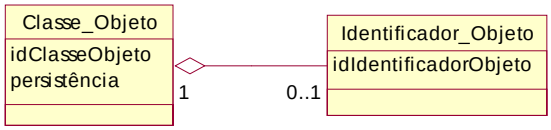
As estruturas Todo-Parte encontradas no modelo apresentado na Figura 4.2 são as seguintes:

1)

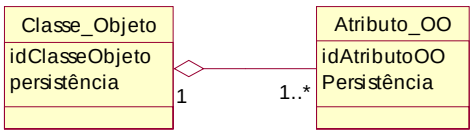
2)



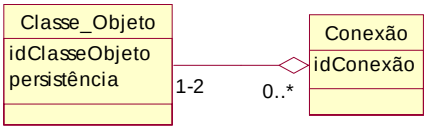
3)



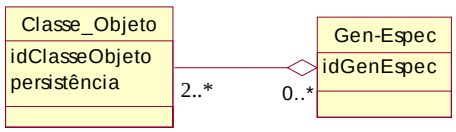
4)



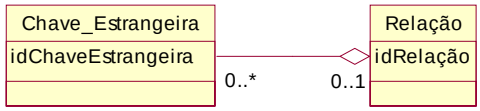
5)



6)

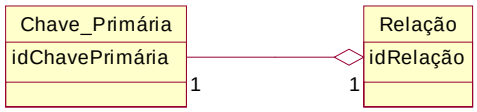


7)

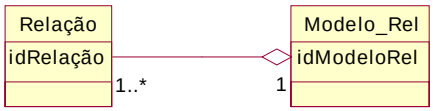


8)

9)

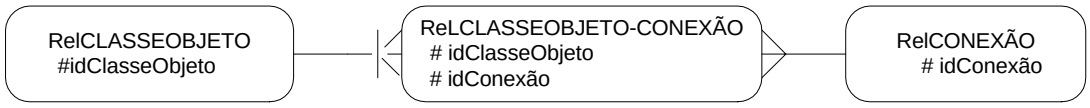


10)

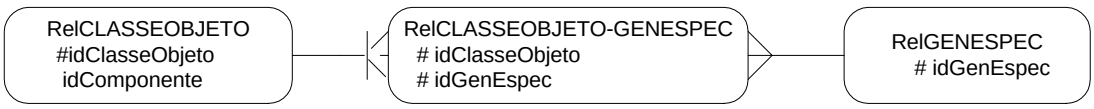


Para a modelagem das estruturas ⑤, ⑥ e ⑧, que possuem respectivamente as conectividades N:1-2, N:2-N e 1-N:1-N, utilizou-se a Estratégia #M13, que implica na utilização da abordagem de relação distinta, sendo esta a única indicada para representar estes tipos de conexões. Esta abordagem também foi utilizada para a estrutura ⑦, que possui conectividade 0-1:N, pois a utilização da abordagem de chave estrangeira para este caso, geraria lacunas na relação. As relações geradas para estas estruturas são apresentadas a seguir:

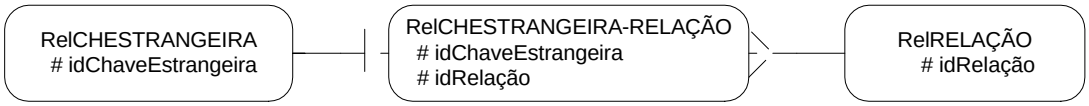
5)



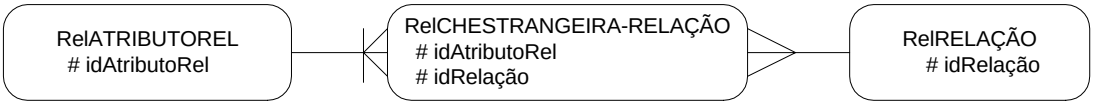
6)



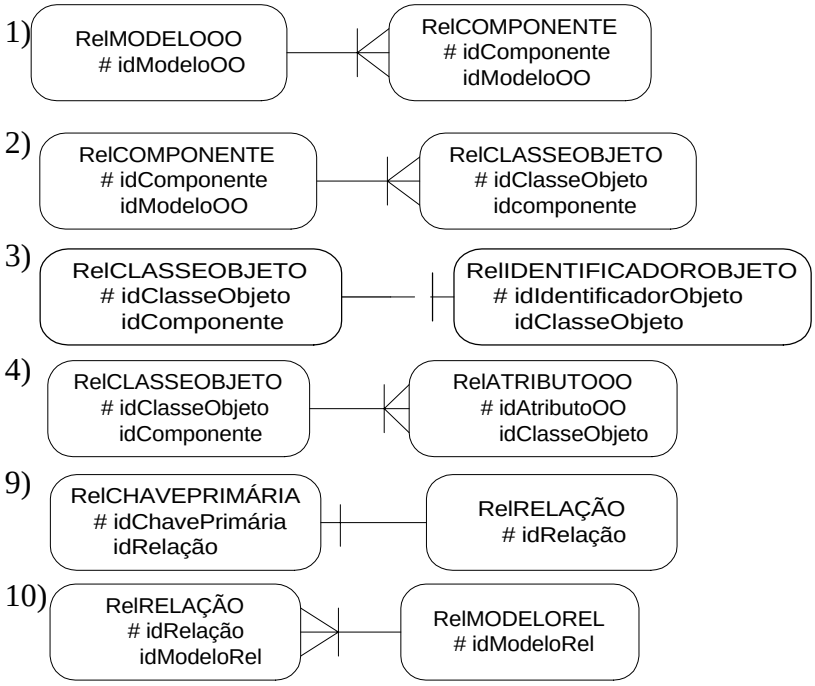
7)



8)

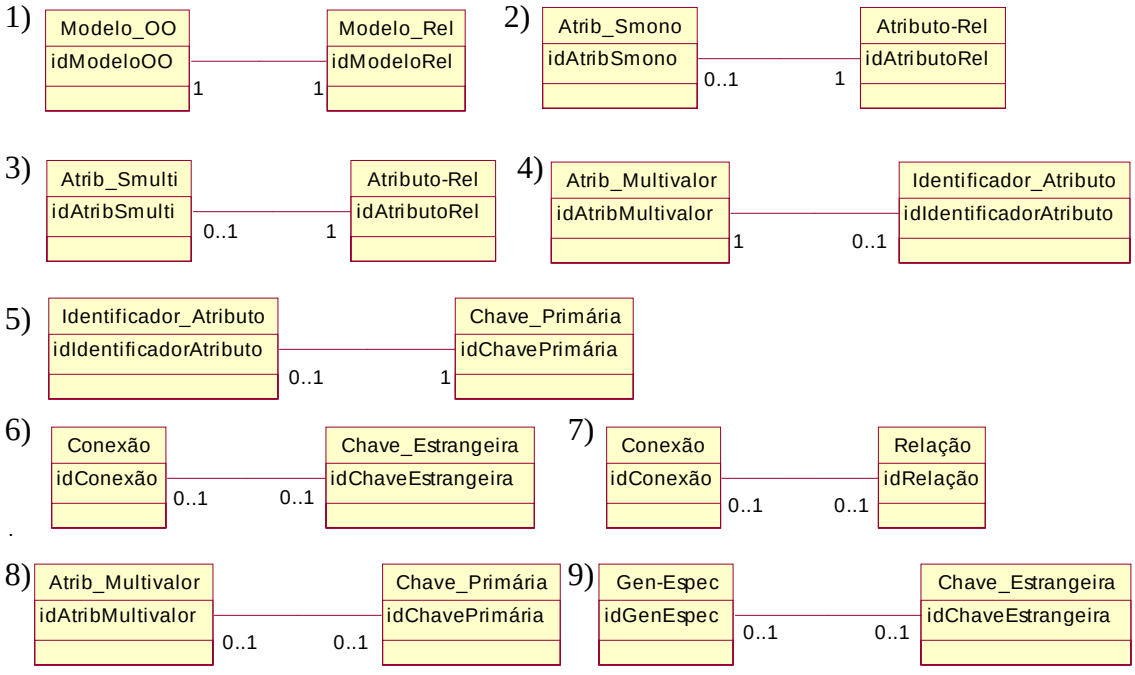


Para a modelagem das demais estruturas, com conectividade 1:1, N:1 e suas respectivas variações, foram utilizadas as Estratégias #M11 e #M12, através das quais decidiu-se utilizar a abordagem de chave embutida, pois ela gera uma característica de performance melhor que a da abordagem de relação distinta. As relações geradas para estas estruturas são apresentadas a seguir:



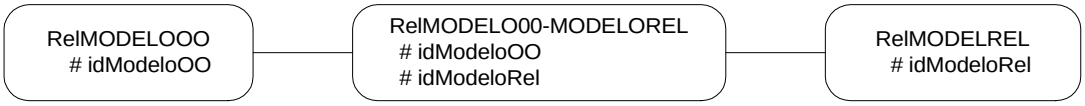
Caso 3: Associações 1:1 e suas variações (0-1:1 e 0-1:0-1)

As Associações 1:1 (com suas variações) encontradas no modelo apresentado na Figura 4.2 são as seguintes:



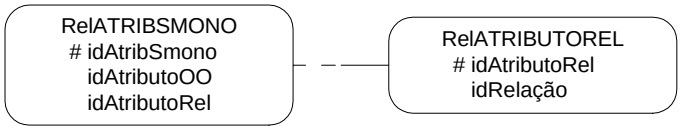
Para a modelagem da estrutura ①, que possui conectividade 1:1, utilizou-se a Estratégia #M11 e decidiu-se utilizar a abordagem de relação distinta, pois tal associação possui uma significativa relevância dentro do sistema, e ainda, cada uma de suas classes-&-objetos participam da representação de *modelos* diferentes dentro do CDP. A relação gerada para esta estrutura é a seguinte:

1)

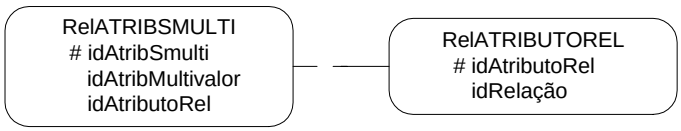


Para a modelagem das estruturas ②, ③, ④ e ⑤, que possuem conectividade 1:0-1, utilizou-se a Estratégia #M11 e decidiu-se utilizar a abordagem de chave embutida, pois ela gera uma característica de performance melhor que a da abordagem de relação distinta. As relações geradas para estas estruturas são apresentadas a seguir:

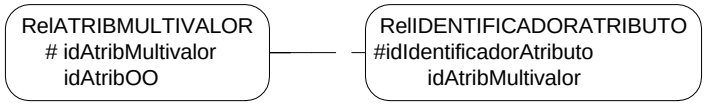
2)



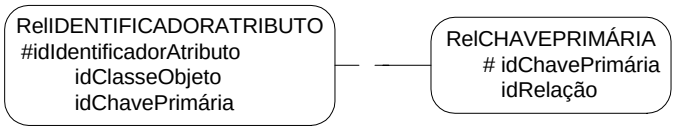
3)



4)

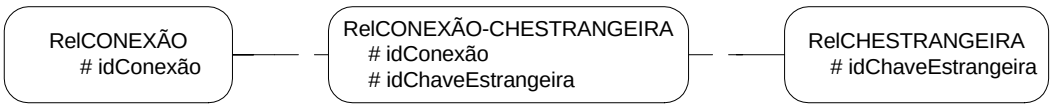


5)

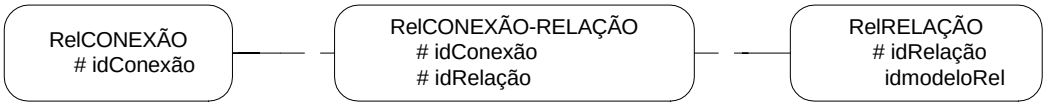


Para a modelagem das estruturas ⑥, ⑦, ⑧ e ⑨, que possuem conectividade 0-1:0-1, foi utilizada a Estratégia #M11, que implica na utilização da abordagem de relação distinta, pois ela é a única indicada para este tipo de associação.

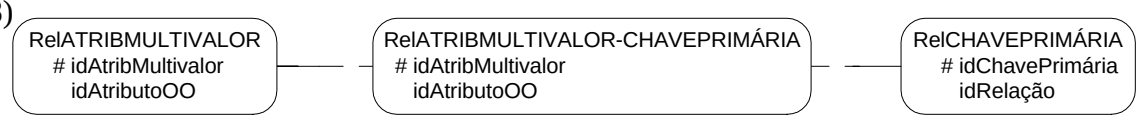
6)



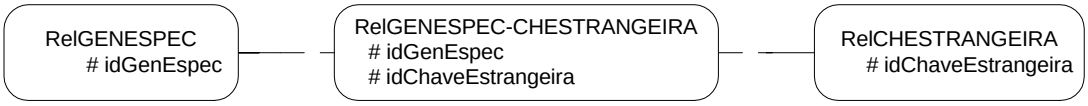
7)



8)



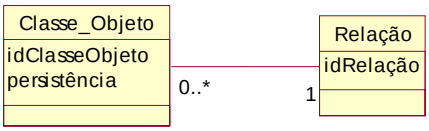
9)



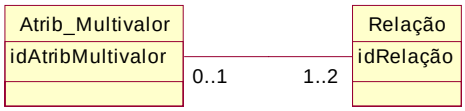
Caso 4: Associações N:1 e suas variações (0-1:1-N, 0-1:1-2 e 0-2:0-1)

As Associações N:1 (com suas variações) encontradas no modelo apresentado na Figura 4.2 são as seguintes:

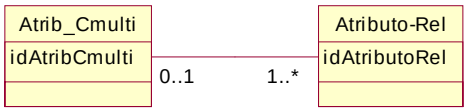
1)



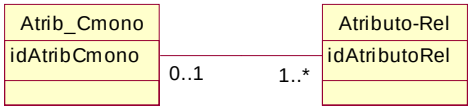
2)



3)



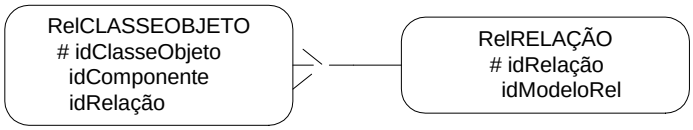
4)



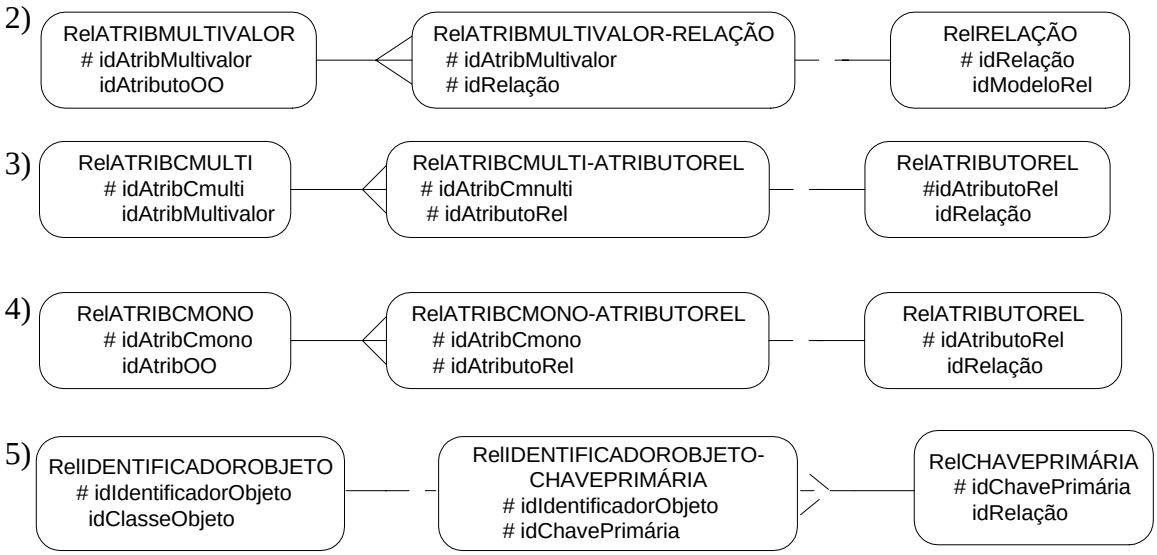
5)

Para a modelagem da estrutura ①, que possui conectividade N:1, utilizou-se a Estratégia #M12 e decidiu-se utilizar a abordagem de chave embutida, pois ela gera uma característica de performance melhor que a da abordagem de relação distinta, como citado anteriormente. A relação gerada para esta estrutura é a seguinte:

1)



Para a modelagem das demais associações, que possuem variações da conectividade N:1, utilizou-se a Estratégia #M12, que implica na utilização da abordagem de relação distinta, pois ela é a única indicada para estes tipos de associações. As relações geradas para as estruturas ②, ③, ④ e ⑤, são apresentadas a seguir:



Todas as relações geradas a partir do modelo apresentado na figura 4.2, encontram-se normalizadas de acordo com a BCNF. Deste modo, não há necessidade para se utilizar a estratégia de normalização. A Figura 4.3 apresenta o modelo final de relações dos metadados do BDEC necessários para o CGD.

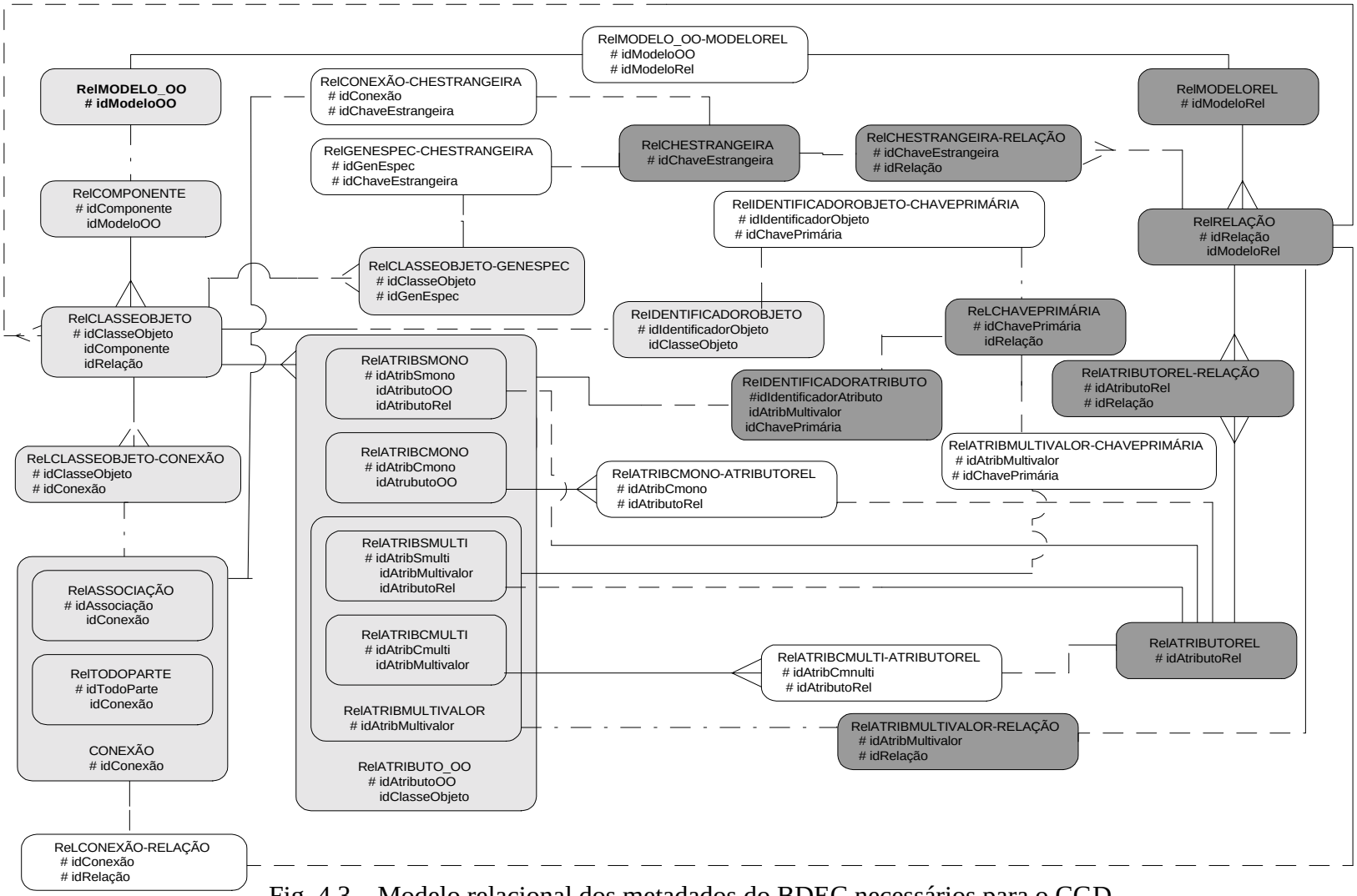


Fig. 4.3 – Modelo relacional dos metadados do BDEC necessários para o CGD

4.2 Estratégias para Alimentação dos Metadados

Para que o CGD da SOFTBOARD seja configurável é necessário que os elementos persistentes do CDP sejam mantidos no BDEC, assim como as informações sobre seus mapeamentos para o banco de dados relacional.

O cadastro dos elementos do CDP, mantidos no BDEC, deve ser feito durante o processo de modelagem da aplicação baseada na SOFTBOARD.

As estratégias apresentadas a seguir, completam as informações do BDEC com o cadastro dos elementos do modelo relacional e das informações de como eles se relacionam com o modelo de objetos do Componente Domínio do Problema.

As estratégias são apresentadas da seguinte forma:

- um identificador da estratégia: uma letra de identificação e uma numeração (por exemplo: #A1, #A2 - onde A identifica que é uma estratégia de alimentação);
- um título, que descreve o propósito da estratégia;
- um tópico de descrição: representa a estratégia propriamente dita, apresentando uma declaração textual dos passos a serem seguidos para se atingir um objetivo específico e
- um tópico de comentário: apresenta uma referência às estratégias de modelagem que geraram os metadados.

Estratégia #A1 // Cadastrar Modelo Relacional

Descrição:

→ Cadastrar modelo relacional.

→ Relacionar o modelo orientado a objetos com o modelo relacional, para o qual ele está sendo mapeado.

Estratégia #A2 // Atualizar Persistência de Classes-&-Objetos e Atributos

Descrição:

→ Atualizar o atributo "*persistência*" de cada classe-&-objetos persistente do

sistema.

→ Atualizar o atributo "*persistência*" de cada atributo persistente do sistema.

Comentários:

Estratégias geradoras: #M1 e #M2.

Estratégia #A3 // Cadastrar Relações Geradas por Classes-&-Objetos

Descrição:

→ Cadastrar relações.

→ Relacionar as classes-&-objetos com as relações, para as quais elas foram mapeadas.

Comentários:

Estratégias geradoras: #M4, #M7, #M8, #M9, #M11, #M12, #M13, #M14 e #M15.

Estratégia #A4 // Cadastrar Atributos Relacionais

Descrição:

→ Cadastrar atributos relacionais.

→ Relacionar atributos simples monovalorados com os atributos relacionais, para os quais eles foram mapeados.

→ Relacionar atributos simples multivalorados com os atributos relacionais, para os quais eles foram mapeados.

→ Relacionar atributos compostos monovalorados com os atributos relacionais, para os quais eles foram mapeados.

→ Relacionar atributos compostos multivalorados com os atributos relacionais, para os quais eles foram mapeados.

→ Relacionar atributos relacionais com as respectivas relações, as quais eles pertencem.

Comentários:

Estratégias geradoras: #M4, #M5, #M7, #M8, #M9, #M11, #M12, #M13, #M14 e #M15.

Estratégia #A5 // Cadastrar Relações Geradas por Atributos Multivalorados

Descrição:

- → Cadastrar relações.
- → Relacionar atributos multivalorados com as relações geradas para representá-los.

Comentários:

Estratégias geradoras: #M4, #M5, #M7, #M8, #M9, #M11, #M12, #M13, #M14 e #M15.

Estratégia #A6 // Cadastrar Chave Estrangeira Gerada para Mapear Estruturas de Generalização/Especialização

Descrição:

- → Cadastrar chave estrangeira.
- → Relacionar estruturas de generalização/especialização com a chave estrangeira, a qual representa o mapeamento da estrutura.
- → Relacionar relações com as chaves estrangeiras que pertencem a elas.

Comentários:

Estratégias geradoras: #M8 e #M9.

Estratégia #A7 // Cadastrar Relações Geradas por Conexões

Descrição:

- → Cadastrar relações.
- → Relacionar conexões com as relações que foram geradas por elas, para as quais elas foram mapeadas.

Comentários:

Estratégias geradoras: #M11, #M12, #M13, #M14 e #M15.

Estratégia #A8 // Cadastrar Chave Estrangeira Gerada para Mapear Conexões

Descrição:

- → Cadastrar chave estrangeira.
- → Relacionar conexões com a chave estrangeira, a qual representa o seu mapeamento.
- → Relacionar relações com as chaves estrangeiras que pertencem a elas.

Comentários:

Estratégias geradoras: #M11, #M12, #M13 e #M15.

Estratégia #A9 // Cadastrar Chaves Primárias das Relações Geradas por Atributos Multivalorados

Descrição:

- → cadastrar chaves primárias.
- → Relacionar relações com as respectivas chaves primárias que cada uma delas possui.
- → Relacionar os atributos multivalorados com as chaves primárias que eles representam, as quais identificam as relações que eles geraram.
- → Relacionar os identificadores de atributos, os quais foram criados para identificar as relações geradas por atributos multivalorados, com as chaves primárias que eles representam. Relacionar ainda, ao identificador de atributo, o atributo multivalorado que o gerou.

Comentários:

Estratégia geradora: #M4, #M5, #M7, #M8, #M9, #M11, #M12, #M13, #M14, #M15.

Estratégia #A10 // Cadastrar Chaves Primárias

Descrição:

- → cadastrar chaves primárias.

- → Relacionar os identificadores das classes-&-objetos com as chaves primárias que eles representam.
- → Relacionar relações com as chaves primárias, as quais elas possuem.

Comentários:

Estratégia geradora: #M4, #M6, #M7, #M8, #M9, #M11, #M12, #M13, #M14, #M15.

CAPÍTULO 5

CONCLUSÕES

Este trabalho de pesquisa concentrou-se nos elementos necessários para a definição de um processo baseado em estratégias e padrões, para a modelagem do banco de dados de uma aplicação que utiliza a arquitetura SOFTBOARD, modelando também as informações necessárias para permitir que o CGD seja configurável , já que ele é o componente responsável pela interface entre os objetos do CDP da aplicação e o banco de dados.

Para a modelagem do banco de dados das aplicações que utilizam a arquitetura SOFTBOARD, foi definido um conjunto de estratégias e padrões, com a finalidade de geração do modelo de dados relacional.

A utilização das estratégias e padrões para a modelagem do banco de dados para sistemas baseados na arquitetura SOFTBOARD, melhora a qualidade, já que contribui para uma menor probabilidade de erros no projeto e uma melhor documentação do mesmo, facilitando ainda, sua concepção e futura manutenção.

Ocorre ainda, um aumento da produtividade, já que há uma diminuição do tempo de trabalho, reduzindo o esforço necessário para se desenvolver e manter o banco de dados.

Através da utilização destas estratégias e padrões, foi definido o modelo de metadados que o CMOOS irá utilizar, para que seja feita a interface entre o CDP da aplicação, o CGD e o banco de dados. Para a alimentação do modelo de metadados, foram criadas estratégias, que devem ser utilizadas durante o processo de modelagem do banco de dados.

A definição do modelo de metadados contribui sobretudo para a reutilização de software, permitindo que o CGD seja configurável para várias aplicações que utilizam a arquitetura de SOFTBOARD.

O benefício primordial da reutilização é a produtividade. A qualidade também é outra razão para se enfatizar a reutilização, pois um componente reutilizável sempre requer mais qualidade do que o seu correspondente não reutilizável.

Portanto, de modo geral, através do processo de modelagem do banco de dados e da definição do modelo de metadados, este trabalho de pesquisa contribui para a concretização do Componente Gerenciador de Dados da SOFTBOARD, contribuindo assim, para a melhoria da qualidade e da produtividade na construção dos sistemas de software.

Espera-se que o processo de modelagem apresentado neste trabalho, seja utilizado para a modelagem dos bancos de dados das aplicações baseadas em SOFTBOARD, e possivelmente complementada, através de propostas de novas estratégias e padrões, a medida que novas necessidades forem surgindo.

Espera-se que seja implementado o modelo de metadados sugerido, e que este possa ser utilizado como base para a definição do modelo dos demais metadados necessários para que os componentes da SOFTBOARD sejam configuráveis.

Dentre os trabalhos futuros, espera-se que sejam projetados e implementados os componentes: Gerenciador de Configuração de Sistema Orientado a Objetos, Gerenciador de Cenários e Interface com outros Sistemas, para que o projeto da SOFTBOARD possa ser concretizado.

REFERÊNCIAS BIBLIOGRÁFICAS

Agarwal, S.; Keene, C.; Keller, A. **Architecting object applications for high performance with relational databases**. [online]. <http://www-db.stanford.edu/keller/1995/high-perf.ps>. 15 Feb. 1998.

Alexander, C. **Timeless way of building**. New York: Oxford University Press, 1979. 552 p.

Ambler, S. W. **Building object applications that work.** Cambridge: Cambridge University Press, 1998. 476 p.

Anderson, J. R. **Cognitive psycology and its implications.** 4. ed. New York: W. H. Freeman, 1995. 519 p.

Booch, G. **Patterns**. Rational Software Corporation. [online]. <http://www.rational.com/sitewide/support/whitepapers/dynamic.jtmpl?doc_key=283>. 18 Aug. 1998.

Brown, K.; Whitenack, B. G. **Crossing chasms:** a pattern language for object-RDBMS integration - “The Static Patterns”. [online]. <http://www.ksccary.com/articles/objectRDBMSPattern/objectRDBMSPattern.htm>. 03 Feb. 1998.

Buzato, L. E.; Rubira, C. M. F. **Construção de sistemas orientados a objetos confiáveis**. Rio de Janeiro: DCC/IM, COPPE Sistemas, 1998. 170 p.

Coad, P.; Yourdon, E. **Análise baseada em objetos**. 2. ed. Rio de Janeiro: Campus, 1992. 225 p.

____ **Projeto baseado em objetos**. Rio de Janeiro: Campus, 1993. 195 p.

Coad, P.; North, D.; Mayfield, M. **Object models: strategies, patterns, & applications.** Upper Saddle River: Prentice Hall, 1995. 505 p.

Coplien, J. O. **Software design patterns:** common questions and answers. [online]. <ftp://st.cs.uiuc.edu/pub/patterns/papers/PatQandA.ps>. 01 Oct. 1998a.

____ **Progress on patterns:** highlights of PloP/94. [online]. <ftp://st.cs.uiuc.edu/pub/patterns/papers/objectExpoPLoP.ps>. 10 Sep. 1998b.

Cougo, P. **Modelagem conceitual e projeto de bancos de dados**. Rio de Janeiro: Campus, 1997. 284 p.

Cunha, J. B. S. **Uma abordagem de desenvolvimento de software para a melhoria da qualidade e da produtividade**. São José dos Campos. 176 p. Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, 1997. No prelo.

Date, C. J. **Banco de dados tópicos avançados.** Rio de Janeiro: Campus, 1988. 361 p.

____ **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Campus, 1990. 674 p.

Davis, J. **Extended relational DBMSs:** the technology, part1. [online]. <http://www.dbmsmag.com/9706d13.html>. 03 Feb. 1998.

Dias, A. S. **Estratégias e padrões para desenvolvimento de sistemas de software baseado na arquitetura SOFTBOARD**. São José dos Campos. Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais. Em desenvolvimento.

Ferreira, M. G. V. **Componente gerenciador de dados configurável**. São José dos Campos. 126 p. Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, 1996. No prelo.

Fussell, M. L. **Foundations of relational mapping**. [online]. <http://www.chimu.com/publications/objectRelational/index.html>. 16 Feb. 1998.

Hay, D. C. **Princípios de modelagem de dados**. São Paulo: Makron Books, 1999. 271 p.

Korth, H. F.; Silberschatz, A. **Sistemas de banco de dados**. 2. ed. São Paulo: Makron Books, 1994. 748 p.

Martin, R. **Patterns:** PLoP, PLoP—fizz, fizz. C++ Report**.** OOD Column. [online]. <http://oma.com/pdf/>. (arquivo: PLoP.pdf). 18 Aug. 1998.

McKeown, D. W.; Saiedian, H. Triggers for object-oriented databases systems. **JOOP.** [online]. vol. 10, n. 2, may, 1997. <http://www.sigs.com>. 01 Dec. 1997.

Melton, J.; Simon, A.R. **Understanding the new SQL: a complete guide**. San Mateo: Morgan Kauffmann Publishers, 1993. 536 p.

Meszaros, G.; Doble, J. **A pattern language for pattern writing.** [online]. <http://www.hillside.net/patterns/writing/patterns.html>. 17 Sep. 1998.

Object Matter, Inc. **Object relational mapping strategies**. [online]. <http://www.objectmatter.com/vbsf/docs/maptool/ormapping.html>. 20 Aug. 1998.

Palepu, R. **Modeling the real world:** application of patterns to reduce complexity in the software development process. [online]. <http://www.scs.carleton.ca/~papelu/pat.html>. 22 Sep. 1998.

Poet Software. **Why use an ODBMS? an comparison between relational and object oriented databases for object oriented application development**. [online]. <http://www.poet.com/t_oovsre.htm>. 16 Feb. 1998.

Roddick, J. F. SQL/SE - A query language extension for database supporting schema evolution. **SIGMOD Record**, vol. 21, n. 3, Sep. 1992. p. 10-16.

Rothwell, D. **Storing objects in a database**. [online]. <http://www.kinetica.com/ootips/persist-objects.html>. 01 Apr. 1998.

Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensers, W. **Object oriented modeling and design**. New Jersey: Prentice Hall, 1991. 528 p.

Setzer, V. W. **Banco de dados – conceitos, modelos, gerenciadores, projeto lógico, projeto físico**. 3. ed. São Paulo: Edgard Blücher, 1995. 289 p.

Siegler, R. S., Jenkins, E. **How children discover new strategies.** New Jersey: Lawrence Erlbaum Associates, 1989, p. 1-20.

Souza, G. J. **Interface configurável**. São José dos Campos. 132 p. Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, 1997.

APÊNDICE A

BIBLIOGRAFIA COMPLEMENTAR

Appleton, B. **Patterns and software:** essential concepts and terminology. [online]. <http://shell.enteract.com/~bradapp/docs/patterns-intro.html>. 08 august 1998.

Blanco, C. da C. **Uso de banco de dados relacionais em sistemas orientados a objeto.** [online] <http://www.dcc.ufrj.br/~labbd/leitcomp/oobdrel.htm>. 18 maio 1998.

Buschman, F.; Meunier, R.; Rhonert, H; Sommerlad, Peter; Stal, M. **Pattern oriented software oriented architecture: a system of pattern**. John Wiley & Sons, 1996. 476p.

Fiorini, S. T.; Staa, A. V.; Baptista, R. M. **Engenharia de software com CMM**. Rio de Janeiro: Brasport Livros e Multimídia, 1998. 346 p.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J**. Design patterns: elements of reusable object-oriented software.** 11 ed. Reading: Addison-Wesley, 1994. 395 p.

Khan, E. H.; Al-A'ali, M.; Girgis, M. R. Object-oriented programming for structured procedural programmers. **Computer Magazine**, v. 28, n.10, p. 48-57, oct. 1995.

Khoshafian, S. **Banco de dados orientado a objeto**. Rio de Janeiro: Infobook S.A., 1994. 353 p.

Oliveira, D. L. de **Aspectos teóricos do modelo relacional de dados e projeto de relações**. São José dos Campos. 255 p. Dissertação (Mestrado em ciências na área de Informática) - Instituto Tecnológico de Aeronáutica, 1988.

Pressman, R. S. **Engenharia de software**. São Paulo: Makron Books, 1995. 1056 p.

Schmidt, D., Johnson, R. E.; Fayad, M. Software Patterns. **Comunications of the ACM**, vol. 39, n. 10, october 1996. Special issue on patterns and patterns languages. p. 37-39.

APÊNDICE B

NOTAÇÃO SIMPLIFICADA DA UML PARA O DIAGRAMA DE CLASSES

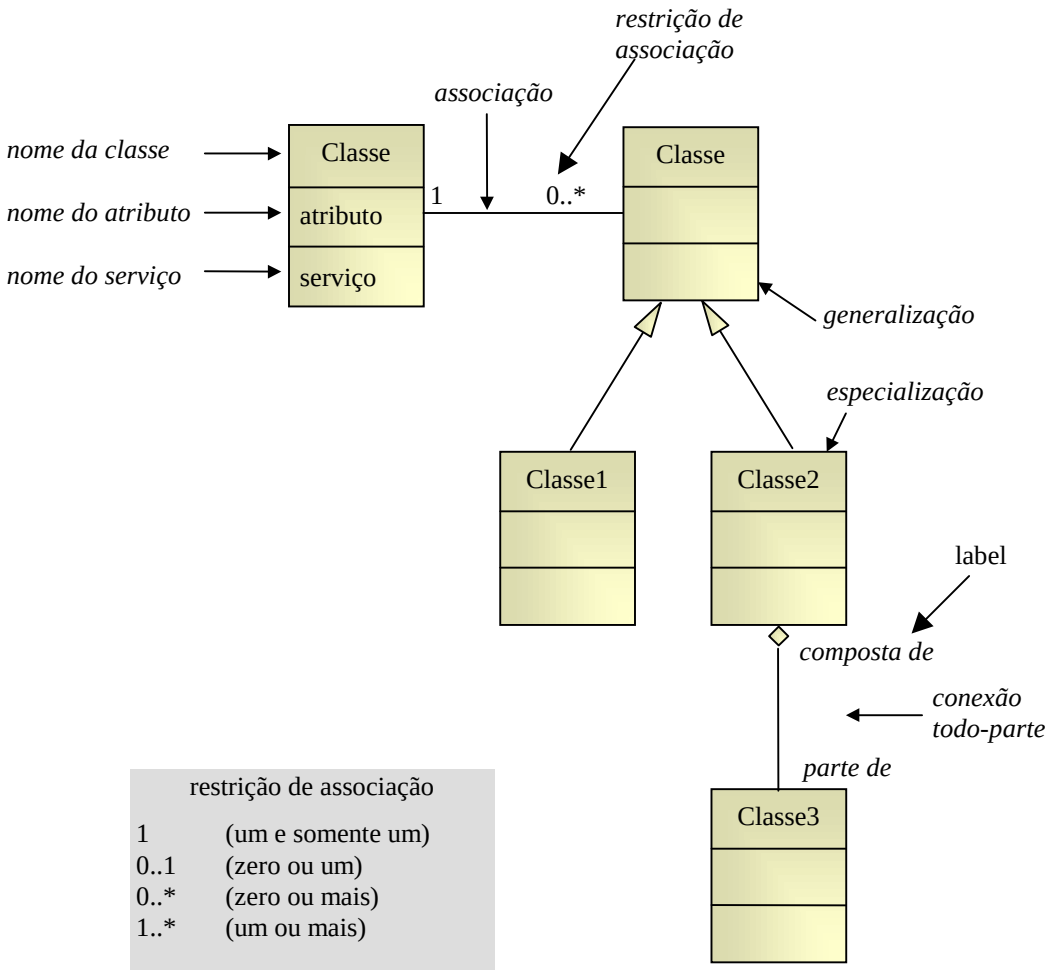


Fig. B.1 - Notação simplificada da UML para o diagrama de classes.

APÊNDICE C

NOTAÇÃO CASE METHOD – UTILIZADA PARA REPRESENTAÇÃO DE RELAÇÕES

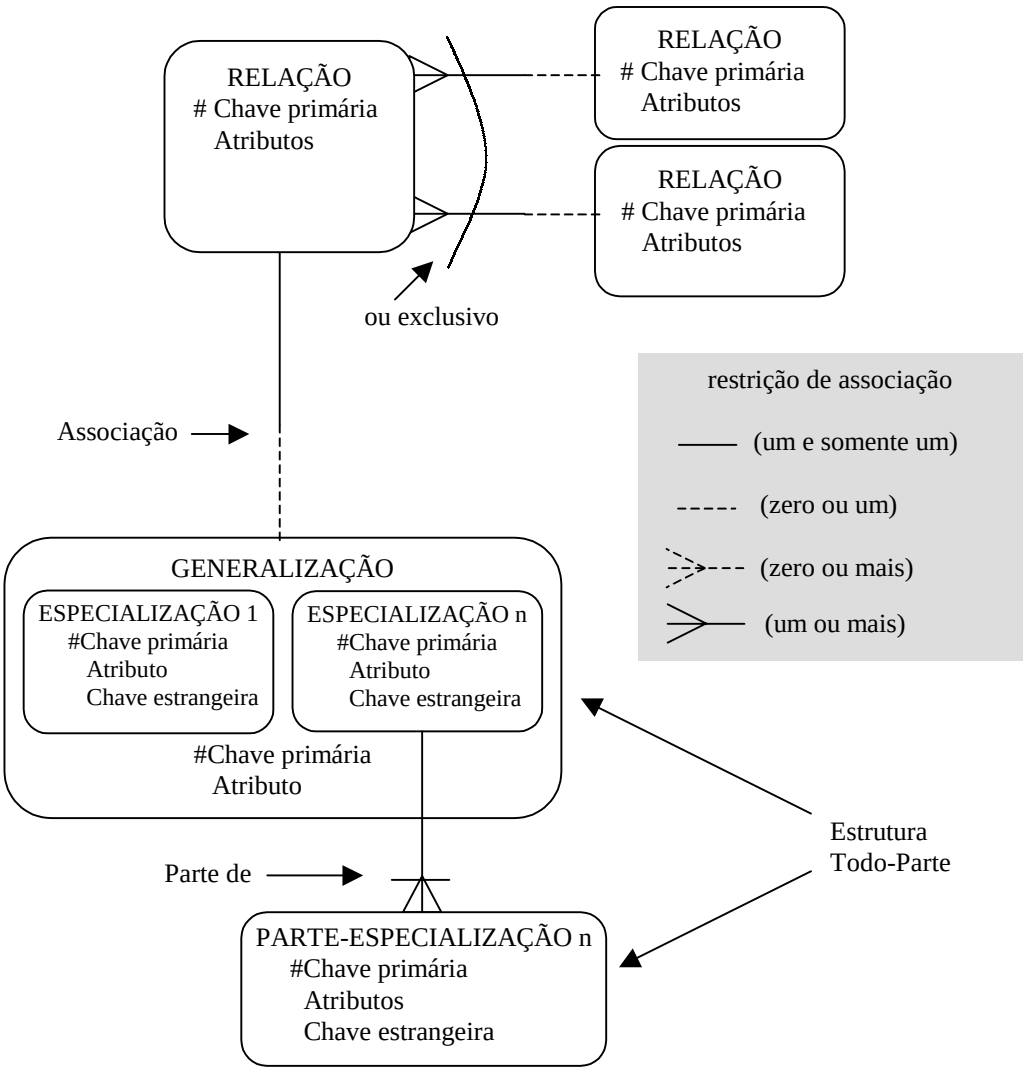


Fig. C.1 - Notação simplificada do Case Method.

Adaptada de Hay (1999).

APÊNDICE D

RELAÇÃO ENTRE AS ESTRATÉGIAS DE MODELAGEM DO BANCO DE DADOS E AS ESTRATÉGIAS DE ALIMENTAÇÃO DO BDEC

A Figura D.1 mostra a relação existente entre as estratégias de modelagem do Banco de Dados para uma aplicação baseada em SOFTBOARD e as estratégias de alimentação do BDEC.

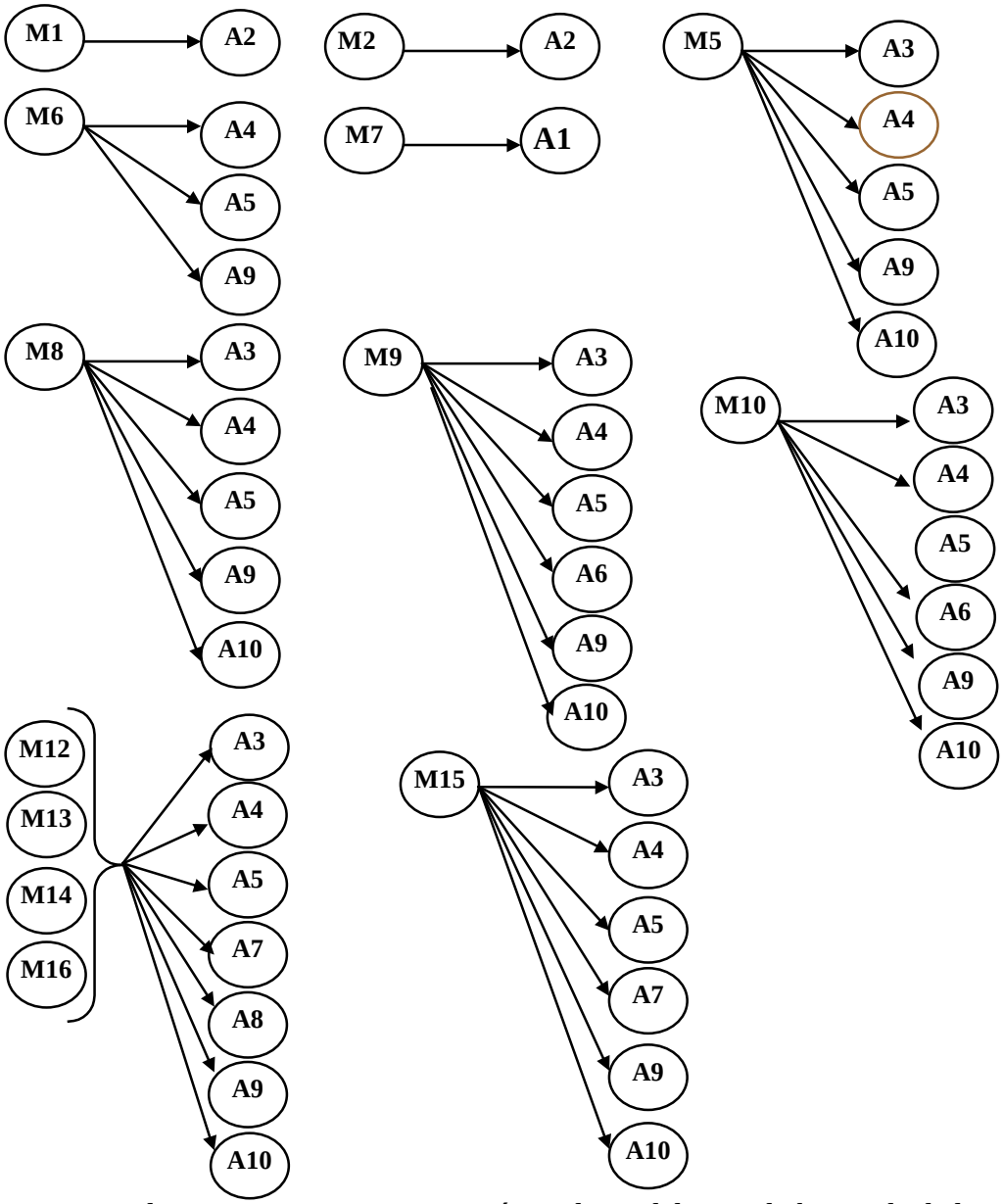


Fig. D.1 - Relacionamento entre as estratégias de modelagem do banco de dados e as estratégias de Alimentação do BDEC.

Como apresentado anteriormente nos capítulos 3 e 4, as estratégias de modelagem do banco de dados são identificadas por um “M” e uma numeração e as estratégis de alimentação do BDEC são identificadas por um “A” e uma numeração. As setas indicam as gerações, por exemplo, ao se utilizar a estratégia “M1” é gerado um cadastro no BDEC utilizando-se a estratégia “A2”.

APÊNDICE E

ESTUDO DE CASO - SISTEMA SIMPLIFICADO DE BIBLIOTECA

Este apêndice apresenta os resultados da aplicação das estratégias e dos padrões, propostos neste trabalho de pesquisa, para um Sistema Simplificado de Bibliotecas.

O Sistema Simplificado de Bibliotecas possui as seguintes facilidades:

- Cadastrar acervo;
- Cadastrar usuário;
- Registrar empréstimos;
- Cadastrar biblioteca;
- Registrar reservas;
- Listar usuários em atraso;
- Listar acervos emprestados por usuários;
- Listar acervos mais emprestados;
- Listar acervos mais reservados;
- Listar acervos nunca emprestados ou com data de empréstimo inferior a dois anos.
- Listar acervos reservados por usuários.

1. Componente Domínio do Problema:

A Figura a seguir apresenta o modelo do Componente Domínio do Problema do Sistema Simplificado de Biblioca, utilizando a notação da UML.

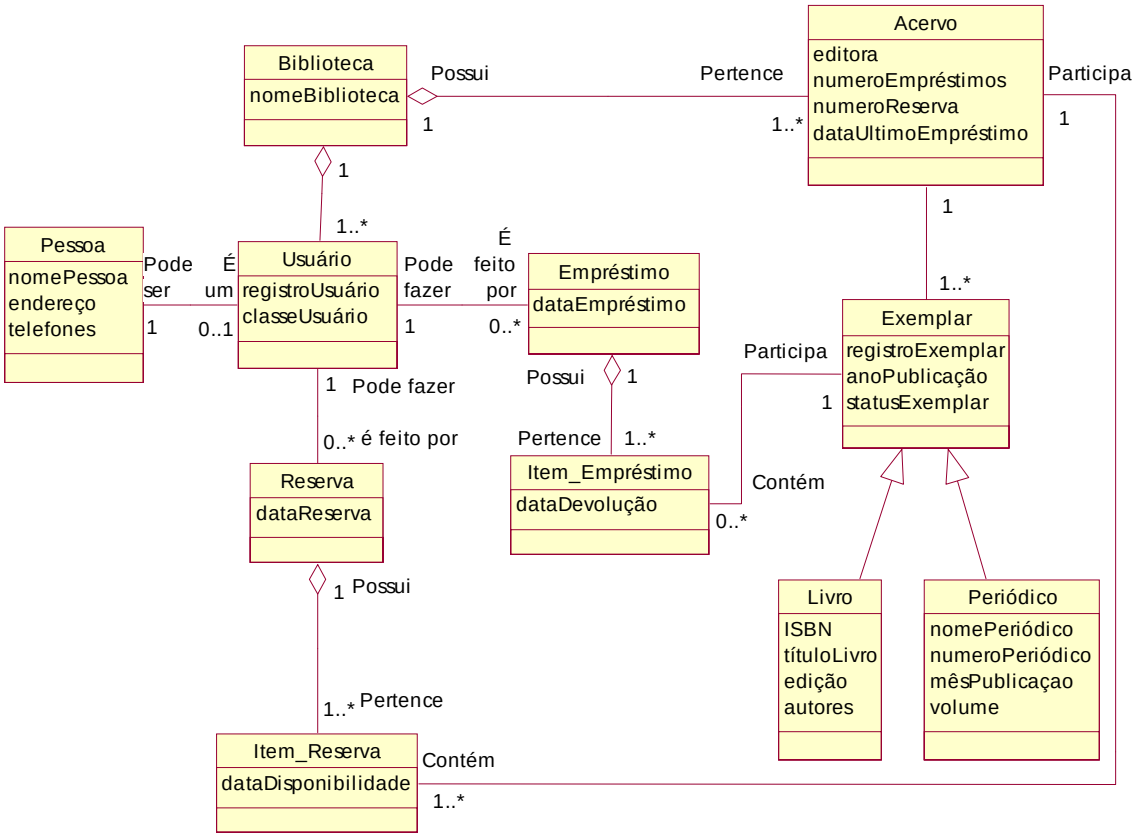


Fig. E.1 – Modelo do Componente Domínio do Problema do Sistema Simplificado de Biblioteca.

2. Utilização das estratégias e padrões para a modelagem do banco de dados da SOFTBOARD e das estratégias para a alimentação dos metadados do BDEC

- ✔ As estratégias para a alimentação do BDEC, serão utilizadas ao se identificar os elementos persistentes do sistema e, assim que for gerada cada relação no banco de dados.

- ✔ Para a alimentação dos metadados do BDEC, assume-se que as relações que possuem os dados do modelo orientado a objetos, já estejam cadastradas em suas devidas relações.

- ✔ Para a modelagem do banco de dados devem ser utilizadas, inicialmente, as estratégias de identificação das classes-&-objetos e atributos persistentes e a seguir, a de identificação dos padrões em que os objetos do Componente

Domínio do Problema se apresentam.

2.1 Utilização das estratégias de identificação

Estratégia #M1 // Identificar as Classes-&-Objetos Persistentes

Cadastro no BDEC:

Estratégia #A1 // Cadastrar Modelo Relacional

Relação: RelMODELOREL

IdModeloRel
ModSSB

Estratégia #A2 // Atualizar Persistência de Classes-&-Objetos e Atributos

Relação: RelCLASSEOBJETO

IdClasseObjeto	idComponente	Persistência	idRelação
Biblioteca	CDP	sim	
Pessoa	CDP	sim	
Usuário	CDP	sim	
Reserva	CDP	sim	
Item_Reserva	CDP	sim	
Empréstimo	CDP	sim	
Item_Empréstimo	CDP	sim	
Acervo	CDP	sim	
Exemplar	CDP	sim	
Livro	CDP	sim	
Periódico	CDP	sim	

Estratégia #M2 // Identificar Atributos Persistentes

Cadastro no BDEC:

Estratégia #A2 // Atualizar Persistência de Classes-&-Objetos e Atributos

Relação: RelATRIBUTO_OO

IdAtributoOO	idClasseObjeto	Persistência
NomeBiblioteca	Biblioteca	Sim
NomePessoa	Pessoa	Sim

continua

Relação: RelATRIBUTO_OO (conclusão)

Endereço	Pessoa	Sim
Telefones	Pessoa	Sim
RegistroUsuário	Usuário	Sim
ClasseUsuário	Usuário	sim
DataReserva	Reserva	sim
DataDisponibilidade	Item_Reserva	sim
DataEmpréstimo	Empréstimo	sim
DataDevolução	Item_Empréstimo	sim
Editora	Acervo	sim
NúmeroEmpréstimos	Acervo	sim
NúmeroReserva	Acervo	sim
DataUltimoEmpréstimo	Acervo	sim
RegistroExemplar	Exemplar	sim
anoPublicação	Exemplar	sim
statusExemplar	Exemplar	sim
ISBN	Livro	sim
títuloLivro	Livro	sim
edição	Livro	sim
autores	Livro	sim
nomePeriódico	Periódico	sim
númeroPeriódico	Periódico	sim
mêsPublicação	Periódico	sim
volume	Periódico	sim

- ✓ Ao identificarmos as classes-&-objetos persistentes do sistema e seus respectivos atributos persistentes montamos a tabela de seleção apresentada a seguir, incluindo ainda os identificadores do objeto (gerados pelo Componente Gerenciador de Dados da SOFTBOARD):

TABELA E.1 – SELEÇÃO DE OBJETOS E ATRIBUTOS PERSISTENTES DO SISTEMA E DENOMINAÇÃO DOS IDENTIFICADORES DOS OBJETOS

Classes-&-Objetos Persistentes	**Identificador**	**Atributos Persistentes**
Biblioteca	idBiblioteca	nomeBiblioteca
Pessoa	idPessoa	nomePessoa endereço telefones
Usuário	idUsuário	registroUsuário classe

continua

TABELA E.1 - conclusão

Reserva	idReserva	dataReserva
Item-Reserva	idItemReserva	dataDisponibilidade
Empréstimo	idEmpréstimo	dataEmpréstimo
Item_Empréstimo	IdItemEmpréstimo	dataDevolução
Acervo	idAcervo	númeroEmpréstimo editora númeroReserva dataUltimoEmpréstimo
Exemplar	idExemplar	registroExemplar anoPublicação statusExemplar
Livro	idLivro	títuloLivro ISBN edição autores
Periódico	idPeriódico	nomePeriódico númeroPeriódico, mêsPublicação, volume

Estratégia #M3 // Identificar os padrões em que os objetos do Componente Domínio do Problema se apresentam

- ✓ Ao identificarmos os padrões em que os objetos do Componente Domínio do Problema se apresentam, montamos a tabela de identificação apresentada a seguir.

TABELA E.2 – IDENTIFICAÇÃO DOS PADRÕES EM QUE AS ASSOCIAÇÕES DO CDP SE APRESENTAM.

Padrões identificados	**Associações**
Generalização-Especialização	Exemplar-(livro/ periódico)
Item-Item Específico	Acervo-Exemplar
Ator-Participante	Pessoa-Usuário
Participante-Transação	Usuário-Empréstimo
	Usuário-Reserva
Transação-Item de Linha de Transação	Empréstimo-Item_Empréstimo
	Reserva-Item_Reserva
Item de Linha de Transação-Item de Linha de Transação Subsequente	Item_Reserva-Acervo
	Item_Empréstimo-Exemplar

continua

TABELA E.2 - conclusão

Recipiente-Conteúdo	Biblioteca-Acervo
Grupo-Membro	Biblioteca-Usuário

- ✔ Reconhecidos os padrões, deve-se escolher uma associação a ser modelada e, através do padrão ao qual ela se identifica, aplicam-se as estratégias de representação relacionadas a eles.

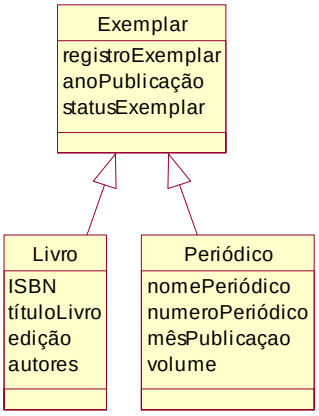
- ✔ Antes de iniciarmos a utilização das estratégias de representação, é importante observar que a estratégia de representação dos atributos persistentes deve ser utilizada, com exceção da estratégia de representação dos identificadores dos objetos, juntamente com todas as demais estratégias de representação. Tal estratégia somente será mostrada quando houver necessidade.

2.2 Utilização das estratégias de representação e dos padrões

- ✔ Para a aplicação das estratégias de representação, as associações foram agrupadas em vários casos, de acordo com os padrões em que se apresentam.

CASO 1:

Padrão #11 // Generalização-Especialização



- ✔ De acordo com o Padrão #11, identifica-se uma estrutura de generalização-especialização, portanto deve-se inicialmente escolher o tipo de representação

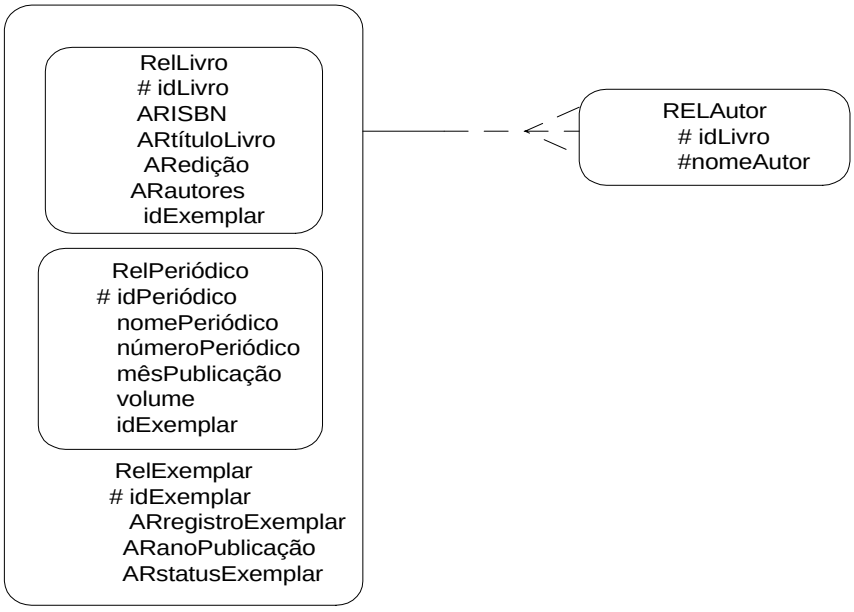
de herança que melhor se adequa ao sistema em questão.

Como não foram encontrados problemas de espaço e velocidade de acesso aos dados, optamos pela utilização da Estratégia #M9 // Representação de herança usando uma relação por classe-&-objetos da hierarquia, pois assim haverá um menor número de relacionamentos entre as relações que serão geradas.

- ✔ Ao utilizar-se a estratégia #M9, será criada uma relação para cada classe-&-objetos da hierarquia. O relacionamento entre elas, será estabelecido através da criação de uma coluna de chave embutida nas classes-&-objetos concretas da hierarquia, contendo a representação do identificador da classe ancestral. A chave primária de cada relação será a representação do identificador da própria classe-&-objetos que elas representam.

- ✔ Através da estratégia de representação dos atributos persistentes, encontrou-se a necessidade da criação de uma relação para representar o atributo “autores”, do tipo simples multivalorado, o qual pertence a classe-objetos “Livro”.

- ✔ Como neste caso o atributo possui uma baixa importância individual dentro do sistema, a nova relação será composta pelo próprio atributo e pela identificação da classe-&-objetos da qual ele originou. A chave primária será composta por ambas as colunas. Assim, obtêm-se as seguintes relações:



Cadastro no BDEC:

Estratégia #A3 // Cadastrar Relações Geradas por Classes-&-Objetos

Relação: RelRELAÇÃO

idRelação	idModeloRel
RELexemplar	ModRelSSB
RELlivro	ModRelSSB
RELperiódico	ModRelSSB

Relação: RelCLASSEOBJETO

idClasseObjeto	idComponente	Persistência	idRelação
Biblioteca	CDP	sim	
Pessoa	CDP	sim	
Usuário	CDP	sim	
Reserva	CDP	sim	
Item_Reserva	CDP	sim	
Empréstimo	CDP	sim	
Item_Empréstimo	CDP	sim	
Acervo	CDP	sim	
Exemplar	CDP	sim	RELexemplar
Livro	CDP	sim	RELlivro
Periódico	CDP	sim	RELperiódico

Estratégia #A4 // Cadastrar Atributos Relacionais

Relação: RelATRIBUTOREL

idAtributoRel
ARregistroExemplar
ARanoPublicação
ARstatusExemplar
ARISBN
ARtituloLivro
ARedição
ARnomePeriódico
ARnumeroPeriodico
ARmêsPublicação
ARvolume
ARnomeAutor

Relação: RelATRIBSOMONO

idAtribSmono	idAtributoOO	idAtributoRel
ASMOregistroExemplar	registroExemplar	ARregistroExemplar
ASMOanoPublicação	anoPublicação	ARanoPublicação
ASMOstatusExemplar	statusExemplar	ARstatusExemplar
ASMOISBN	ISBN	ARISBN
ASMOtituloLivro	tituloLivro	ARtituloLivro
ASMOedição	edição	ARedição
ASMOnomePeriódico	nomePeriódico	ARnomePeriódico
ASMOnumeroPeriodico	numeroPeriodico	ARnumeroPeriodico
ASMOmêsPublicação	mêsPublicação	ARmêsPublicação
ASMOvolume	volume	ARvolume

Relação: RelATRIBSMULTI

idAtribSmulti	idAtribMultivalor	idAtributoRel
ASMUautores	AMautores	ARnomeAutor

Relação: RelATRIBUTOREL-RELAÇÃO

idAtributoRel	idRelação
ARregistroExemplar	RELExemplar
ARanoPublicação	RELExemplar
ARstatusExemplar	RELExemplar
ARISBN	RELLivro
ARtituloLivro	RELLivro
ARedição	RELLivro
ARnomePeriódico	RELPeriódico
ARnumeroPeriódico	RELPeriódico
ARmêsPublicação	RELPeriódico
ARvolume	RELPeriódico
ARnomeAutor	RELAutor

Estratégia #A5 // Cadastrar Relações Geradas por Atributos Multivalorados

Relação: RelRELAÇÃO

idRelação	idModeloRel
RELExemplar	ModRelSSB
RELLivro	ModRelSSB
RELPeriódico	ModRelSSB
RELAutor	ModRelSSB

Relação: RelATRIBMULTIVALOR-RELAÇÃO

idAtribMultivalor	idRelação
AMautores	RELAutor

Estratégia #A6 // Cadastrar Chave Estrangeira Gerada para Mapear Estruturas de Generalização/Especialização

Relação: RelCHESTRANGEIRA

idChaveEstrangeira
idExemplar

Relação: RelGENESPEC-CHAVEESTRANGEIRA

idGenEspec	IdChaveEstrangeira
idGE01	idExemplar
idGE02	idExemplar

Relação: RelCHESTRANGEIRA-RELAÇÃO

idChaveEstrangeira	idRelação
idExemplar	RELLivro
idExemplar	RELPeriódico

Estratégia #A9 // Cadastrar Chaves Primárias Geradas por Atributos Multivalorados

Relação: RelCHAVEPRIMÁRIA

idChavePrimária	idRelação
idLivro	RELAutor
nomeAutor	RELAutor

Relação: RelATRIBMULTIVALOR-CHAVEPRIMÁRIA

idAtribMultivalor	IdChavePrimária
AMautores	IdLivro
AMautores	NomeAutor

Estratégia #A10 // Cadastrar Chaves Primárias

Relação:RelCHAVEPRIMÁRIA

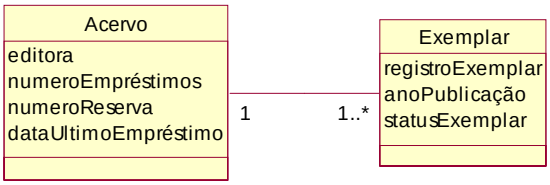
idChavePrimária	idRelação
idExemplar	RELExemplar
idLivro	RELLivro
idPeriódico	RELPeriódico

Relação: RelIDENTIFICADOROBJETO-CHAVEPRIMÁRIA

idIdentificadorObjeto	idChavePrimária
idExemplar	idExemplar
idLivro	idLivro
idPeriódico	idPeriódico

CASO #2:

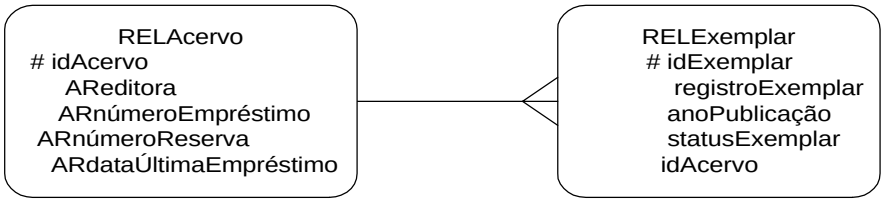
Padrão #10 // Item – Item Específico



- ✔ De acordo com o Padrão #10, identifica-se uma associação N:1. Será utilizada a estratégia #M10 // Avaliação das abordagens de representação de associação para avaliar a abordagem a ser empregada e a estratégia #M12 // Representação de associações N:1, para representar a associação.

- ✔ Neste caso deve ser utilizada a abordagem de chave embutida, pois ela gera maior eficiência e é a isto que se visa. A abordagem da relação distinta gera maior flexibilidade, o que não se faz necessário neste caso.

- ✔ Sendo assim, obtêm-se as seguintes relações:



Cadastro no BDEC:

Estratégia #A3 // Cadastrar Relações Geradas por Classes-&-Objetos

Relação: RelRELAÇÃO

idRelação	IdModeloRel
RelExemplar	ModRelSSB
RelLivro	ModRelSSB
RelPeriódico	ModRelSSB
RelAutor	ModRelSSB
RelAcervo	ModRelSSB

Relação: RelCLASSEOBJETO

idClasseObjeto	IdComponente	Persistência	idRelação
Biblioteca	CDP	sim	
Pessoa	CDP	sim	
Usuário	CDP	sim	
Reserva	CDP	sim	
Item_Reserva	CDP	sim	
Empréstimo	CDP	sim	
Item_Empréstimo	CDP	sim	
Acervo	CDP	sim	RelAcervo
Exemplar	CDP	sim	RelExemplar
Livro	CDP	sim	RelLivro
Periódico	CDP	sim	RelPeriódico

Estratégia #A4 // Cadastrar Atributos Relacionais

Relação: RelATRIBUTOREL

idAtributoRel
ARregistroExemplar
ARanoPublicação
ARstatusExemplar
ARISBN
ARtituloLivro
ARedição
ARnomePeriódico
ARnumeroPeriodico
ARmêsPublicação
ARvolume

continua

Relação: RelATRIBUTOREL (conclusão)

ARnomeAutor
AReditora
ARnumeroEmpréstimos
ARnumeroReserva
ARdataUltimoEmpréstimo

Relação: RelATRIBSMONO

idAtribSmono	idAtributoOO	idAtributoRel
ASMOregistroExemplar	registroExemplar	ARregistroExemplar
ASMOanoPublicação	AnoPublicação	ARanoPublicação
ASMOstatusExemplar	StatusExemplar	ARstatusExemplar
ASMOISBN	ISBN	ARISBN
ASMOtituloLivro	TituloLivro	ARtituloLivro
ASMOedição	edição	ARedição
ASMOnomePeriódico	nomePeriódico	ARnomePeriódico
ASMOnumeroPeriodico	numeroPeriodico	ARnumeroPeriodico
ASMOmêsPublicação	mêsPublicação	ARmêsPublicação
ASMOvolume	volume	ARvolume
ASMOeditora	editora	AReditora
ASMOnumeroEmpréstimos	numeroEmpréstimos	ARnumeroEmpréstimos
ASMOnumeroReserva	numeroReserva	ARnumeroReserva
ASMOdataUltimoEmpréstimo	dataUltimoEmpréstimo	ARdataUltimoEmpréstimo

Relação: RelATRIBUTOREL-RELAÇÃO

idAtributoRel	idRelação
ARregistroExemplar	RELExemplar
ARanoPublicação	RELExemplar
ARstatusExemplar	RELExemplar
ARISBN	RELLivro
ARtituloLivro	RELLivro
ARedição	RELLivro
ARnomePeriódico	RELPeriódico
ARnúmeroPeriódico	RELPeriódico
ARmêsPublicação	RELPeriódico
ARvolume	RELPeriódico
ARnomeAutor	RELAutor
AReditora	RELAcervo
ARnúmeroEmpréstimos	RELAcervo
ARnúmeroReserva	RELAcervo
ARdataUltimoEmpréstimo	RELAcervo

Estratégia #A8 // Cadastrar Chave Estrangeira Gerada para Mapear Conexões

Relação: RelCHESTRANGEIRA

idChaveEstrangeira
idExemplar
idAcervo

Relação: RelCHESTRANGEIRA-RELAÇÃO

idChaveEstrangeira	idRelação
idExemplar	RELLivro
idExemplar	RELPeriódico
idAcervo	RELExemplar

Relação: RelCONEXÕES-CHESTRANGEIRA

idConexões	idChaveEstrangeira
idCon09	idAcervo

Estratégia #A10 // Cadastrar Chaves Primárias

Relação:RelCHAVEPRIMÁRIA

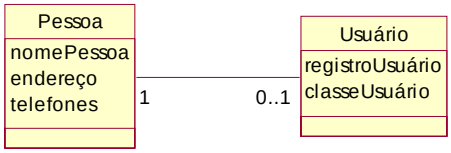
idChavePrimária	idRelação
idExemplar	RELExemplar
idLivro	RELLivro
idPeriódico	RELPeriódico
idAcervo	RELAcervo

Relação: RelIDENTIFICADOROBJETO-CHAVEPRIMÁRIA

idIdentificadorObjeto	idChavePrimária
idExemplar	idExemplar
idLivro	idLivro
idPeriódico	idPeriódico
idAcervo	idAcervo

CASO #3 :

Padrão #1 // Ator – Participante



- ✔ De acordo com o Padrão #1, identifica-se uma associação 0-1:1. Será utilizada a estratégia #M10 // Avaliação das abordagens de representação de associação, para avaliar a abordagem a ser emprega e a estratégia #M11 // Representação de associações 1:1, para representar a associação.

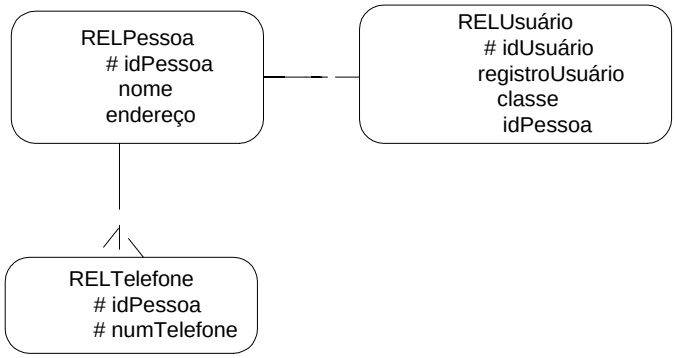
- ✔ Como há uma associação 0-1:1, não se deve utilizar a abordagem da classe embutida, pois se a sua utilização, para toda pessoa que não for usuário, haverá tuplas vazias e isto implica em desperdício de espaço.

- ✔ A abordagem da relação distinta também não é aconselhável neste caso, pois esta associação nunca poderá passar a ser N:N. Portanto, somente se estaria desperdiçando espaço e tempo de acesso.

- ✔ Assim, foi decidido o uso da abordagem de chave embutida. Como a associação é de 0-1:1, a relação na qual a chave será embutida será aquela que representa a classe-&-objetos com conectividade “1”, para que não existam tuplas vazias na relação.

- ✔ Através da estratégia de representação dos atributos persistentes, encontrou-se a necessidade da criação de uma relação para representar o atributo “telefones”, do tipo simples multivalorado, o qual pertence a classe-objetos “Pessoa”.

- ✔ Como neste caso o atributo possui uma baixa importância individual dentro do sistema, a nova relação será composta pelo próprio atributo e pela identificação da classe-&-objetos, da qual ele originou. A chave primária será composta por ambas as colunas. Assim, obtêm-se as seguintes relações:



Cadastro no BDEC:

Estratégia #A3 // Cadastrar Relações Geradas por Classes-&-Objetos

Relação: RelRELAÇÃO

idRelação	idModeloRel
RELExemplar	ModRelSSB
RELLivro	ModRelSSB
RELPeriódico	ModRelSSB
RELAutor	ModRelSSB
RELAcervo	ModRelSSB
RELPessoa	ModRelSSB
RELUsuário	ModRelSSB

Relação: RelCLASSEOBJETO

idClasseObjeto	idComponente	Persistência	idRelação
Biblioteca	CDP	sim	
Pessoa	CDP	sim	RELPessoa
Usuário	CDP	sim	RELUsuário
Reserva	CDP	sim	
Item_Reserva	CDP	sim	
Empréstimo	CDP	sim	
Item_Empréstimo	CDP	sim	
Acervo	CDP	sim	RELAcervo
Exemplar	CDP	sim	RELExemplar
Livro	CDP	sim	RELLivro
Periódico	CDP	sim	RELPeriódico

Estratégia #A4 // Cadastrar Atributos Relacionais

Relação: RelATRIBUTOREL

idAtributoRel
ARregistroExemplar
ARanoPublicação
ARstatusExemplar
ARISBN
ARtituloLivro
ARedição
ARnomePeriódico
ARnumeroPeriodico
ARmêsPublicação
ARvolume
ARnomeAutor
AReditora
ARnumeroEmpréstimos
ARnumeroReserva
ARdataUltimoEmpréstimo
ARnomePessoa
ARendereco
ARnúmeroTelefone
ARregistroUsuário
ARclasseUsuário

Relação: RelATRIBSMONO

idAtribSmono	idAtributoOO	idAtributoRel
ASMOregistroExemplar	registroExemplar	ARregistroExemplar
ASMOanoPublicação	anoPublicação	ARanoPublicação
ASMOstatusExemplar	statusExemplar	ARstatusExemplar
ASMOISBN	ISBN	ARISBN
ASMOtituloLivro	tituloLivro	ARtituloLivro
ASMOedição	edição	ARedição
ASMOnomePeriódico	nomePeriódico	ARnomePeriódico
ASMOnumeroPeriodico	numeroPeriodico	ARnumeroPeriodico
ASMOmêsPublicação	mêsPublicação	ARmêsPublicação
ASMOvolume	volume	ARvolume
ASMOeditora	editora	AReditora
ASMOnumeroEmpréstimos	numeroEmpréstimos	ARnumeroEmpréstimos
ASMOnumeroReserva	numeroReserva	ARnumeroReserva
ASMOdataUltimoEmpréstimo	dataUltimoEmpréstimo	ARdataUltimoEmpréstimo
ASMOnomePessoa	nomePessoa	ARnomePessoa
ASMOregistroUsuário	registroUsuário	ARregistroUsuário
ASMOclasseUsuário	classeUsuário	ARclasseUsuário

Relação: RelATRIBCMONO

idAtribCmono	IdAtributoRel
ACMOendereço	ARendereço

Relação: RelATRIBSMULTI

idAtribSmulti	IdAtribMultivalor	idAtributoRel
ASMUautores	AMautores	ARnomeAutor
ASMUtelefones	AMtelefones	ARnúmeroTelefone

Relação: RelATRIBUTOREL-RELAÇÃO

idAtributoRel	idRelação
ArregistroExemplar	RELExemplar
AranoPublicação	RELExemplar
ArstatusExemplar	RELExemplar
ARISBN	RELLivro
ArtituloLivro	RELLivro
ARedição	RELLivro
ARnomePeriódico	RELPeriódico
ARnúmeroPeriódico	RELPeriódico
ARmêsPublicação	RELPeriódico
ARvolume	RELPeriódico
ARnomeAutor	RELAutor
AReditora	RELAcervo
ARnúmeroEmpréstimo	RELAcervo
ARnúmeroReserva	RELAcervo
ARdataUltimoEmpréstimo	RELAcervo
ARnomePessoa	RELPessoa
ARendereço	RELPessoa
ARnúmerotelefone	RELTelefone
ARregistroUsuário	RELUsuário
ARclasseUsuário	RELclasseUsuário

Estratégia #A5 // Cadastrar Relações Geradas por Atributos Multivalorados

Relação: RelRELAÇÃO

idRelação	idModeloRel
RELExemplar	ModRelSSB
RELLivro	ModRelSSB
RELPeriódico	ModRelSSB

continua

Relação: RelRELAÇÃO (conclusão)

RELAutor	ModRelSSB
RELAcervo	ModRelSSB
RELPessoa	ModRelSSB
RELUsuário	ModRelSSB
RELTelefone	ModRelSSB

Relação: RelATRIBMULTIVALOR-RELAÇÃO

idAtribMultivalor	idRelação
AMautores	RELAutor
AMtelefones	RELTelefone

Estratégia #A8 // Cadastrar Chave Estrangeira Gerada para Mapear Conexões

Relação: RelCHESTRANGEIRA

idChaveEstrangeira
idExemplar
idAcervo
idPessoa

Relação: RelCHESTRANGEIRA-RELAÇÃO

idChaveEstrangeira	idRelação
idExemplar	RELLivro
idExemplar	RELPeriódico
idAcervo	RELExemplar
idPessoa	RELUsuário

Relação: RelCONEXÕES-CHESTRANGEIRA

idConexões	idChaveEstrangeira
idCon09	idAcervo
idCon02	idPessoa

Estratégia #A9 // Cadastrar Chaves Primárias Geradas por Atributos Multivalorados

Relação: RelCHAVEPRIMÁRIA

idChavePrimária	idRelação
idLivro	RELAutor
nomeAutores	RELAutor
idPessoa	RELTelefone
numeroTelefone	RELTelefone

Relação: RelATRIBMULTIVALOR-CHAVEPRIMÁRIA

idAtribMultivalor	idChavePrimária
AMautores	idLivro
AMautores	nomeAutor
AMtelefones	idPessoa
AMtelefones	númeroTelfone

Estratégia #A10 // Cadastrar Chaves Primárias

Relação:RelCHAVEPRIMÁRIA

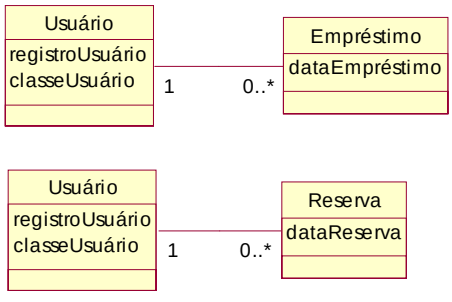
idChavePrimária	idRelação
idExemplar	RELExemplar
idLivro	RELLivro
idPeriódico	RELPeriódico
idAcervo	RELAcervo
idPessoa	RELPessoa
idUsuário	RELUsuário

Relação: RelIDENTIFICADOROBJETO-CHAVEPRIMÁRIA

idIdentificadorObjeto	idChavePrimária
idExemplar	idExemplar
idLivro	idLivro
idPeriódico	idPeriódico
idAcervo	idAcervo
idPessoa	idPessoa
idUsuário	idUsuário

CASO #4:

Padrão #2 // Participante - Transação

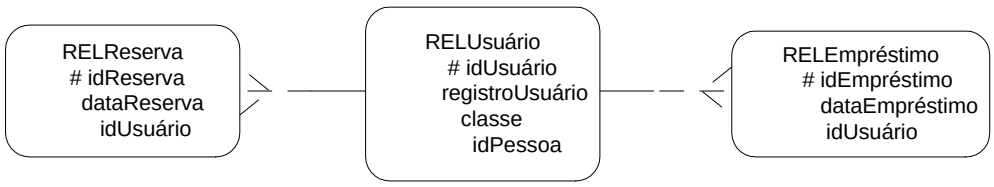


- ✓ De acordo com o Padrão #2, identifica-se uma associação N:1 nos dois casos. Será utilizada a estratégia #M10 para avaliar a abordagem a ser empregada e a estratégia #M12, para representar as associações.

Estratégia #M10 // Avaliação das abordagens de representação de associação

Estratégia #M12 // Representação de associações N:1

- ✓ A abordagem utilizada para as duas associações será a mesma, já que elas possuem as mesmas características.
- ✓ Neste caso, deve-se utilizar a abordagem de chave embutida, pois ela gera maior eficiência e é isto a que se visa. A abordagem da relação distinta pode ser descartada, pois neste caso, não se necessita de flexibilidade.
- ✓ Sendo assim, obtêm-se as seguintes relações:



Cadastro no BDEC:

Estratégia #A3 // Cadastrar Relações Geradas por Classes-&-Objetos

Relação: RelRELAÇÃO

idRelação	idModeloRel
RELExemplar	ModRelSSB
RELLivro	ModRelSSB
RELPeriódico	ModRelSSB
RELAutor	ModRelSSB
RELAcervo	ModRelSSB
RELPessoa	ModRelSSB
RELUsuário	ModRelSSB
RELTelefone	ModRelSSB
RELReserva	ModRelSSB
RELEmpréstimo	ModRelSSB

Relação: RelCLASSEOBJETO

idClasseObjeto	idComponente	Persistência	idRelação
Biblioteca	CDP	sim	
Pessoa	CDP	sim	RELPessoa
Usuário	CDP	sim	RELUsuário
Reserva	CDP	sim	RELReserva
Item_Reserva	CDP	sim	
Empréstimo	CDP	sim	RELEmpréstimo
Item_Empréstimo	CDP	sim	
Acervo	CDP	sim	RELAcervo
Exemplar	CDP	sim	RELExemplar
Livro	CDP	sim	RELLivro
Periódico	CDP	sim	Periódico

Estratégia #A4 // Cadastrar Atributos Relacionais

Relação: RelATRIBUTOREL

idAtributoRel
ARregistroExemplar
ARanoPublicação
ARstatusExemplar
ARISBN
ARtituloLivro
ARedição
ARnomePeriódico
ARnumeroPeriodico
ARmêsPublicação
ARvolume
ARnomeAutor
AReditora
ARnumeroEmpréstimos
ARnumeroReserva
ARdataUltimoEmpréstimo
ARnomePessoa
ARendereco
ARnúmeroTelefone
ARregistroUsuário
ARclasseUsuário
ARdataReserva
ARdataEmpréstimo

Relação: RelATRIBCMONO

idAtribCmono	idAtributoRel
ACMOendereco	ARendereço
ACMOdataReserva	ARdataReserva
ACMOdataEmpréstimo	ARdataEmpréstimo

Relação: RelATRIBUTOREL-RELAÇÃO

idAtributoRel	idRelação
ARregistroExemplar	RELExemplar
ARanoPublicação	RELExemplar
ARstatusExemplar	RELExemplar
ARISBN	RELLivro
ARtituloLivro	RELLivro
ARedição	RELLivro
ARnomePeriódico	RELPeriódico
ARnúmeroPeriódico	RELPeriódico
ARmêsPublicação	RELPeriódico
ARvolume	RELPeriódico
ARnomeAutor	RELAutor
AReditora	RELAcervo
ARnúmeroEmpréstimos	RELAcervo
ARnúmeroReserva	RELAcervo
ARdataUltimoEmpréstimo	RELAcervo
ARnomePessoa	RELPessoa
ARendereço	RELPessoa
ARnúmerotelefone	RELTelefone
ARregistroUsuário	RELUsuário
ARclasseUsuário	RELUsuário
ARdataReserva	RELReserva
ARdataEmpréstimo	RELEmpréstimo

Estratégia #A8 // Cadastrar Chave Estrangeira Gerada para Mapear Conexões

Relação: RelCHESTRANGEIRA

idChaveEstrangeira
idExemplar
idAcervo
idPessoa
idUsuario
idReserva
idEmpréstimo

Relação: RelCHESTRANGEIRA-RELAÇÃO

idChaveEstrangeira	idRelação
idExemplar	RELLivro
idExemplar	RELPeriódico
idAcervo	RELExemplar
idPessoa	RELUsuário
idUsuário	RELReserva
idUsuário	RELEmpréstimo

Relação: RelCONEXÕES-CHESTRANGEIRA

idConexões	idChaveEstrangeira
idCon09	idAcervo
idCon02	idPessoa
idCon05	idUsuário
idCon04	idUsuário

Estratégia #A10 // Cadastrar Chaves Primárias

Relação:RelCHAVEPRIMÁRIA

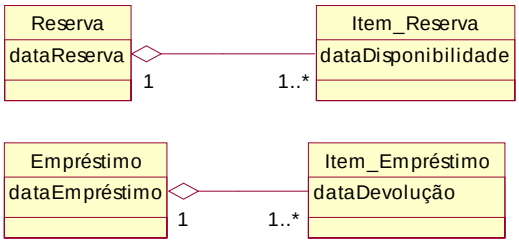
idChavePrimária	idRelação
idExemplar	RELExemplar
idLivro	RELLivro
idPeriódico	RELPeriódico
idAcervo	RELAcervo
idPessoa	RELPessoa
idUsuário	RELUsuário
idReserva	RELReserva
idEmpréstimo	RELEmpréstimo

Relação: RelIDENTIFICADOROBJETO-CHAVEPRIMÁRIA

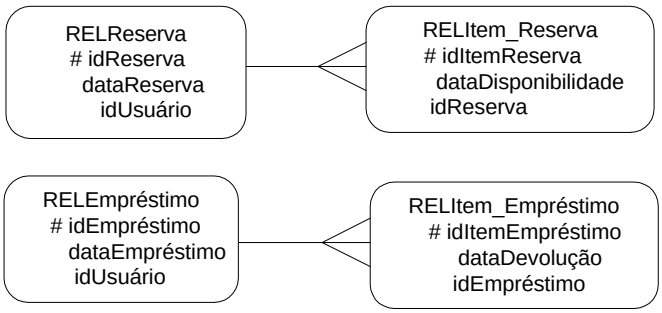
idIdentificadorObjeto	idChavePrimária
idExemplar	idExemplar
idLivro	idLivro
idPeriódico	idPeriódico
idAcervo	idAcervo
idPessoa	idPessoa
idUsuário	idUsuário
idReserva	idReserva
idEmpréstimo	idEmpréstimo

CASO #5:

Padrão #5 // Transação - Item de Linha de Transação



- ✔ De acordo com o Padrão #5, identifica-se uma associação todo-parte nos dois casos. Será utilizada a estratégia #M13 // Representação de associações todo-parte, para representar as associações.
- ✔ Como a conectividade das associações é 1:1-N, será criada uma relação para cada objeto, associando-as através de uma chave estrangeira embutida na relação que representa a "classe-&-objeto possuída", contendo a representação do identificador da "classe-&-objeto proprietária" da associação.
- ✔ Portanto, para a primeira associação, tem-se a chave primária da relação "Reserva" embutida na relação "Item_Reserva" e para a segunda associação, tem-se a chave primária da relação "Empréstimo" embutida na relação "Item_Empréstimo", obtendo-se as seguintes relações:



Cadastro no BDEC:

Estratégia #A3 // Cadastrar Relações Geradas por Classes-&-Objetos

Relação: RelRELAÇÃO

idRelação	idModeloRel
RELExemplar	ModRelSSB
RELLivro	ModRelSSB
RELPeriódico	ModRelSSB
RELAutor	ModRelSSB
RELAcervo	ModRelSSB
RELPessoa	ModRelSSB
RELUsuário	ModRelSSB
RELTelefone	ModRelSSB
RELReserva	ModRelSSB
RELEmpréstimo	ModRelSSB
RELItem_Reserva	ModRelSSB
RELItem_Empréstimo	ModRelSSB

Relação: RelCLASSEOBJETO

idClasseObjeto	idComponente	Persistência	idRelação
Biblioteca	CDP	sim	
Pessoa	CDP	sim	RELPessoa
Usuário	CDP	sim	RELUsuário
Reserva	CDP	sim	RELReserva
Item_Reserva	CDP	sim	RELItem_Reserva
Empréstimo	CDP	sim	RELEmpréstimo
Item_Empréstimo	CDP	sim	RELItem_Empréstimo
Acervo	CDP	sim	RELAcervo
Exemplar	CDP	sim	RELExemplar
Livro	CDP	sim	RELLivro
Periódico	CDP	sim	RELPeriódico

Estratégia #A4 // Cadastrar Atributos Relacionais

Relação: RelATRIBUTOREL

idAtributoRel
ARregistroExemplar
ARanoPublicação
ARstatusExemplar
ARISBN
ARtituloLivro
ARedição
ARnomePeriódico

continua

Relação: RelATRIBUTOREL (conclusão)

ARnumeroPeriodico
ARmêsPublicação
ARvolume
ARnomeAutor
AReditora
ARnumeroEmpréstimos
ARnumeroReserva
ARdataUltimoEmpréstimo
ARnomePessoa
ARendereco
ARnúmeroTelefone
ARregistroUsuário
ARclasseUsuário
ARdataReserva
ARdataEmpréstimo
ARdataDisponibilidade
ARdataDevolução

Relação: RelATRIBCMONO

idAtribCmono	idAtributoRel
CMONOendereco	ARendereço
CMONOdataReserva	ARdataReserva
CMONOdataEmpréstimo	ARdataEmpréstimo
CMONOdataDisponibilidade	ARdataDisponibilidade
CMONOdataDevolução	ARdataDevolução

Relação: RelATRIBUTOREL-RELAÇÃO

idAtributoRel	idRelação
ARregistroExemplar	RELExemplar
ARanoPublicação	RELExemplar
ARstatusExemplar	RELExemplar
ARISBN	RELLivro
ARtituloLivro	RELLivro
ARedição	RELLivro
ARnomePeriódico	RELPeriódico
ARnúmeroPeriódico	RELPeriódico
ARmêsPublicação	RELPeriódico
ARvolume	RELPeriódico
ARnomeAutor	RELAutor

continua

Relação: RelATRIBUTOREL-RELAÇÃO (conclusão)

AReditora	RELAcervo
ARnúmeroEmpréstimos	RELAcervo
ARnúmeroReserva	RELAcervo
ARdataUltimoEmpréstimo	RELAcervo
ARnomePessoa	RELPessoa
ARendereço	RELPessoa
ARnúmerotelefone	RELTelefone
ARregistroUsuário	RELUsuário
ARclasseUsuário	RELUsuário
ARdataReserva	RELReserva
ARdataEmpréstimo	RELEmpréstimo
ARdataDisponibilidade	RELReserva
ARdataDevolução	RELEmpréstimo

Estratégia #A8 // Cadastrar Chave Estrangeira para Gerada para Mapear Conexões

Relação: RelCHESTRANGEIRA

idChaveEstrangeira
idExemplar
idAcervo
idPessoa
idUsuario
idReserva
idEmpréstimo

Relação: RelCHESTRANGEIRA-RELAÇÃO

idChaveEstrangeira	**idRelação**
idExemplar	RELLivro
idExemplar	RELPeriódico
idAcervo	RELExemplar
idPessoa	RELUsuário
idUsuário	RELReserva
idUsuário	RELEmpréstimo
idReserva	RELItem_Reserva
idEmpréstimo	RELItem_Empréstimo

Relação: RelCONEXÕES-CHESTRANGEIRA

idConexões	idChaveEstrangeira
idCon09	idAcervo
idCon02	idPessoa
idCon05	idUsuário
idCon04	idUsuário
idCon06	idReserva
idCon07	idEmpréstimo

Estratégia #A10 // Cadastrar Chaves Primárias

Relação:RelCHAVEPRIMÁRIA

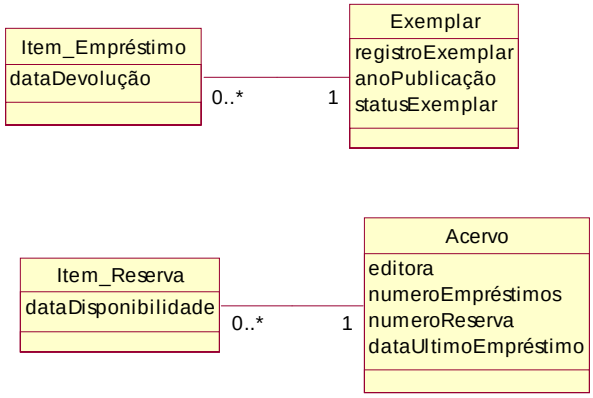
idChavePrimária	idRelação
idExemplar	RELExemplar
idLivro	RELLivro
idPeriódico	RELPeriódico
idAcervo	RELAcervo
idPessoa	RELPessoa
idUsuário	RELUsuário
idReserva	RELReserva
idEmpréstimo	RELEmpréstimo
idItemReserva	RELReserva
idItemEmpréstimo	RELEmpréstimo

Relação: RelIDENTIFICADOROBJETO-CHAVEPRIMÁRIA

idIdentificadorObjeto	idChavePrimária
idExemplar	idExemplar
idLivro	idLivro
idPeriódico	idPeriódico
idAcervo	idAcervo
idPessoa	idPessoa
idUsuário	idUsuário
idReserva	idReserva
idEmpréstimo	idEmpréstimo
idReserva	idItemReserva
idEmpréstimo	idItemEmpréstimo

CASO #6:

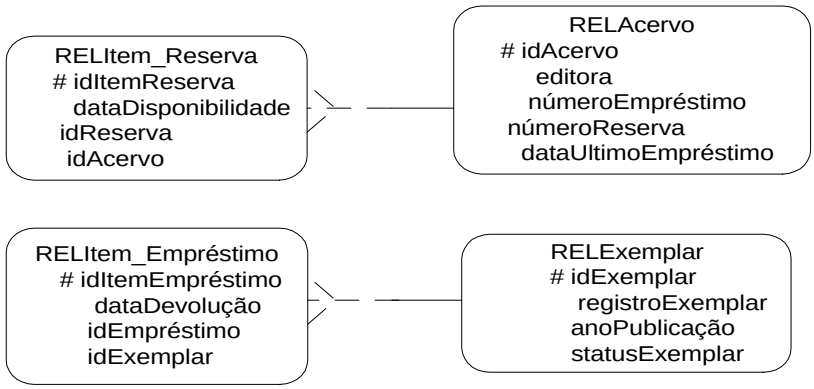
Padrão #7 // Item de Linha de Transação – Item de Linha de Transação Subsequente



- ✔ De acordo com o Padrão #7, identifica-se uma associação N:1 nos dois casos. Será utilizada a estratégia #M10 // Avaliação das abordagens de representação de associação para avaliar a abordagem a ser empregada e a estratégia #M12 // Representação de associações N:1, para representar as associações.

- ✔ A abordagem utilizada para as duas associações será a mesma, já que elas possuem as mesmas características.

- ✔ Neste caso deve-se utilizar a abordagem de chave embutida, pois ela gera maior eficiência e é isto a que se visa. A abordagem da relação distinta gera maior flexibilidade, o que não se faz necessário neste caso. Sendo assim, obtêm-se as seguintes relações:



Cadastro no BDEC:

Neste caso, as relações referentes as estratégias #A3, #A4 e #A10, não se alteram, pois estas relações, assim como seus atributos e suas chaves primárias já se encontram cadastrados.

Estratégia #A8 // Cadastrar Chave Estrangeira para Gerada para Mapear Conexões

A relação “Chave_Estrangeira”, neste caso, também não se altera em relação a anterior.

Relação: RelCHESTRANGEIRA-RELAÇÃO

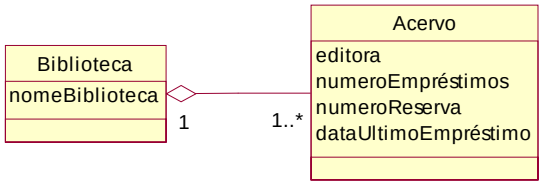
idChaveEstrangeira	idRelação
IdExemplar	RELLivro
IdExemplar	RELPeriódico
IdAcervo	RELExemplar
IdPessoa	RELUsuário
IdUsuário	RELReserva
idUsuário	RELEmpréstimo
idReserva	RELItem_Reserva
idEmpréstimo	RELItem_Empréstimo
idExemplar	RELItem_Empréstimo
idAcervo	RELItem_Reserva

Relação: RelCONEXÕES-CHESTRANGEIRA

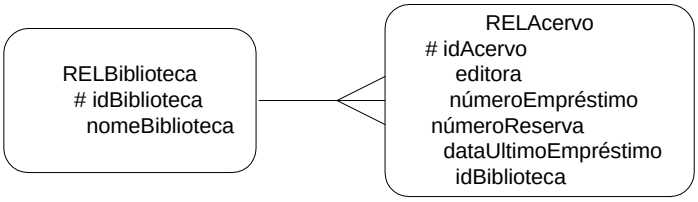
idConexões	idChaveEstrangeira
idCon09	idAcervo
idCon02	idPessoa
idCon05	idUsuário
idCon04	idUsuário
idCon06	idReserva
idCon07	idEmpréstimo
idcon08	idExemplar
idCon12	idReserva

CASO #7:

Padrão #12 // Recipiente - Conteúdo



- ✔ De acordo com o Padrão #12, identifica-se uma associação todo-parte. Será utilizada a estratégia #M13 // Representação de associações todo-parte, para representar a associação.
- ✔ Como a conectividade da associação é 1:1-N, será criada uma relação para cada objeto, associando-as através de uma chave estrangeira embutida na relação que representa a "classe-&-objeto possuída", relativa a relação que representa a "classe-&-objeto proprietária" da associação.
- ✔ Portanto, será utilizada a chave primária da relação "Biblioteca", embutida na relação "Acervo", obtendo-se assim as seguintes relações:



Cadastro no BDEC:

Estratégia #A3 // Cadastrar Relações Geradas por Classes-&-Objetos

Relação: RelRELAÇÃO

idRelação	idModeloRel
RELExemplar	ModRelSSB
RELLivro	ModRelSSB
RELPeriódico	ModRelSSB
RELAutor	ModRelSSB
RELAcervo	ModRelSSB
RELPessoa	ModRelSSB
RELUsuário	ModRelSSB
RELTelefone	ModRelSSB
RELReserva	ModRelSSB
RELEmpréstimo	ModRelSSB
RELItem_Reserva	ModRelSSB
RELItem_Empréstimo	ModRelSSB
RELBiblioteca	ModRelSSB

Relação: RelCLASSEOBJETO

idClasseObjeto	IdComponente	Persistência	idRelação
Biblioteca	CDP	sim	RELBiblioteca
Pessoa	CDP	sim	RELPessoa
Usuário	CDP	sim	RELUsuário
Reserva	CDP	sim	RELReserva
Item_Reserva	CDP	sim	RELItem_Reserva
Empréstimo	CDP	sim	RELEmpréstimo
Item_Empréstimo	CDP	sim	RELItem_Empréstimo
Acervo	CDP	sim	RELAcervo
Exemplar	CDP	sim	RELExemplar
Livro	CDP	sim	RELLivro
Periódico	CDP	sim	RELPeriódico

Estratégia #A4 // Cadastrar Atributos Relacionais

Relação: RelATRIBUTOREL

idAtributoRel
ARregistroExemplar
ARanoPublicação
ARstatusExemplar
ARISBN
ARtituloLivro
ARedição
ARnomePeriódico
ARnumeroPeriodico
ARmêsPublicação
ARvolume
ARnomeAutor
AReditora
ARnumeroEmpréstimos
ARnumeroReserva
ARdataUltimoEmpréstimo
ARnomePessoa
ARendereco
ARnúmeroTelefone
ARregistroUsuário
ARclasseUsuário
ARdataReserva
ARdataEmpréstimo
ARdataDisponibilidade
ARdataDevolução
ARnomeBiblioteca

Relação: RelATRIBSMONO

idAtribSmono	idAtributoOO	idAtributoRel
SMONOregistroExemplar	registroExemplar	ARregistroExemplar
SMONOanoPublicação	anoPublicação	ARanoPublicação
SMONOstatusExemplar	statusExemplar	ARstatusExemplar
SMONOISBN	ISBN	ARISBN
SMONOtituloLivro	tituloLivro	ARtituloLivro
SMONOedição	edição	ARedição
SMONOnomePeriódico	nomePeriódico	ARnomePeriódico
SMONOnumeroPeriodico	numeroPeriodico	ARnumeroPeriodico
SMONOmêsPublicação	mêsPublicação	ARmêsPublicação
SMONOvolume	volume	ARvolume
SMONOeditora	editora	AReditora
SMONOnumeroEmpréstimos	numeroEmpréstimos	ARnumeroEmpréstimos
SMONOnumeroReserva	numeroReserva	ARnumeroReserva
SMONOdataUltimoEmpréstimo	dataUltimoEmpréstimo	ARdataUltimoEmpréstimo
SMONOnomePessoa	nomePessoa	ARnomePessoa
SMONOregistroUsuário	registroUsuário	ARregistroUsuário
SMONOclasseUsuário	classeUsuário	ARclasseUsuário
SMONOnomeBiblioteca	nomeBiblioteca	ARnomeBiblioteca

Relação: RelATRIBUTOREL-RELAÇÃO

idAtributoRel	idRelação
ARregistroExemplar	RELExemplar
ARanoPublicação	RELExemplar
ARstatusExemplar	RELExemplar
ARISBN	RELLivro
ARtituloLivro	RELLivro
ARedição	RELLivro
ARnomePeriódico	RELPeriódico
ARnúmeroPeriódico	RELPeriódico
ARmêsPublicação	RELPeriódico
ARvolume	RELPeriódico
ARnomeAutor	RELAutor
AReditora	RELAcervo
ARnúmeroEmpréstimos	RELAcervo
ARnúmeroReserva	RELAcervo
ARdataUltimoEmpréstimo	RELAcervo
ARnomePessoa	RELPessoa
ARendereço	RELPessoa
ARnúmerotelefone	RELTelefone

continua

Relação: RelATRIBUTOREL-RELAÇÃO (conclusão)

ARregistroUsuário	RELUsuário
ARclasseUsuário	RELUsuário
ARdataReserva	RELReserva
ARdataEmpréstimo	RELEmpréstimo
ARdataDisponibilidade	RELReserva
ARdataDevolução	RELEmpréstimo
ARnomeBiblioteca	RELBiblioteca

Estratégia #A8 // Cadastrar Chave Estrangeira para Gerada para Mapear Conexões

Relação: RelCHESTRANGEIRA

idChaveEstrangeira
idExemplar
idAcervo
idPessoa
idUsuario
idReserva
idEmpréstimo
idBiblioteca

Relação: RelCHESTRANGEIRA-RELAÇÃO

idChaveEstrangeira	idRelação
idExemplar	RELLivro
idExemplar	RELPeriódico
idAcervo	RELExemplar
idPessoa	RELUsuário
idUsuário	RELReserva
idUsuário	RELEmpréstimo
idReserva	RELItem_Reserva
idEmpréstimo	RELItem_Empréstimo
idExemplar	RELItem_Empréstimo
idAcervo	RELItem_Reserva
idBiblioteca	RELAcevo

Relação: RelCONEXÕES-CHESTRANGEIRA

idConexões	idChaveEstrangeira
idCon09	idAcervo
idCon02	idPessoa
idCon05	idUsuário
idCon04	idUsuário
idCon06	idReserva
idCon07	idEmpréstimo
idcon08	idExemplar
idCon12	idReserva
idCon03	idBiblioteca

Estratégia #A10 // Cadastrar Chaves Primárias

Relação:RelCHAVEPRIMÁRIA

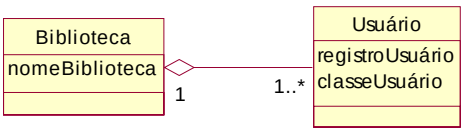
idChavePrimária	idRelação
idExemplar	RELExemplar
idLivro	RELLivro
idPeriódico	RELPeriódico
idAcervo	RELAcervo
idPessoa	RELPessoa
idUsuário	RELUsuário
idReserva	RELReserva
idEmpréstimo	RELEmpréstimo
idItemReserva	RELReserva
idItemEmpréstimo	RELEmpréstimo
idBiblioteca	RELBiblioteca

Relação: RelIDENTIFICADOROBJETO-CHAVEPRIMÁRIA

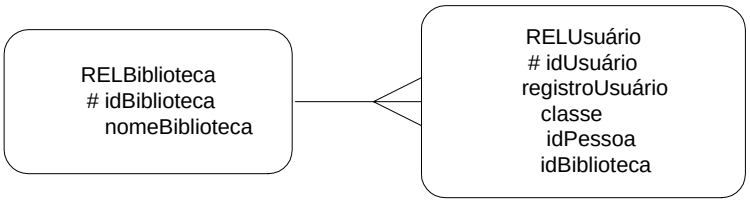
idRelação	idChavePrimária
idExemplar	idExemplar
idLivro	idLivro
idPeriódico	idPeriódico
idAcervo	idAcervo
idPessoa	idPessoa
idUsuário	idUsuário
idReserva	idReserva
idEmpréstimo	idEmpréstimo
idReserva	idItemReserva
idEmpréstimo	idItemEmpréstimo
idBiblioteca	idBiblioteca

CASO #8

Padrão #14 // Grupo-Membro



- ✔ De acordo com o Padrão #14, identifica-se uma associação todo-parte. Será utilzada a estratégia #M13 // Representação de associações todo-parte, para representar a associação.
- ✔ Como a conectividade da associação é 1:1-N, será criada uma relação para cada objeto associando-as através de uma chave estrangeira embutida na relação que representa a "classe-&-objeto possuída", contendo a representação do identificador da "classe-&-objeto proprietária" da associação.
- ✔ Portanto, será utilizada a chave primária da relação "Biblioteca", embutida na relação "Usuário", obtendo-se assim as seguintes relações:



Cadastro no BDEC:

Neste caso, as relações referentes as estratégias #A3, #A4 e #A10, não se alteram, pois estas relações assim como seus atributos e suas chaves primárias, já se encontram cadastrados.

Estratégia #A8 // Cadastrar Chave Estrangeira Gerada para Mapear Conexões

A relação Chave_Estrangeira, neste caso, também não se altera em relação a anterior.

Relação: RelCHESTRANGEIRA-RELAÇÃO

idChaveEstrangeira	idRelação
idExemplar	RELLivro
idExemplar	RELPeriódico
idAcervo	RELExemplar
idPessoa	RELUsuário
idUsuário	RELReserva
idUsuário	RELEmpréstimo
idReserva	RELItem_Reserva
idEmpréstimo	RELItem_Empréstimo
idExemplar	RELItem_Empréstimo
idAcervo	RELItem_Reserva
idBiblioteca	RELVAcevo
idBiblioteca	RELUsuário

Relação: RelCONEXÕES-CHESTRANGEIRA

idConexões	idChaveEstrangeira
idCon09	idAcervo
idCon02	idPessoa
idCon05	idUsuário
idCon04	idUsuário
idCon06	idReserva
idCon07	idEmpréstimo
idcon08	idExemplar
idCon12	idReserva
idCon03	idBiblioteca
idCon02	idBiblioteca

Utilização da estratégia de Normalização:

2.3 Estratégia #M16 // Normalizar de acordo com a BCNF (Boyce-Codd Normal Form)

- ✓ Através da utilização da estratégia #M16, verifica-se que todas as relações se encontram normalizadas, pois todo determinante de cada relação gerada é chave candidata.

A Figura B.2, apresenta o modelo relacional de dados gerado para o Sistema Simplificado de Biblioteca.

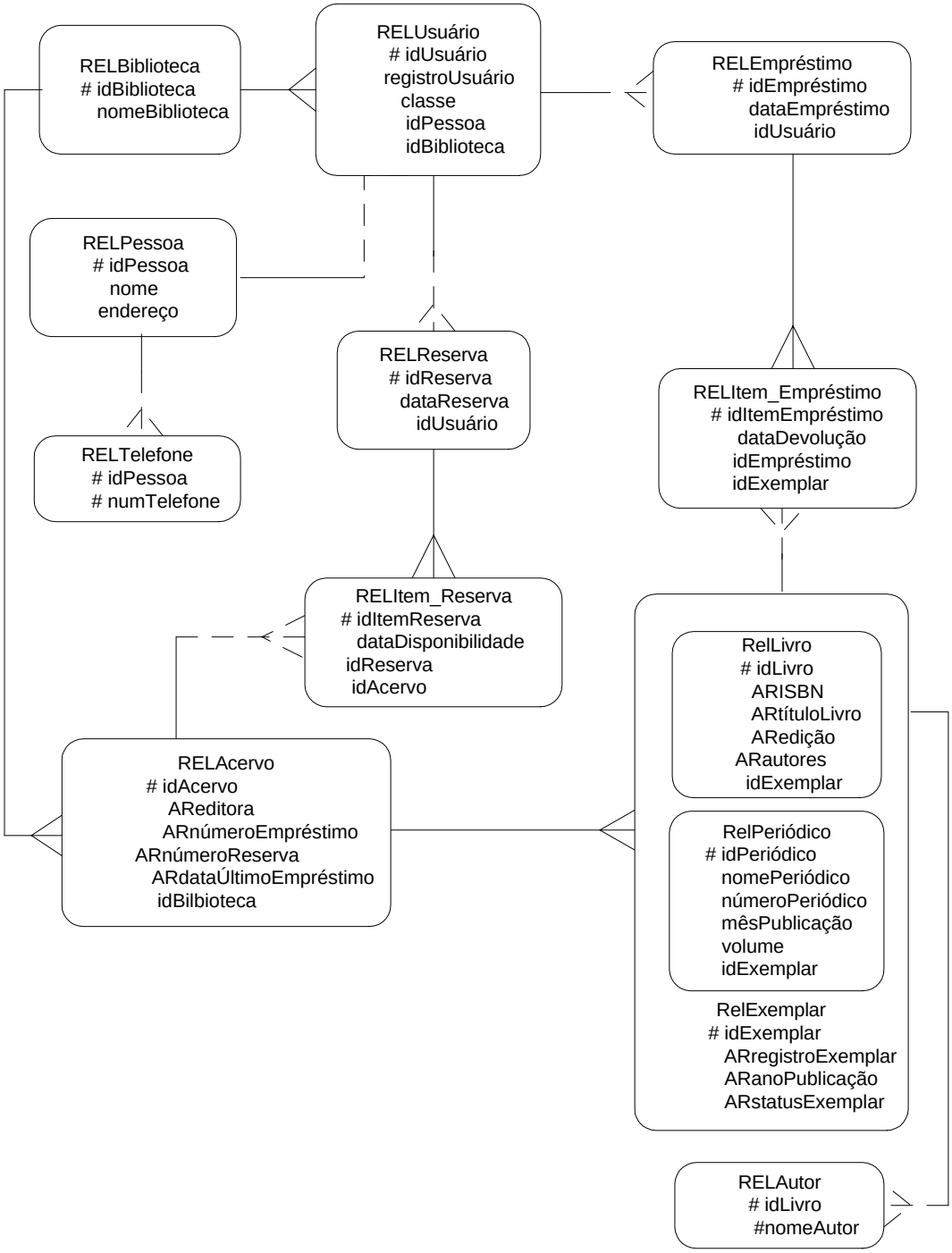


Fig. E.2 – Representação do modelo relacional de dados do Sistema Simplificado de Biblioteca.

2.4 Conclusão

A elaboração do estudo de caso utilizando as estratégias e padrões apresentados procuram elucidar e evidenciar a sua aplicação num problema comum, mas de generalidade suficiente para proporcionar a transposição para uma outra situação.

Ao aplicar-se esta abordagem, observa-se como consequência a simplificação das tarefas de modelagem e documentação, que são gradualmente constituídas através da finalização de cada etapa. Este procedimento gera, ao final da modelagem, informação organizada e suficiente para que se tenha um modelo bem estruturado. Esta característica contribui também para auxiliar na manutenção do sistema.

As estratégias definem soluções alternativas para diferentes situações encontradas nos mais diversos problemas. O conhecimento deste procedimento proporciona economia de tempo, já que assim são exploradas somente as possibilidades viáveis e de interesse para a solução do problema em questão.

Um aspecto importante com relação ao emprego das estratégias e dos padrões apresentados é o fator reutilização, consequência direta da idéia de generalidade. A reutilização contribui para o incremento tanto da qualidade como da produtividade para os subsequentes sistemas.