Rebeca Schroeder

Uma Abordagem para Projeto Lógico de Banco de Dados XML baseada em Informações de Carga de Dados

Florianópolis - SC
2008

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Rebeca Schroeder

Uma Abordagem para Projeto Lógico de Banco de Dados XML baseada em Informações de Carga de Dados

dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de mestre em Ciência da Computação.

Orientador: Dr. Ronaldo dos Santos Mello

Florianópolis, setembro de 2008

Uma Abordagem para Projeto Lógico de Banco de Dados XML baseada em Informações de Carga de Dados

Rebeca Schroeder

Esta dissertação foi julgada adequada para a obtenção do título de mestre em Ciência da Computação, área de concentração Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Dr. Frank Augusto Siqueira

Banca Examinadora

Dr. Ronaldo dos Santos Mello

Dr. Carlos Alberto Heuser (UFRGS)

Dr. Renato Fileto (UFSC)

Dr. Frank Augusto Siqueira (UFSC)

À Deus, porque sem Ele nada do que foi feito se fez.

Agradecimentos

Estou plenamente convicta de que não conquistamos nada sozinhos, mesmo em trabalhos classificados como individuais. Durante o período do mestrado muitas pessoas se tornaram fundamentais para esta conquista, pessoas que considero participantes desta vitória junto comigo.

Em primeiro lugar gostaria de agradecer de forma especial a Deus pelo dom da vida, pelo sustento recebido em todas as coisas e por estar presente em cada passo desta caminhada.

Sou muito grata a minha família pelo apoio e incentivo que sempre me proporcionaram. Sei que a vida acadêmica inicia com pequenos passos, e foram vocês que me ensinaram a dar os primeiros. Muito, muito obrigada! Também agradeço ao meu noivo e a sua família que estiveram sempre comigo e compreenderam meus momentos de ausência durante o período do mestrado. Amo todos vocês!

Agradeço ao meu orientador, Ronaldo Mello, por ter acreditado em mim mesmo sem me conhecer no início. Seu incentivo e conselhos foram fundamentais para que isto tudo fosse possível. Muito obrigada pelos valiosos ensinos! Gostaria de agradecer também aos professores Carlos Heuser, Renato Fileto e Frank Siqueira pelas sugestões quanto à continuidade deste trabalho.

Agradeço a Universidade Federal de Santa Catarina pela oportunidade concedida e a CAPES pelo apoio financeiro durante o primeiro ano de curso.

Gostaria de concluir agradecendo a todas as pessoas que estiveram indiretamente envolvidas neste processo. Especialmente aos meus amigos que sempre estão por perto!

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Embora XML seja um padrão de fato para o intercâmbio e representação de informações, atualmente observa-se que documentos XML passaram a atuar como verdadeiros *containers* de dados a serem recuperados pelas aplicações. Como prova disto, sistemas gerenciadores de banco de dados passaram a fornecer meios para a persistência e manipulação de documentos XML. Diante desta nova atuação para este formato de dados, a necessidade de metodologias para a organização de esquemas XML se torna evidente. Diversos trabalhos na literatura fornecem contribuições para o projeto de documentos XML. Entretanto, não existem metodologias completamente adequadas e consolidadas para atuar no projeto de um banco de dados XML.

Este trabalho vem ao encontro desta necessidade, provendo uma metodologia para o projeto lógico de banco de dados XML. O foco deste trabalho está no processo de transformação de um esquema conceitual em um esquema lógico XML. A metodologia proposta considera todos os construtores do modelo conceitual EER (*Extended Entity-Relationship*), bem como as restrições conceituais, para a geração de uma estrutura XML equivalente. Esta estrutura é definida por um modelo lógico que abstrai os modelos de implementação XML existentes. Informações referentes à carga estimada para o banco de dados são utilizadas para efetuar otimizações na estrutura XML durante o processo de transformação. Os resultados apresentados por este trabalho demonstram que a metodologia de conversão proposta produz esquemas XML que representam devidamente os conceitos e restrições de um esquema conceitual, que evitam a redundância de dados e que apresentam uma estrutura otimizada capaz de responder com eficiência às operações mais frequentes de um banco de dados XML.

Abstract

Even though XML is a *de facto* standard for information exchanging and representation, XML documents are being currently used as data containers whose data are retrieved by several kinds of applications. In fact, database management systems have been extended to be able to store and manipulate XML documents. Because of this trend, the need for methodologies for organizing XML schemata becomes evident. Several work related to XML document modeling are available in the literature. However, there are no suitable methodologies for designing XML databases.

This work provides an approach for the logical design of XML databases. It focuses on translating a conceptual schema into an XML logical schema. Our methodology considers all EER (Extended Entity-Relationship) conceptual model constructs as well as conceptual constraints for generating an equivalent XML structure. This structure is defined by a logical model which abstracts the existing XML implementation models. Workload estimated on a database is considered to perform structure improvements during the conversion process. Experiments demonstrate that our conversion methodology produces XML schemata which correctly represent the concepts and constraints of a conceptual schema as well as do not allow data redundancy. Despite that, the structure of generated schemata allows efficient answers to the frequent operations of the XML database.

Capítulo 1

Introdução

O modelo de dados XML se consagrou como um padrão para o intercâmbio e representação de dados. Diversas aplicações começaram a manipular documentos definidos em XML, criando seus próprios padrões para a representação das informações de um domínio. Com o aumento do volume de dados XML disponível, principalmente na *Web*, documentos XML passaram a ser utilizados não apenas como um formato de transferência, mas também como *containers* de dados recuperados e atualizados por diversos tipos de aplicação.

Segundo Bourret, R. [BOU 05], a linguagem XML, especificada pelo *World Wide Web Consortium (W3C)* [w3c 08], assemelha-se à tecnologia de Banco de Dados (BD) por apresentar um formato de armazenamento de dados (documentos XML), esquemas (DTD, XML Schema, etc), linguagens de consultas (XPath, XQuery, etc) e interfaces de programação (SAX, DOM, JDOM, etc). No entanto, esta tecnologia não suporta muitas das características próprias de um Sistema de Gerência de BD (SGBD), como eficiência no armazenamento, indexação, segurança e transações.

O uso da XML como um formato para o armazenamento de dados ocasionou o surgimento de uma série de sistemas destinados a gerenciar este tipo de dado [SAL 01]. SGBDs comerciais, em especial os relacionais, passaram a prover soluções para a persistência e manipulação de documentos XML [DIL 05, ENN 00]. Paralelamente a este fato surgiram SGBDs destinados a armazenar documentos XML em sua forma nativa

[SCH 01, JAG 02]. Porém, BDs XML não constituem uma nova versão para os legados BDs hierárquicos, nem tão pouco um substituto para os consolidados BDs relacionais, mas uma solução para a persistência de novos tipos de informação que se adequam à estrutura semi-estruturada provida pelo modelo de dados XML [BOU 05].

Segundo Moro, M. et al. [MOR 07], a persistência de dados na forma nativa XML é adequada para representar alguns tipos específicos de informação. Em primeiro lugar estão os artefatos de negócio de aplicações empresariais, onde é comum encontrar elementos de dados que só fazem sentido se agrupados na forma de documentos. Estes documentos, em geral, também contêm campos de texto livre e estruturado, o que caracteriza a natureza semi-estruturada de dados XML. Em segundo lugar, devido à grande quantidade de sistemas de informação que trocam dados através de mensagens XML, algumas aplicações necessitam armazenar estas mensagens para posterior rastreamento das informações. Uma terceira razão está ligada a variações nos esquemas XML. Estas variações se referem às diversas formas de representar um conteúdo, devido à presença de atributos ou elementos opcionais, bem como devido a várias versões para representação do modelo de conteúdo dos conceitos.

O crescente volume de dados XML manipulado pelos mais diversos tipos de aplicação, tem evidenciado pontos em aberto na área de gerência de dados XML, em especial a necessidade de metodologias adequadas para o projeto de BDs XML [SAL 01]. Uma metodologia *top-down* semelhante ao projeto de BDs convencionais pode ser aplicada neste contexto, iniciando-se pela modelagem do domínio através de um modelo de alto grau de abstração, até alcançar um modelo de implementação XML. Embora alguns projetistas prefiram produzir documentos XML definindo-os diretamente através de um modelo de implementação, o projeto *top-down* se mostra mais adequado, pois, após representar adequadamente um domínio de aplicação através de um modelo de alta abstração, é possível determinar uma representação mais adequada para as informações no modelo de implementação XML. Além disto, durante o projeto *top-down* a estrutura XML pode ser determinada de forma que a redundância de dados seja evitada e as principais operações do BD possam ser desempenhadas da maneira mais eficiente possível.

Diversos trabalhos na literatura estão focados no projeto de documentos XML,

contudo não há consenso quanto a metodologias para o projeto de BDs XML. Algumas propostas definem o processo de produção de esquemas XML a partir de esquemas conceituais, como as metodologias propostas em [CON 00, BIR 00, FON 06, PIG 05]. Entretanto, muitos destes trabalhos utilizam modelos que não se enquadram adequadamente aos 3 níveis de modelagem de um BD. Por exemplo, algumas metodologias como [SEN 03, PSA 01] consideram construções específicas do modelo de dados XML em seus modelos conceituais, conferindo um caráter mais lógico do que conceitual a estes modelos. Outros trabalhos como [CHO 03, LIU 06a, KRU 03] partem de um modelo conceitual como o EER e a UML para a representação direta dos conceitos em um modelo de implementação XML específico, ignorando a abstração do nível de modelagem lógica do BD.

Além de modelos inadequados para o projeto de BDs XML, algumas abordagens que definem processos de conversão de um modelo conceitual para um modelo XML não apresentam estratégias adequadas de conversão para construções conceituais como generalizações e tipos união [ELM 85]. Uma outra deficiência na área está relacionada à falta de soluções para prover otimizações na estrutura lógica dos documentos XML, em especial metodologias para a produção de esquemas XML compactos, bem estruturados e que possam garantir uma boa eficiência no desempenho das operações mais freqüentes do BD. Com relação a isto, informações da carga do BD XML, por exemplo, poderiam ser estimadas para auxiliar na construção de estruturas XML mais adequadas.

Diante das problemáticas levantadas, o presente trabalho propõe uma abordagem para o projeto lógico de BDs XML. Esta abordagem está focada na conversão de esquemas conceituais EER em esquemas lógicos XML. Os esquemas XML são definidos por um modelo lógico proposto para atuar como um modelo de abstração para os modelos de implementação XML existentes. O processo de conversão é composto por um conjunto de regras de mapeamento que conferem alternativas de conversão para todos os construtores do modelo EER definido segundo Batini et al. [BAT 92].

O objetivo principal desta abordagem é produzir esquemas lógicos XML que representam adequadamente as informações modeladas no nível conceitual. Os esquemas lógicos XML são produzidos de forma a não gerar redundância de dados para os documentos

XML conformados a estes esquemas. Além disto, o processo de conversão considera estimativas da carga do BD para gerar estruturas XML mais otimizadas. Tais otimizações se referem à estrutura de aninhamento entre os elementos XML e ao tamanho do esquema gerado. O volume de dados e as principais operações estimadas para o BD são consideradas como informações de carga para a geração de esquemas XML que possam responder com mais eficiência às consultas mais freqüentes sobre o BD.

1.1 Objetivo Geral

Este trabalho visa definir uma abordagem capaz de apoiar o projeto lógico de BDs XML, representando adequadamente as informações modeladas no projeto conceitual em um esquema otimizado definido por um modelo lógico XML.

1.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- **Definir um modelo para atuar como o modelo lógico XML**. A abordagem de conversão produz esquemas definidos por um modelo lógico XML que abstrai os modelos de implementação XML existentes, como DTD [BRA 00] e XML Schema [THO 04]. A abstração deste modelo é adequada para atuar no nível lógico do projeto de um BD, conferindo portabilidade à abordagem;

- **Definir regras de conversão para os construtores de um modelo conceitual**. Um conjunto de regras de conversão é definido por este trabalho com a finalidade de prover estratégias de conversão para todos os construtores conceituais do modelo EER;

- **Estabelecer um processo de conversão EER-XML para a aplicação das regras**. Um processo de conversão é definido na forma de um algoritmo para a aplicação organizada das regras de conversão. O algoritmo produz esquemas XML compactos,

pois efetua a conversão de cada fragmento conceitual através das regras de conversão que geram a menor porção de esquema XML e que são adequadas para a conversão da construção conceitual em questão. Este processo tem como objetivo produzir esquemas lógicos XML não redundantes e bem estruturados quanto às restrições impostas pelos relacionamentos conceituais entre os conceitos do esquema conceitual. Um processo de conversão automático é estabelecido com a finalidade de facilitar a produção de esquemas XML pelas aplicações;

- **Definir uma metodologia para a modelagem da carga do BD**. Uma metodologia para modelagem da carga do BD é proposta com a finalidade de gerar informações que possam guiar o processo de conversão na produção de esquemas XML mais otimizados;

- **Estabelecer um processo de conversão EER-XML baseado em análise da carga do BD**. O processo de conversão EER-XML é alterado para considerar informações da carga estimada para o BD. Esquemas XML capazes de responder com mais eficiência às principais operações do BD são produzidos por este processo.

1.3 Estrutura do Documento

Este trabalho está organizado em mais 6 capítulos. No capítulo 2 são apresentados os princípios do projeto lógico de BDs convencionais, envolvendo os objetivos desta fase de modelagem, os modelos considerados e os requisitos para a produção de esquemas lógicos de BDs a partir de esquemas conceituais. O capítulo 3 apresenta uma análise dos trabalhos relacionados a metodologias para o projeto de esquemas XML. Esta análise visa identificar contribuições destes trabalhos para o projeto lógico de BDs XML, bem como identificar pontos em aberto nesta área.

No capítulo 4 é proposta a metodologia para conversão de esquemas EER em esquemas lógicos XML. Neste capítulo é definido o modelo lógico utilizado pela metodologia, as regras de conversão aplicadas sobre cada construtor do EER e o processo de conversão que define a ordem de aplicação das regras sobre os fragmentos do esquema conceitual.

No capítulo seguinte, a metodologia de conversão é aprimorada para considerar estimativas da carga prevista para o BD que está sendo modelado. A metodologia para a modelagem da carga de um BD é também apresentada por este capítulo. As informações de carga se referem a estimativas quanto ao volume de dados e as principais operações a serem executadas pelo BD. Estas informações são analisadas com a finalidade de produzir estruturas XML capazes de responder com eficiência às principais operações do banco.

O capítulo 6 apresenta um estudo de caso realizado para o domínio de compras e vendas de empresas. Um esquema lógico XML é produzido através da metodologia proposta no capítulo 4, e um segundo esquema a partir da metodologia baseada em análise da carga do BD definida no capítulo 5. Estes esquemas foram comparados, juntamente com um esquema XML que foi produzido para representar esquemas gerados pelas metodologias correlatas a este trabalho. Na avaliação experimental, produziu-se documentos XML a partir destes esquemas com a finalidade de executar as operações sobre estes. Estas operações foram executadas sobre o BD XML Nativo *Tamino* [SCH 01]. Os resultados demonstraram o ganho do esquema gerado pela análise da carga do BD em termos do tempo de resposta das operações, bem como em termos do número de acessos à elementos dos documentos XML produzidos. Por fim, as contribuições deste trabalho e as conclusões referentes a este são apresentadas pelo capítulo 7, juntamente com sugestões para trabalhos futuros.

Capítulo 2

Projeto Lógico de Banco de Dados

O projeto de banco de dados é um processo constituído por três fases de modelagem de dados [HEU 04, ELM 00, BAT 92]: (i) projeto conceitual, (ii) projeto lógico e (iii) projeto físico. Iniciando-se pelo projeto conceitual, um esquema representando as informações de um domínio é modelado através de um modelo de alto nível de abstração. Nos níveis subsequentes, o esquema conceitual é transformado em esquemas de mais baixa abstração até alcançar uma representação adequada ao modelo alvo, o modelo no qual o Banco de Dados (BD) será fisicamente estabelecido. Este processo *top-down* é considerado uma das melhores formas de realizar a modelagem de um banco de dados, visto ser mais claro e natural para o projetista modelar as informações de uma aplicação a partir de modelos que refletem mais diretamente os fatos do mundo real e seus relacionamentos [MOK 06, EMB 98, TEO 86].

A Tabela 2.1 apresenta a dependência de cada uma das fases de projeto quanto ao modelo lógico e ao modelo específico do Sistema Gerenciador de Banco de Dados (SGBD) para o qual o BD está sendo modelado. Entende-se por *modelo lógico* o modelo que representa a classe, ou a natureza dos dados manipulados pelo SGBD. São exemplos de modelos lógicos os modelos hierárquicos, relacionais, orientados a objetos e outros. O *modelo específico* se refere ao modelo de armazenamento estabelecido pelo SGBD em questão. O projeto conceitual é a etapa de maior abstração, pois não apresenta dependência a nenhum modelo específico ou lógico de BD. O projeto lógico apresenta uma

Tabela 2.1: Dependência das etapas de projeto quanto aos modelos de um SGBD [BAT 92]

Níveis de Projeto	**Modelo Lógico do SGBD**	**Modelo Específico do SGBD**
Projeto Conceitual	Não	Não
Projeto Lógico	Sim	Não
Projeto Físico	Sim	Sim

abstração média, pois é dependente do modelo lógico mas não do modelo específico do SGBD. Por último, o projeto de mais baixo nível é estabelecido pelo projeto físico, visto ser totalmente dependente de um SGBD específico.

A metodologia de modelagem em três níveis constitui um consenso para o projeto de BD tradicional. Assim como ocorre para BDs tradicionais, o projeto de BD XML tem sido estabelecido nestes três níveis de abstração [SAL 01, ROU 02]. Com o foco na modelagem lógica, este capítulo apresenta os princípios da metodologia para projeto de BDs tradicionais com a finalidade de embasar o projeto lógico de BDs XML.

2.1 Projeto Conceitual de Bancos de Dados Tradicionais

Segundo Elmasri, R. et al. [ELM 00], o modelo utilizado pelo projeto conceitual deve ser: (i) *Expressivo*, provendo estruturas de modelagem e conceitos suficientes para representar os dados de um domínio de informação; (ii) *Simples*, sendo capaz de ser compreendido por usuários leigos; (iii) *Mínimo*, no sentido que cada conceito tem um significado distinto dos demais conceitos providos pelo modelo; e (iv) *Formal*, sendo que cada conceito do modelo deve apresentar uma interpretação bem definida e única.

Vale observar que um modelo conceitual deve equilibrar sua expressividade e simplicidade para que os requisitos mencionados sejam atendidos, visto que o aumento da expressividade, em geral, reduz a simplicidade do modelo. A busca pela expressividade de modelos conceituais impulsiona o surgimento de uma série de novos modelos conceituais para melhor representar características de um modelo de implementação. Isto vem ocorrendo também no campo da modelagem conceitual de BD XML, onde alguns autores vêm propondo novos modelos conceituais específicos para tratar o modelo de da-

dos XML [PSA 01, SEN 03, EMB 04]. Embora chamados de modelos conceituais, estes novos modelos apresentam características lógicas do modelo de implementação além das construções convencionais de um modelo conceitual. Conseqüentemente, esta nova composição de modelos fere a independência do modelo conceitual, atribuindo a este uma dependência ao modelo de implementação.

Metodologias consolidadas para a modelagem de BD assumem que o modelo conceitual deve ser independente de quaisquer modelos de implementação e que a independência entre os modelos do projeto de um BD é um requisito fundamental para garantir a portabilidade da metodologia [BAT 92, ELM 02]. Exemplos de modelos conceituais que atendem a estes requisitos são: o modelo Entidade-Relacionamento (ER) [CHE 76], o modelo Entidade-Relacionamento Estendido (EER), Object Role Modeling (ORM) [HAL 96] e o diagrama de classes da UML.

Os construtores conceituais utilizados por um modelo conceitual podem ser divididos em três tipos de abstração, segundo [BAT 92]: (i) *Abstração de Classificação*, (ii) *Abstração de Agregação*, e (iii) *Abstração de Generalização*. Na *abstração de classificação* um conceito é identificado como uma classe ou entidade do mundo real juntamente com suas propriedades ou atributos. Na *abstração de agregação* os conceitos identificados são associados uns aos outros, gerando novas classes que representam estas associações ou relacionamentos. Na *abstração de generalização* é estabelecido um relacionamento de subconjunto entre os elementos de duas ou mais classes. Neste caso, todas as propriedades de uma classe definida como genérica são herdadas pelas classes que constituem especializações da classe genérica.

Em geral, cada modelo conceitual tradicional provê construtores de modelagem para os três tipos de abstração apresentados. Por ser um modelo que provê construtores adequados para estes tipos de abstração, o modelo EER é considerado um modelo expressivo para a modelagem conceitual de bancos de dados [ELM 00]. A seguir são apresentados os construtores conceituais providos por este modelo para cada um dos tipos de abstração. A Figura 2.1 é utilizada para fins de exemplificação.

1. *Abstração de Classificação:* Os conceitos são classificados em *Tipos Entidade*, os

quais são constituídos de *atributos* que representam as propriedades deste conceito. Os atributos são caracterizados por suas *ocorrências mínimas e máximas* e pelo *modelo de conteúdo* que apresentam dentro de um tipo entidade. Quanto a *ocorrência mínima*, um atributo é caracterizado como *opcional* ou como *obrigatório*. Um atributo *opcional* apresenta ocorrência mínima igual a 0, o que denota que este atributo pode não ser especificado (nulo) para uma instância do tipo entidade ao qual pertence. Um atributo *obrigatório* apresenta ocorrência mínima igual a 1, denotando que este atributo será sempre especificado para qualquer instância do tipo entidade. Quanto a *ocorrência máxima*, um atributo é caracterizado como *monovalorado* ou *multivalorado*. A ocorrência máxima de um atributo *monovalorado* é sempre 1 e de um *multivalorado* deve ser superior a 1. O atributo *a3* da Figura 2.1 é um exemplo de atributo multivalorado e obrigatório. Quanto ao *modelo de conteúdo*, um atributo é caracterizado como *simples* ou *composto*. O conteúdo de um atributo *simples* é um valor de tipo simples, enquanto que um atributo *composto* constitui uma composição de outros atributos. O atributo *a2* é um exemplo de atributo composto, obrigatório e multivalorado. Além destas nomenclaturas, um atributo pode também ser *identificador*. Um atributo *identificador* atua como identificador único para as instâncias de um tipo entidade. O identificador de um tipo entidade pode ser composto por mais de um atributo identificador, o que caracteriza uma identificação composta. O atributo *id1* é um exemplo de atributo identificador, neste caso

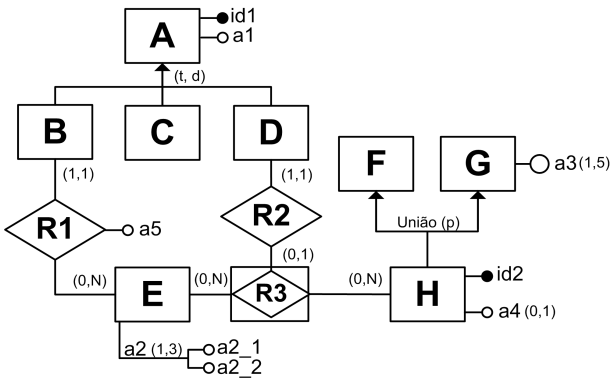

Figura 2.1: Exemplo de um esquema EER

atuando como o identificador da entidade *A*.

2. *Abstração de Agregação:* Uma associação entre conceitos estabelece um *tipo relacionamento*. Um *tipo relacionamento* possui um grau que é definido pelo número de tipos entidade que são associados pelo relacionamento. Por exemplo, o tipo relacionamento *R1* é um relacionamento binário, visto que possui duas entidades participantes. Além disto, cada entidade participante de um relacionamento define suas ocorrências mínimas e máximas no relacionamento. No exemplo, a entidade *B* tem participação (0,N) em *R1*, o que denota que uma instância de *B* pode estar associada a nenhuma ou a N instâncias de *E* através do relacionamento *R1*. Já a entidade *E* tem participação (1,1) em *R1*, denotando que todas as instâncias de *E* devem estar associadas a uma e apenas uma instância de *A* através de *R1*. A participação (1,1) de uma entidade em um relacionamento é conhecida como *dependência de existência*. A cardinalidade de um tipo relacionamento é dada pelas ocorrências máximas das entidades envolvidas nele. Por exemplo, diz-se que a cardinalidade de *R1* é 1:N pois a participação máxima de *E* é 1 e a de *B* é N. Um tipo relacionamento também pode possuir atributos, como o atributo *a5* em *R1*. Além disto, um tipo relacionamento nomeado *RA* pode estar sendo associado por um outro tipo relacionamento, nomeado *RB*. Neste caso, *RA* é transformado em uma *entidade associativa* como ocorre para *R3* que está sendo associado por *R2* na Figura 2.1.

3. *Abstração de Generalização:* Associações de herança entre tipos entidade envolvem entidades caracterizadas como superclasse (entidades genéricas) e entidades especializadas denominadas de subclasses. Basicamente existem duas composições providas por modelos conceituais com relação à abstração de generalização: *hierarquias de generalização* e *tipos união (ou categorias)* [ELM 85]. Estes dois tipos de composições impõem diferentes restrições sobre uma associação de herança de tipos entidade. Em hierarquias de generalização, as restrições de disjunção e de totalidade estabelecem quatro possibilidades de restrição sobre hierarquias de generalização [HEU 04, BAT 92]: total e disjunta (t, d); parcial e disjunta (p, d), total e compartilhada (t,c), parcial e compartilhada (p, c). A hierarquia de generaliza-

ção envolvendo *A* como superclasse e *B*, *C* e *D* como subclasses é um exemplo de uma generalização total e disjunta. A restrição *total* estabelece que uma instância da superclasse *A* deve ser especializada por pelo menos uma instância de uma das subclasses. A restrição *disjunta* estabelece que a superclasse *A* deve ser especializada por apenas uma das subclasses em cada instância da superclasse. Os *tipos união* [ELM 85] são um caso mais restrito de herança múltipla, onde a coleção de instâncias de uma subclasse é representada pela união das instâncias das suas superclasses. Entretanto, uma instância de subclasse pode ser a especialização de apenas uma das suas superclasses. Para tipos união apenas a restrição de totalidade é imposta, o que estabelece duas possibilidades de restrição: tipos união totais e tipos união parciais. O tipo união parcial apresentado pela Figura 2.1 estabelece que nem todas as instâncias de *F* e *G* são especializadas por *H*.

Além dos construtores mencionados, existem composições conceituais importantes como *auto-relacionamentos* ou relacionamentos *recursivos*, *entidades fracas*, hierarquias de generalização de múltiplos níveis e hierarquias de generalização com casos de herança múltipla [BAT 92]. Pela razão que estas composições constituem particularidades ou restrições de tipos relacionamento e hierarquias de generalização, estes conceitos não são discutidos por esta seção, porém devem ser considerados na modelagem lógica de um BD.

2.2 Projeto Lógico de Bancos de Dados Tradicionais

A atividade fundamental do projeto lógico de um BD é transformar um esquema conceitual em um esquema lógico de dados. Enquanto o objetivo do projeto conceitual é produzir um esquema expressivo e capaz de representar os dados de um domínio de informação, o objetivo do projeto lógico é transformar um esquema conceitual em uma representação equivalente em um modelo lógico que se aproxima ao modelo de implementação do BD. Além de representar as informações de um esquema conceitual, o projeto lógico visa alcançar um esquema lógico capaz de utilizar com eficiência as construções permiti-

das pelo modelo lógico em questão [BAT 92].

Segundo [BAT 92], o projeto lógico pode ser subdividido em duas fases: *projeto lógico de alto-nível* e *projeto lógico dependente de modelo*. No *projeto lógico de alto-nível*, o esquema conceitual é transformado em um esquema mais otimizado que é representado através de um modelo lógico independente (abstrato) de qualquer modelo de implementação existente para o tipo de BD em questão. Na fase seguinte, no *projeto lógico dependente de modelo*, o esquema lógico de alto-nível é transformado em um esquema adequado a um modelo de implementação específico. A Figura 2.2 ilustra este projeto lógico de duas fases.

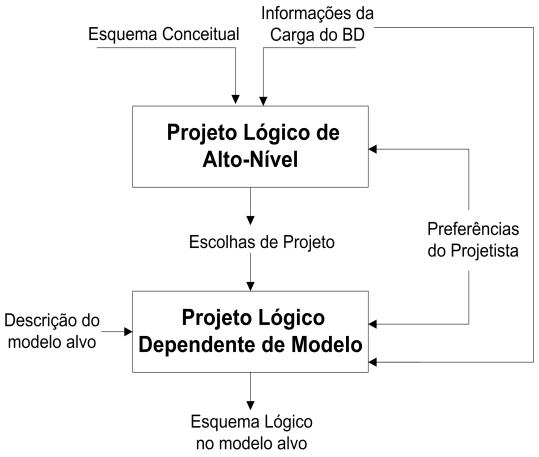

Figura 2.2: Um abordagem de duas fases para o projeto lógico de BD [BAT 92]

A primeira fase tem como entrada um esquema conceitual e, eventualmente, informações referentes a estimativas da carga do BD. Informações da carga do BD podem ser utilizadas pelo projetista para efetuar otimizações no esquema lógico a ser gerado por esta etapa. Na segunda fase, o esquema gerado pela fase anterior é transformado em um esquema mais próximo ao modelo de dados no qual o BD será implementado. Nesta fase é necessário conhecer os construtores e restrições impostas pelo modelo de implementação para que a transformação seja efetivada. O projetista pode interagir com o processo de transformação em cada uma das fases no sentido de escolher por alternativas de conversão de sua preferência. A Figura 2.3 apresenta um exemplo da modelagem conceitual, lógica de alto nível e lógica dependente de modelo para um BD relacional segundo Batini, C. et

al. [BAT 92]. O esquema conceitual representado no modelo Entidade-Relacionamento Estendido (EER - Extended Entity-Relationship) é transformado em um esquema mais otimizado em um modelo lógico de alto-nível. A metodologia proposta em [BAT 92] utiliza o modelo EER também como o modelo de alto-nível para o projeto lógico de BDs relacionais, pois os autores acreditam que é possível inferir características lógicas do modelo relacional no modelo EER. Neste caso, apenas as construções do EER que podem ser facilmente representadas pelo modelo relacional são utilizadas durante o projeto de alto-nível. No projeto lógico dependente de modelo, o esquema EER otimizado é transformado em um esquema relacional.

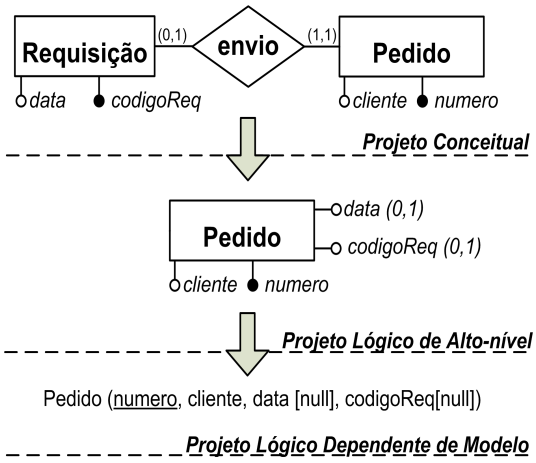

Figura 2.3: Exemplo de modelagem conceitual e lógica para BD relacional segundo Batini, C. et al. [BAT 92]

O esquema gerado pela segunda fase do projeto lógico é submetido ao projeto físico do BD. O projeto físico é responsável pela definição da estrutura de armazenamento física dos dados bem como pelas otimizações que devem ser realizadas sobre esta estrutura com a finalidade de melhorar o desempenho do acesso e recuperação de dados. O projeto físico não é discutido neste trabalho.

De acordo com a abordagem de duas fases apresentada pela Figura 2.2, inicialmente o esquema conceitual é transformado em um esquema lógico independente de qualquer modelo de implementação do BD. Para tanto, é necessário considerar cada um dos construtores conceituais discutidos na seção anterior com a finalidade de prover estratégias de

conversão para o modelo lógico de alto-nível.

Ainda na primeira fase do projeto lógico, algumas otimizações são realizadas para a geração de uma estrutura lógica simplificada. Para produzir estas otimizações podem ser consideradas as decisões do projetista e também informações a respeito da carga prevista para o BD. Considera-se como informações de carga o volume de dados estimados para o BD assim como as operações de consulta e atualização que serão executadas pelo BD. As informações da carga podem ser utilizadas para identificar conceitos modelados pouco utilizados pelas operações previstas para o BD e que poderão ser representados de forma simplificada pelo esquema lógico. Além disto, a estrutura do esquema lógico pode ser melhor formulada se forem identificados os principais caminhos de navegação no esquema que serão percorridos pelas operações que gerarão a maior carga para o BD.

Na segunda fase do projeto lógico em dois níveis, o esquema lógico de alto-nível é então efetivamente transformado em um esquema de acordo com o modelo de implementação do BD. Nesta etapa, alternativas de conversão também precisam ser providas pelo processo para a conversão dos construtores do modelo de alto-nível para os construtores do modelo de implementação. Preferências do usuário são também consideradas durante esta conversão. Segundo [BAT 92], as otimizações a serem aplicadas nesta fase em geral são mínimas, visto que a estrutura entre os conceitos modelados já foi simplificada pela fase anterior. Simplificações nesta etapa estão relacionadas apenas a formas alternativas de representação de uma estrutura do esquema no modelo de implementação, o que justifica a baixa otimização desta etapa.

Segundo [LIU 06a, MOK 06, BIR 00], um projeto de esquemas XML de qualidade deve gerar esquemas que *preservam as informações* modeladas no esquema conceitual, que não permitam a *redundância de dados* e que apresentam uma *estrutura altamente aninhada*. A perda de informação deve ser evitada de forma que toda informação apresentada por um esquema conceitual seja devidamente representada no esquema XML. A redundância de dados nos documentos XML adequados aos esquemas produzidos pelo projeto também deve ser evitada para que operações de atualização nestes documentos não gerem eventual inconsistência de dados. O esquema XML também deve apresentar uma boa estrutura de aninhamento entre os seus elementos de forma que operações de

consulta e atualização sobre esta estrutura sejam processadas eficientemente.

2.3 Considerações Finais

O processo de conversão que envolve a etapa de *Projeto Lógico* é a principal tarefa a ser tratada por este trabalho. O modelo lógico XML a ser utilizado por este trabalho é definido e exemplificado pelo capítulo 4. Uma abordagem de conversão para um modelo lógico XML que considera todos os conceitos do EER é também proposta por este mesmo capítulo. Além de considerar todos os construtores do EER, a abordagem provê um conjunto de alternativas de conversão para cada um dos construtores conceituais. Uma alternativa de conversão pode ser escolhida pelo projetista ou por um processo automático na conversão de um fragmento de esquema EER. No capítulo 5, estimativas da carga do BD são utilizadas para a escolha das alternativas de conversão mais apropriadas e também para a geração de uma estrutura XML otimizada.

Capítulo 3

Trabalhos Relacionados

Diversas propostas para a modelagem de dados XML são encontradas na literatura. Embora muitas destas abordagens não estejam focadas no projeto de BD XML, elas trazem contribuições para a área, principalmente em termos de metodologias para a conversão de modelos conceituais em modelos XML. Diferentes níveis de abstração da modelagem de dados XML são tratados pelas propostas mencionadas, sendo que algumas, inclusive, apresentam novos modelos e formalismos para a modelagem desta natureza de dados.

Abordagens relevantes para a área de modelagem de dados XML são revisadas por este capítulo. O objetivo desta revisão é identificar as contribuições destes trabalhos para o projeto lógico de BD XML e, principalmente, endereçar pontos em aberto na área. Para fins de análise e classificação, os trabalhos foram separados em três grupos de acordo com o modelo base de suas respectivas metodologias: (i) Abordagens baseadas no modelo ER, (ii) Abordagens baseadas no modelo UML, e (iii) Abordagens baseadas em novos modelos para a modelagem de dados XML. A partir desta revisão, uma análise comparativa entre os trabalhos é realizada com o objetivo de relacionar os níveis de modelagem atendidos por cada proposta e a abrangência das metodologias de conversão de modelos de alta abstração para modelos XML.

Um artigo contendo esta análise dos trabalhos relacionados foi publicado nos anais da *III Escola Regional de Banco de Dados* [SCH 07].

3.1 Trabalhos Baseados no modelo ER

O modelo ER e em especial a sua versão estendida, o EER, são modelos conceituais que, embora tradicionais, são adequados para representar os dados de um domínio de informação em um alto nível de abstração, mesmo para a modelagem de dados complexos como é o caso da modelagem de dados XML [EMB 98]. Embora o modelo ER não apresente construções conceituais equivalentes a algumas construções do modelo de dados XML, a sua expressividade limitada garante a independência do modelo em relação a qualquer modelo lógico, neste caso o XML. Conforme discutido no capítulo anterior, esta independência é considerada, por muitos autores, um requisito fundamental para conferir portabilidade a metodologias de projeto de BD.

Alguns trabalhos utilizam o modelo ER/EER como modelo base para suas metodologias de modelagem XML. O objetivo de parte destas metodologias é produzir esquemas XML segundo as especificações DTD ou XML Schema. Desta forma, tais metodologias são direcionadas a produzir esquemas XML em um modelo de implementação XML específico. Outra parcela de trabalhos estabelece a conversão de esquemas ER para estruturas hierárquicas que denotam um modelo lógico XML e abstrato quanto aos modelos de implementação XML existentes.

Os trabalhos propostos por Choi et. al. [CHO 03] e Fong et. al. [FON 06] apresentam metodologias para produzir esquemas XML em DTD a partir do EER. Uma metodologia de projeto unificado para XML é proposto em [CHO 03]. Neste trabalho, um esquema EER é transformado diretamente em uma especificação em DTD que, por sua vez, é transformada em um esquema físico para armazenamento dos dados. O objetivo desta metodologia unificada é a modelagem de mensagens XML que serão trocadas por operações de B2B e que mais tarde devem ser armazenadas em um BD Orientado a Objetos. Os autores de [FON 06] endereçam o problema de aplicações que desejam manter seus dados em BDs relacionais mas precisam trabalhar com estes dados no formato XML no nível da aplicação. Neste sentido, uma abordagem de engenharia reversa é utilizada para transformar um esquema relacional em um esquema EER que posteriormente é convertido em uma DTD. A metodologia de conversão EER-DTD proposta por

[FON 06] considera grande parte dos construtores conceituais do EER e suas restrições para gerar um esquema XML em DTD. Os construtores conceituais considerados pelas abordagens são listados e comparados pela Tabela 3.2 na seção 3.4.

Assim como proposto em [CHO 03] e [FON 06], os autores de [LIU 06a] e [PIG 05] fornecem metodologias para a conversão do modelo ER/EER em um modelo de implementação XML específico. No caso das propostas em [LIU 06a] e [PIG 05], um esquema XML é produzido em uma especificação *XML Schema* [THO 04]. A desvantagem comum entre todas estas soluções é a dependência das abordagens com relação a uma linguagem de definição de esquema XML específica.

Os trabalhos de Elmasri et. al. [ELM 02] e Lee et. al. [LEE 01] fornecem metodologias para a conversão de esquemas ER para estruturas hierárquicas. Em [ELM 02] é proposto um algoritmo para a geração de visões hierárquicas customizadas a partir de uma modelagem EER. Inicialmente, os eventuais ciclos de um esquema EER são removidos e, na sequência, o usuário informa uma entidade de partida para a geração de uma estrutura hierárquica. Esta entidade de partida se transforma no elemento raiz da estrutura hierárquica, sendo que todos os caminhos no esquema EER que têm relacionamentos com a entidade de partida formarão a hierarquia de elementos da estrutura a ser gerada. A estrutura hierárquia é dita customizada visto que diferentes visões podem ser obtidas mediante a escolha da entidade de partida pelo usuário. Um algoritmo para a geração de esquemas em *XML Schema* a partir de estruturas hierárquicas é também apresentado em [ELM 02]. Em [LEE 01] é apresentada uma proposta de projeto para BDs semi-estruturados. O modelo ER é utilizado na modelagem conceitual, sendo mapeado para um esquema lógico em forma de grafo denominado *S3-Graph* (SemiStructured Schema Graph). *S3-Graph* relaciona os conceitos hierarquicamente ou através de referências. Ele também permite representar a seqüência de disposição dos nós e a cardinalidade máxima de um conceito na hierarquia. Ambos os trabalhos fornecem meios de conversão de modelos ER para um modelo lógico XML representado por estruturas hierárquicas. Embora tais trabalhos estejam de acordo com a metodologia de projeto lógico XML, algumas deficiências são encontradas principalmente nos algoritmos de conversão propostos. Tais deficiências serão discutidas na seção 3.4.

Os autores de [WIW 06, JAG 04] apresentam uma metodologia para geração de esquemas XML de *multi-cores* a partir de esquemas ER. Cada cor de um esquema de *multi-cores* representa uma visão do esquema conceitual completo. Em outras palavras, cada uma das cores contém uma hierarquia de elementos para representar porções do esquema conceitual. A justificativa para tal abordagem é que, em geral, um esquema conceitual não pode ser diretamente representando por um esquema hierárquico sem que haja redundância de dados. A redundância de dados é geralmente evitada pelo uso de relacionamentos de referência em esquemas hierárquicos. Porém, os autores endereçam o problema de que relacionamentos de referência elevam consideravelmente o custo da execução de consultas sobre documentos XML. Sendo assim, a proposta é a eliminação destes relacionamentos de referência e também de redundância de dados utilizando, para isto, um modelo lógico XML de *multi-cores*. Neste caso, não é necessária a utilização de referências entre elementos do esquema XML, pois todos os relacionamentos conceituais são representados de forma hierárquica por alguma das cores consideradas.

Cada cor de um esquema *multi-cores* atua como um esquema XML individual. Para a execução de consultas sobre um esquema multi-cores, todas as cores são acessadas para a recuperação das informações de um único esquema conceitual. A Figura 3.1 apresenta um exemplo de um esquema *multi-cores*, onde as cores são posteriormente sobrepostas para a execução de consultas. No exemplo observa-se que o elemento *order* é representado como sub-elemento de elementos distintos em cada uma das cores apresentadas. Por exemplo, *order* é representado como um sub-elemento de *make* na porção do esquema de cor azul, e como um sub-elemento de *billing* na cor vermelha. Desta forma, dependendo da intenção de consulta, os elementos do tipo *order* são recuperados através da cor azul (partindo do elemento *make*) ou através da cor vermelha (partindo do elemento *billing*).

O principal problema desta abordagem é a necessidade de estender linguagens de definição de esquema XML assim como linguagens de consulta XML para suportar estes esquemas XML de *multi-cores*. Além disto, um *overhead* é gerado na execução das consultas para considerar as diversas cores sobre documentos XML adequados a estes esquemas. Quanto ao nível físico, os autores não deixam claro qual o modelo físico utilizado para a persistência de documentos XML conformados a esquemas *multi-cores*.

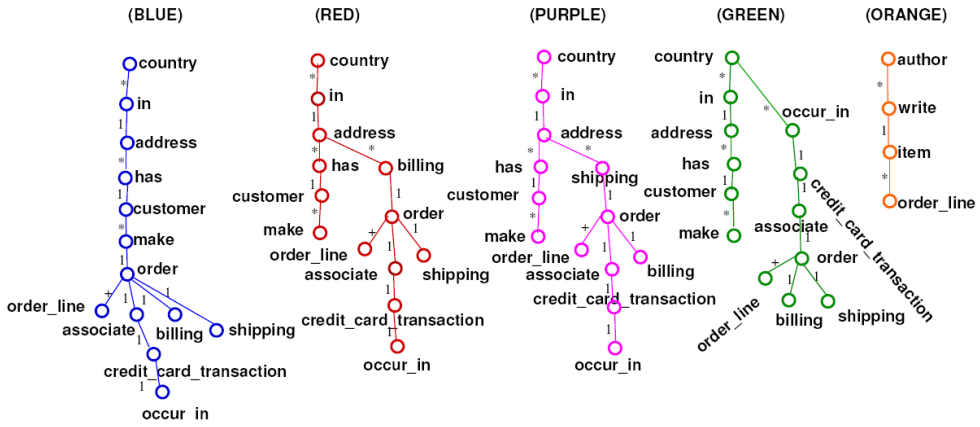

Figura 3.1: Exemplo de esquema *multi-cores* [WIW 06, JAG 04]

3.2 Trabalhos Baseados no modelo UML

Assim como as abordagens baseadas em ER, outras propostas utilizam a UML como modelo base para metodologias de conversão para o modelo XML. Estas propostas utilizam o diagrama de classes, a parte estática da UML, como o diagrama base para a modelagem conceitual. Alguns trabalhos propõem extensões para os diagramas UML com a finalidade de representar características específicas do modelo de dados XML a serem utilizadas no projeto lógico. Estas extensões são inseridas através dos mecanismos próprios da UML para a extensão, como estereótipos, restrições e valores rotulados.

Os trabalhos de Routledge et. al. [ROU 02, BIR 00] e Liu et. al. [LIU 06b] propõem um projeto de BD em três níveis. No nível conceitual são utilizadas as notações de classes, atributos, classes associativas e relacionamentos de associação do diagrama de classes UML. No nível lógico, os diagramas de classe são estendidos com estereótipos, especificando conceitos específicos do modelo de dados XML, como elementos simples, elementos complexos e atributos. A modelagem explícita de conceitos XML através de diagramas UML estendidos permite definir regras de conversão para o nível de implementação, onde são utilizadas linguagens de esquema como *XML Schema* e DTD.

A UML é uma linguagem criada para modelagem de sistemas orientados a objetos (OO) [EMB 98]. Consequentemente, o diagrama de classes, por exemplo, apresenta construtores do modelo de dados orientado a objetos. Embora apresentando caracterís-

ticas específicas de um modelo lógico de dados (o modelo OO), diagramas UML vêm sendo utilizados na modelagem conceitual de bancos de dados. No entanto, para cumprir devidamente o papel de um modelo conceitual, devem ser adicionadas algumas restrições conceituais não existentes no diagrama de classes da UML [HAL 98]. Por exemplo, a restrição de identificação primária de uma classe. Um outro exemplo de adaptação endereçada em [LIU 06b] é a introdução de um construtor de agregação genérico onde o número de componentes agregados é ilimitado, considerando que até a UML2 é possível construir apenas agregações binárias.

O uso do diagrama de classes da UML com extensões para desempenhar o papel de modelo lógico deve ser questionado. Primeiramente, porque torna o modelo lógico escolhido dependente do modelo conceitual utilizado. Esta situação compromete a flexibilidade do projeto se, por exemplo, um outro modelo conceitual precisa ser utilizado. Em segundo lugar, estes esquemas lógicos representam implicitamente, no interior das classes modeladas, a estrutura hierárquica do esquema XML a ser gerado. O esquema lógico é composto por um conjunto de classes UML não mais relacionadas graficamente por notações de associação, sendo que os sub-elementos de um elemento XML são representados como atributos dentro da classe UML que representa tal elemento. Graficamente, tal modelo é pouco claro, visto que a estrutura hierárquica dos esquemas lógicos XML é a principal composição a ser avaliada por este nível de projeto.

Outros trabalhos como Eckstein et. al. [CON 00, ECK 04] e Combi et. al. [COM 06] têm como foco a representação de esquemas XML diretamente em diagramas de classe UML. Para tanto, eles partem do nível lógico, definindo extensões para especificar cada um dos conceitos das linguagens de definição de esquema XML. Eles especificam um metamodelo que define elementos, tipos simples e complexos, e restrições que cada uma das linguagens de definição de esquema XML pode apresentar, como restrições de grupos (all, sequence, choice) e identificadores (id, idref, key, keyref), dentre outros. Na sequência, estes esquemas lógicos em UML são transformados em especificações em DTD ou *XML Schema*. A etapa de modelagem conceitual não é considerada por estes dois trabalhos, pois os autores assumem que o domínio de informação pode ser diretamente modelado por um modelo que apresenta tanto características conceituais quanto

lógicas da aplicação. Novamente, abordagens como estas ferem a independência e, por consequência, a portabilidade de suas metodologias de modelagem.

Os trabalhos de Quang et. al. [QUA 05] e Kudrass et. al. [KUD 03, KRU 03] propõem conversões de esquemas conceituais em UML diretamente para um modelo de implementação XML específico. Em [QUA 05], os autores apresentam uma taxonomia de estilos de projeto de esquemas XML. A classificação é baseada na estrutura resultante das instâncias dos documentos XML. Basicamente, a taxonomia define dois estilos de projeto: *nesting* e *linking*. O estilo *nesting* estabelece uma estrutura em que os elementos são aninhados uns aos outros. No estilo *linking*, o conteúdo de um elemento é associado a outro elemento através de mecanismos como ID/IDREF/IDREFS, key/keyref, XLink e XPointer das linguagens de definição de esquema XML. Embora classificados como dois tipos de projetos distintos, ambos os estilos podem ser encontrados em um esquema XML visto que classes, atributos e alguns relacionamentos conceituais são geralmente representados de forma hierarquizada (*nesting*) e alguns tipos de relacionamentos precisam ser representados por valores/relacionamentos de referência (*linking)*.

A abordagem proposta por Routledge et. al. [ROU 02, BIR 00] traz contribuições importantes na parte de transformação do modelo conceitual para o lógico, apresentando regras para determinar o aninhamento de classes em nível lógico a partir da navegação entre as classes em nível conceitual. Este trabalho prioriza a representação aninhada dos elementos, de forma a maximizar a conectividade entre os elementos e minimizar a redundância de dados sempre que possível. Em [BIR 00] são apresentadas 12 heurísticas para geração do "melhor"esquema XML a partir de um esquema ORM. Os autores de [MOK 06, EMB 01] apresentam uma abordagem semelhante, onde um esquema conceitual definido por um modelo conceitual genérico é transformando em um esquema XML compacto e livre de redundância de dados. Embora apresentando soluções para otimizar a estrutura XML, tais trabalhos são baseados em critérios relacionados apenas as restrições impostas pelo esquema conceitual para a determinação da estrutura dos esquemas XML. Informações como o volume de dados e as operações esperadas para a aplicação XML não são consideradas por tais metodologias.

3.3 Trabalhos Baseados em Novos Modelos

Uma terceira categoria de abordagens para modelagem XML se baseia em novos modelos específicos para XML. Os trabalhos de Mani [MAN 04], Sengupta et. al. [SEN 03] e Psaila et. al. [PSA 01] estendem o modelo ER com características específicas do modelo de dados XML. O modelo *EReX* (ER extended for XML) [MAN 04] propõe uma extensão do ER para representar conceitualmente características do modelo XML como *union types* e ordenação de elementos. Um algoritmo para traduzir esquemas *EReX* para esquemas em um modelo XML ideal é também proposto por este trabalho. Este modelo ideal, chamado *XGrammar*, é uma gramática baseada na notação de linguagens de esquema como DTD, *XML Schema* e RELAX-NG. Desta forma, a *XGrammar* atua como um modelo lógico XML.

Em [SEN 03] e [PSA 01], extensões do ER são criadas para representar elementos simples, complexos, mistos, bem como relações de generalização, associação e ordenação, dentre outros. Um exemplo de esquema do modelo *XER* (eXtensible Entity-Relationship) [SEN 03] é apresentado pela Figura 3.2. As notações gráficas para os conceitos de *sequence*, *choice*, *all* e de elementos mistos são indicadas neste exemplo diretamente nas entidades do esquema. Tais abordagens, que optam por estender o ER adicionando restrições específicas do modelo de dados XML, impõem ao ER um caráter lógico. Conforme justificado previamente, considera-se que restrições específicas do XML devam ser tratadas a partir do nível lógico, independentemente do modelo conceitual utilizado.

C-XML (Conceptual XML) [EMB 04, AK 07] propõe um modelo específico para representar conceitualmente *XML Schema* [THO 04]. O modelo define objetos, relacionamentos e restrições sobre estes objetos e relacionamentos. Objetos em *C-XML* distinguem os elementos XML em léxicos e não-léxicos. Cada objeto pode ser associado a um *data frame*, onde são encontradas descrições dos valores que os objetos podem assumir, propriedades e outras restrições. O modelo permite definir objetos ou atributos para relacionamentos e cardinalidades máximas e mínimas dos objetos em um relacionamento. Além disso, é possível representar generalizações ou especializações disjuntas ou compartilhadas. A ênfase desta proposta é a tradução de um esquema *C-XML* para

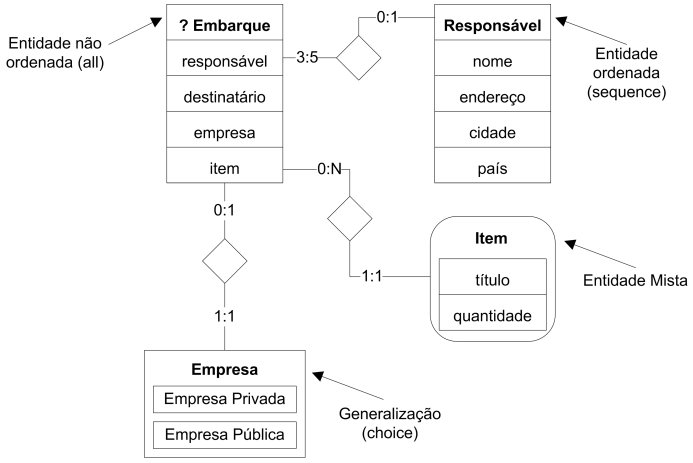

Figura 3.2: Exemplo de esquema XML do modelo XER

um esquema *XML Schema*, bem como o processo de engenharia reversa. Neste último caso, o modelo *C-XML* é utilizado para criar visões sobre um ou mais esquemas em XML Schema.

ORA-SS (Object-Relationship-Attribute for SemiStructured Data) [DOB 01] constitui um modelo hierárquico para dados XML. Este modelo não representa apenas estruturas aninhadas, mas também objetos, relacionamentos (hierárquicos ou de referência) e atributos. Além disto, o modelo mistura alguns conceitos presentes em um modelo conceitual, como a definição de cardinalidades bidirecionais em estruturas hierárquicas, relacionamentos n-ários com $n > 2$ e a distinção entre atributos de objeto e atributos de relacionamento. O modelo *ORA-SS* também permite representar restrições de ordenação em três níveis: entre instâncias de um objeto, entre valores de um atributo e na seqüência de disposição dos atributos em um objeto. Além disso, permite representar disjunções entre atributos e objetos. Quanto aos atributos, ele permite definir atributos de valores fixos com valores *default*, atributos do tipo *ANY* (atributos de conteúdo desconhecido ou heterogêneo), atributos chave, bem como atributos obrigatórios, opcionais ou multivalorados.

A modelagem por redes semânticas proposta por [CHA 02] utiliza também um modelo hierárquico para representar semanticamente objetos do domínio, seus atributos e

suas relações. Uma metodologia para projeto de esquemas XML é proposta em dois níveis: nível semântico e nível de esquema. Assim que o nível semântico é definido, é proposto um mapeamento deste nível para um esquema em *XML Schema*. O nível semântico corresponde à modelagem do domínio, sendo composto por quatro componentes principais: (i) uma série de nós atômicos e complexos representando objetos do domínio; (ii) uma série de setas direcionadas representando os relacionamentos entre esses objetos; (iii) uma série de rótulos denotando diferentes tipos de relacionamentos semânticos como (a) agregação, (g) generalização, (s) associação, e (p) propriedades dos relacionamentos; (iv) uma série de restrições definidas sobre nós (unicidade, integridade referencial, cardinalidades e restrições de domínio) e restrições sobre relacionamentos (ordenação e disjunção). Um exemplo de esquema definido em redes semânticas é apresentado pela Figura 3.3. Conforme pode ser observado neste exemplo, os tipos de relacionamento entre os conceitos (a, g, s ou p) são representados sobre uma composição hierárquica, desta forma misturando notações de um modelo conceitual com notações de um modelo lógico XML.

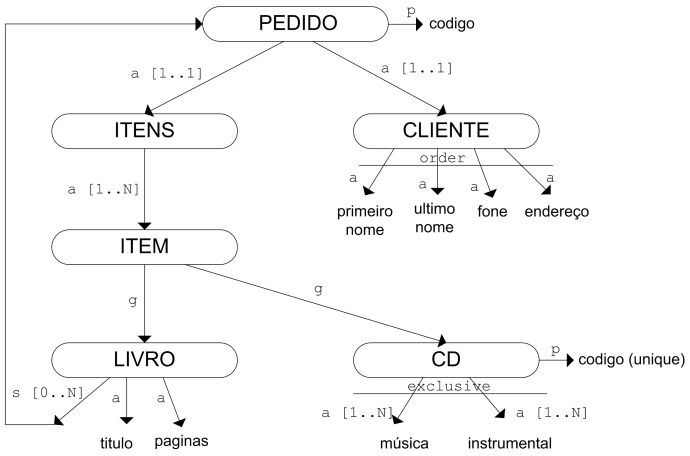

Figura 3.3: Exemplo de esquema XML modelado por redes semânticas

Na próxima seção, os trabalhos analisados são comparados quanto aos níveis de modelagem atendidos por cada proposta. Além desta comparação, são identificadas e comparadas as abordagens que apresentam metodologias de conversão de modelos conceituais para modelos XML.

3.4 Comparação dos Trabalhos Correlatos

3.4.1 Comparação Geral

O quadro comparativo apresentado na Tabela 3.1 situa cada uma das abordagens apresentadas nas etapas de projeto conceitual, lógico e de implementação de um BD XML, mostrando os respectivos modelos de dados utilizados em cada uma delas.

Tabela 3.1: Comparativo Geral entre os Trabalhos Relacionados

Abordagens	**Modelo Conceitual**	**Modelo Lógico**	**Modelo de Implementação**
Abordagens baseadas no ER			
[LEE 01]	ER	S3-Graph	-
[ELM 02]	EER	estruturas hierárquicas	XML-Schema
[CHO 03]	EER	-	DTD
[PIG 05]	ER	-	XML Schema
[FON 06]	EER	-	DTD
[LIU 06a]	EER	-	XML Schema
[WIW 06, JAG 04]	ER	MCT Schema	-
Abordagens baseadas na UML			
[ROU 02, BIR 00]	UML/ORM	UML+estereótipos	XML Schema
[CON 00, ECK 04]	-	UML+estereótipos	DTD/XML Schema
[COM 06]	-	UXS	XML Schema
[LIU 06b]	UML	XUML	XML Schema
[QUA 05]	UML	-	XML Schema
[KUD 03, KRU 03]	UML	-	DTD/XML Schema
Abordagens baseadas em novos modelos			
[MOK 06]	Modelo conceitual genérico	estruturas hierárquicas	-
[SEN 03]	-	XER	XML Schema
[PSA 01]	-	ERX	XML Schema
[MAN 04]	EReX	XGrammar	-
[EMB 04, AK 07]	C-XML	-	XML Schema
[DOB 01]	-	ORA-SS	DTD
[CHA 02]	-	Redes Semânticas	XML Schema

Nota-se que poucas abordagens atuam nas três etapas de projeto ([ELM 02], [ROU 02, BIR 00] e [LIU 06b]). Em [ELM 02], seus esquemas hierárquicos e respectivos esquemas XML gerados limitam-se a representar visões sobre um esquema EER, não considerando todas as relações semânticas concebidas na modelagem conceitual. Além disso, o modelo

hierárquico utilizado é bastante simplificado, definindo apenas uma estrutura de grafo sem considerar restrições específicas do modelo XML. Em [ROU 02, BIR 00] é proposta uma metodologia de projeto baseada na UML, identificando quais seriam os principais passos de um algoritmo de mapeamento do nível conceitual para o lógico, mas sem propô-lo concretamente. Outra abordagem que considera as três etapas de projeto utilizando UML é [LIU 06b]. Entretanto, ela define apenas alguns padrões de projeto a serem aplicados em cada uma das etapas, sem apresentar regras de transformação entre elas.

Quanto aos modelos conceituais utilizados, observa-se o grande emprego do ER e do EER, já que são modelos consagrados para este papel no projeto de BDs. Alguns autores consideram o ER pouco adequado como modelo conceitual para XML, principalmente por não possuir um suporte adequado para representar restrições sobre dados XML. Pode-se constatar isto com o surgimento de propostas que estendem o ER incorporando tais restrições. No entanto, considera-se que estas propostas atuam já no nível lógico de modelagem, pois impõem características do modelo lógico, no caso, o modelo XML.

Outra concepção é o uso da UML como modelo conceitual e de novos modelos conceituais específicos para a modelagem conceitual XML: *C-XML* e *EReX*. Estes novos modelos conceituais são modelos simples e semelhantes ao ER. Porém algumas novas construções conceituais são adicionadas à eles. No caso do *C-XML* é introduzida a representação de objetos léxicos e não-léxicos. Já o *EReX*, estende o ER para melhor representar tipos união do XML e restrições de ordenação entre entidades. Entretanto, acredita-se que o projeto de BD XML deva ser baseado em um modelo conceitual consolidado e que um novo modelo conceitual para este propósito seja desnecessário.

Quanto aos modelos lógicos, há extensões tanto do ER como da UML para impor restrições do modelo de dados XML, assim como modelos hierárquicos como o *S3-Graph*, redes semânticas e *ORA-SS*. O uso de modelos hierárquicos constitui uma escolha adequada, já que impõe a estrutura de aninhamento do modelo XML e permite, desta forma, representar mais facilmente restrições e composições permitidas para este tipo de estrutura. Porém, os modelos hierárquicos dos trabalhos citados não são completamente adequados para atuar no projeto lógico de BDs XML. Por exemplo, os modelos de *ORA-SS* e redes semânticas apresentam alguns construtores conceituais que não são diretamente

suportados pelo modelo XML, como relacionamentos n-ários com $n > 2$ em *ORA-SS* e a distinção semântica de tipos de relacionamento entre conceitos em redes semânticas. Quanto ao modelo *S3-Graph*, algumas limitações são encontradas, como por exemplo a ausência de construtores para representar atributos e a ocorrência mínima de um conceito tratado como sub-elemento de uma hierarquia de elementos.

3.4.2 Comparação entre Algoritmos de Conversão Conceitual-XML

Com o foco na conversão de esquemas conceituais para esquemas XML lógicos ou de implementação, a Tabela 3.2 compara os trabalhos relacionados que apresentam a conversão de modelos conceituais para o modelo XML. Alguns trabalhos apresentados nas seções anteriores foram desconsiderados nesta comparação por não apresentarem processos ou algoritmos definidos para o foco da conversão em questão. Os construtores dos modelos conceituais são relacionados e seus respectivos tratamentos pelos trabalhos em questão são apresentados pela Tabela 3.2. As abordagens que tratam por completo um determinado construtor conceitual são identificadas pelo símbolo (+). Construtores tratados parcialmente são identificados pelo símbolo (-). A ausência de símbolo representa o não-tratamento do construtor pela abordagem.

Tabela 3.2: Comparativo entre Algoritmos de Conversão Conceitual-XML

Abordagens	Entidades/Classes	Atributos	Generalizações	Tipos União	Relacionamentos
[LEE 01]	+	-			+
[ELM 02]	-	-			-
[CHO 03]	+	+	-		+
[PIG 05]	+	-			+
[FON 06]	+	-	-	-	+
[LIU 06a]	+	+	-		+
[WIW 06, JAG 04]	-	-			-
[ROU 02, BIR 00]	-	-			-
[QUA 05]	+	-			-
[KRU 03, KUD 03]	-	-			-
[MOK 06, EMB 01]	-				-
[MAN 04]	+	-			+
[EMB 04]	-	-			-

Considera-se que uma abordagem trata por completo o construtor de *Entidades/ Classes* quando os conceitos de *Entidade/Classe Regular*, *Entidade/Classe Fraca* e *Entidade/Classe Associativa* são devidamente considerados pelo algoritmo de conversão apresentado pela abordagem. Atributos *normais*, *identificadores*, *monovalorados*, *multivalorados* e *compostos* são conceitos devidamente tratados por abordagens que consideram por completo o construtor de *Atributos*. Quanto aos *Relacionamentos*, uma abordagem considera completamente este construtor conceitual quando relacionamentos binários e n-ários com $n > 2$ são devidamente tratados pelo processo de conversão, bem como todas as restrições impostas pela cardinalidade de um relacionamento e a participação mínima e máxima das entidades envolvidas. Em geral, todos os trabalhos comparados apresentam tratamentos para *Entidades/Classes*, *Atributos* e *Relacionamentos*, mesmo que parcialmente.

Quanto à conversão de *Generalizações* e *Tipos União*, é possível constatar uma grande lacuna no tratamento destes construtores pelas abordagens relacionadas. Para o construtor de *Generalizações*, consideram-se as restrições que podem ser impostas sobre uma hierarquia de generalização (total, parcial, disjunta, compartilhada). Além disto, casos de herança simples, múltipla e de múltiplos níveis hierárquicos devem ser também considerados. Para a conversão de *Tipos União* são consideradas as restrições total e parcial que são impostas a este conceito. Uma das justificativas para o não-tratamento destes construtores conceituais é a inexistência de tais construtores nos modelos conceituais considerados por algumas das abordagens, por exemplo, as abordagens que consideram a versão não estendida do ER como modelo conceitual ([LEE 01, PIG 05, WIW 06]).

Grande parte dos trabalhos relacionados estão focados na conversão de associações entre entidades/classes de um esquema conceitual, não apresentando estratégias de conversão para generalizações e uniões. Porém, tipos generalização e união são construtores considerados fundamentais para a modelagem conceitual de BD [SMI 77, BAT 92, ELM 00, ELM 85]. A abstração provida por construtores deste tipo é ainda mais relevante para a modelagem de dados XML, visto que provêem formas de representar conceitualmente construções comumente encontradas em modelos de dados complexos [EMB 98, MAN 04].

Em [AK 05], os autores apresentam algumas formas de representação de tipos generalização e união através dos mecanismos de derivação de tipos, grupos de substituição e elementos abstratos do XML Schema. Os autores identificam situações problemáticas para a representação de generalizações compartilhadas, de generalizações com herança múltipla e também de tipos união em geral. A solução dada pelos autores para estas problemáticas é apresentada na forma de extensões sugeridas para a linguagem *XML Schema*. A vantagem da representação sugerida pelos autores é o fato de que as relações semânticas estabelecidas no esquema conceitual através de generalizações e tipos união são mantidas pelo esquema de implementação. Porém, além de propor formas de conversão através de mecanismos bem específicos do XML Schema, a abordagem proposta em [AK 05] exige a extensão de uma linguagem de definição de esquemas XML recomendada pela *W3C*.

3.4.3 Considerações Finais

Este capítulo revisou trabalhos que apresentam propostas para a modelagem de documentos XML. Foram apresentadas metodologias baseadas nos modelos ER, UML e em novos modelos específicos para a modelagem de dados XML. Os trabalhos relacionados foram comparados quanto aos modelos utilizados para cada etapa de projeto. Além desta comparação, algoritmos de conversão Conceitual-XML foram comparados quanto aos construtores conceituais que são tratados pelas respectivas abordagens de conversão. Algumas limitações dos trabalhos relacionados foram endereçadas quanto aos modelos utilizados pelas propostas e quanto à abrangência de seus algoritmos de conversão.

Uma metodologia para o projeto lógico de BD XML é proposta no próximo capítulo deste trabalho. O objetivo de tal metodologia é propor soluções para a conversão de esquemas conceituais de alta abstração em esquemas lógicos abstratos quanto aos modelos de implementação XML. A conversão entre os modelos provê estratégias de conversão que consideram completamente todos os construtores apresentados na Tabela 3.2.

Com a finalidade de otimizar as estruturas lógicas XML, o capítulo 5 estende esta metodologia de conversão para considerar a carga de dados e operações esperadas para os documentos XML do BD. Informações de carga são utilizadas para a geração de uma

estrutura XML que possa responder de forma eficiente as principais e mais custosas operações do BD XML. Estas otimizações na estrutura são obtidas através de decisões tomadas no mapeamento de estruturas conceituais para estruturas XML, no sentido de não permitir redundância de dados e ao mesmo tempo apresentar uma estrutura adequada para representar conceitos frequentemente acessados pelas operações do BD.

Capítulo 4

Metodologia para Projeto Lógico de Banco de Dados XML

Este capítulo apresenta a metodologia desenvolvida para apoiar o projeto lógico de BD XML. A abordagem proposta por este trabalho é fundamentada na metodologia de projeto lógico em dois níveis provida em [BAT 92] com algumas variações. Em suma, a metodologia está focada na conversão de esquemas conceituais definidos pelo modelo EER em esquemas lógicos definidos por um modelo lógico XML. A Figura 4.1 ilustra a metodologia proposta, que tem como entrada um esquema conceitual EER fornecido por um usuário especialista no domínio da aplicação, ou por alguma ferramenta de projeto de BD responsável pela produção de tal esquema.

A princípio, qualquer modelo conceitual poderia ser utilizado para definir o esquema conceitual de entrada para o *Projeto Lógico*. Para tornar o *Projeto Lógico* mais genérico e para fins de exemplificação considera-se que o esquema conceitual é definido pelo modelo EER. O modelo EER é considerado um modelo adequado para representar os dados de um domínio de aplicação com um alto grau de abstração. Embora criticado por sua expressividade limitada, o EER contem os principais construtores utilizados para modelagem conceitual de softwares em geral. Em virtude destas qualidades e também pela sua simplicidade, o modelo EER pode ser considerado um modelo genérico quanto aos demais modelos conceituais existentes. Desta forma, uma abordagem de projeto lógico

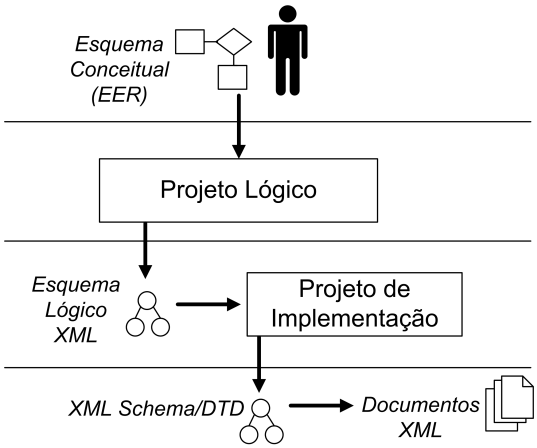

Figura 4.1: Abordagem de Projeto para Banco de Dados XML

para BDs XML baseada no modelo EER pode ser considerada flexível, pois poderia ser facilmente adaptada para trabalhar com outros modelos conceituais como a UML, por exemplo.

Durante o *projeto lógico*, o esquema conceitual é transformado em um esquema lógico XML através de um processo baseado em regras de conversão. O modelo lógico XML adotado é um modelo abstrato definido neste trabalho para representar os principais modelos de implementação XML. Consideram-se como modelos de implementação as principais linguagens de definição de esquema para XML: DTD e *XML Schema*. Embora o *projeto de implementação* seja considerado por esta metodologia, este trabalho está focado na modelagem lógica de BD XML, visto que o modelo lógico XML empregado por esta metodologia pode ser facilmente mapeado para um esquema de implementação XML e que poucos desafios podem ser endereçados no mapeamento entre estes modelos.

Este capítulo é dividido em 4 seções principais. A primeira seção se destina a apresentar o modelo lógico XML proposto. As regras de conversão dos construtores do EER para os respectivos construtores do modelo lógico são apresentadas na segunda seção. Na terceira seção é apresentado o algoritmo de conversão, o qual determina o processo de conversão através da aplicação ordenada das regras consideradas. Na última seção são apresentadas as conclusões deste capítulo.

4.1 Modelo Lógico XML

Um modelo lógico XML é definido para representar o modelo de dados XML. Tal modelo não constitui um novo formalismo, mas uma adaptação de modelos hierárquicos existentes acrescido com os construtores e as restrições do modelo de dados XML. Sua base formal e parte de sua notacão gráfica vêm dos modelos hierárquicos *Redes Semânticas* [CHA 02], *S3-Graph* [LEE 01] e ORA-SS (*Object-Relationship-Attribute for Semi-Structured Data*) [DOB 01].

O modelo lógico XML foi definido com o objetivo de abstrair as principais recomendações de linguagens de definição de esquema XML (*XML Schema* e DTD). Por este motivo, apenas conceitos essenciais e comuns a estas linguagens são considerados no modelo lógico. Sendo assim, mecanismos específicos de uma linguagem de definição de esquema não estão presentes no modelo lógico, como derivações e grupos do *XML Schema*, além de outros construtores. A abstração deste modelo permite que um esquema lógico gerado de acordo com este modelo possa ser facilmente traduzido para uma das linguagens de definição de esquema XML, tornando o processo de conversão conceitual-lógico independente de modelo de implementacão.

Os componentes base do modelo lógico XML são: atributos, elementos e relacionamentos. Na sequência cada componente base é descrito. Para fins de apresentação da notação gráfica de cada componente, o exemplo de esquema lógico XML da Figura 4.2 é utilizado.

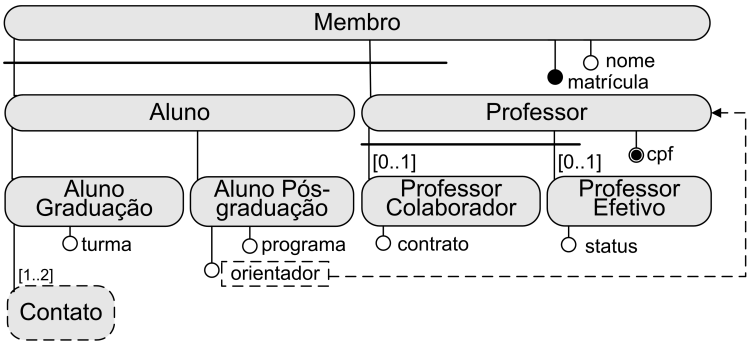

Figura 4.2: Exemplo de Esquema Lógico XML

Os atributos constituem propriedades de um elemento XML e podem ser do tipo *normal*, *único*, *identificador* ou de *referência*. Um atributo do tipo *normal* não impõe nenhuma restrição ao elemento que pertence, sendo representado graficamente por um círculo sem preenchimento. Círculos concêntricos representam atributos *únicos*, cujos valores são únicos considerando todas as instâncias de tal atributo em um documento XML. Os atributos *nome* e *cpf* dos elementos *Membro* e *Professor* na Figura 4.2 são exemplos de atributos normais e únicos, respectivamente.

Um atributo do tipo *identificador* faz parte do conjunto de atributos cujos valores identificam unicamente uma instância do elemento ao qual pertencem. Ele é representado graficamente por uma círculo com preenchimento, como é o caso do atributo *matrícula*, que representa o identificador único do elemento *Membro*.

Atributos de *referência* são utilizados para endereçar instâncias de elementos, isto é, mantêm valores de atributos identificadores. Sua representação gráfica é semelhante à de atributos normais, porém dele parte uma seta pontilhada que indica o elemento referenciado. O atributo *orientador* do elemento *Aluno Pós-graduação* que se refere ao elemento *Professor* é um exemplo de atributo de referência.

Elementos podem ser *simples* ou *complexos*. Um elemento *simples* representa informações que são definidas por um tipo de dado simples que não possui atributos nem sub-elementos definidos em seu modelo de conteúdo. O elemento *Contato* é um exemplo de elemento *simples*. Um elemento *complexo* pode ser formado por atributos e/ou elementos componentes. Os elementos componentes de um elemento complexo são organizados por um *construtor de ordem* definido para o elemento complexo que os contêm. O *construtor de ordem* de um elemento complexo pode ser *ordenado* ou *exclusivo*. O construtor *ordenado* define uma seqüência de inserção para os elementos componentes. Já o construtor *exclusivo* define uma disjunção entre os elementos componentes. O construtor de ordem *default* para elementos complexos é o *ordenado*. Os elementos *Aluno* e *Professor* são exemplos de elementos complexos com construtor de ordem *ordenado* e *exclusivo*, respectivamente.

Relacionamentos podem ser *hierárquicos* ou de *referência*. Relacionamentos hierárquicos são representados por linhas contínuas entre o conceito origem e o conceito des-

tino. Um relacionamento hierárquico define as ocorrências mínima e máxima do conceito destino no conceito origem. Por exemplo, a uma instância do elemento *Professor* pode estar associada zero ou uma (1) instância de *Professor Colaborador*. Quando não especificado no esquema, as ocorrências mínimas e máximas correspondem a 1 ([1..1]). Relacionamentos de referência são representados por linhas pontilhadas entre um atributo de referência (ou um conjunto de atributos de referência) e um elemento complexo. O relacionamento entre o atributo *orientador* e o elemento *Professor* é um exemplo deste tipo de relacionamento.

Conforme justificado previamente, um esquema lógico XML pode ser convertido para uma linguagem de definição de esquema XML como *DTD* e *XML Schema*. Esta conversão é considerada um processo direto e natural, exceto pela representação de atributos únicos, identificadores e de referência pois *DTD* e *XML Schema* provêem diferentes representações para estes conceitos. Por esta razão, opta-se por representar estes conceitos de forma abstrata no modelo lógico XML. Uma representação equivalente para os conceitos em questão deve então ser escolhida durante o projeto de implementação, considerando uma linguagem de definição de esquema específica.

4.2 Regras de Conversão EER-Modelo Lógico XML

O processo de conversão proposto por este trabalho é baseado em regras para transformar os construtores do EER em representações equivalentes no modelo lógico XML. Três grupos de regras fornecem alternativas para conversão de conceitos do EER: (i) regras para conversão de entidades e atributos; (ii) regras para conversão de tipos generalização e união; e (iii) regras para conversão de tipos relacionamento. As regras alternativas para cada grupo são definidas nas seções seguintes.

4.2.1 Conversão de Entidades e Atributos

Em geral, entidades EER são transformadas em elementos complexos e atributos EER em atributos no modelo lógico XML. Entretanto, alguns variações a esta regra geral

são aplicadas a atributos EER multivalorados e compostos. A seguir, as Regras *E*, *ANC* e *AC* são definidas para converter tipos entitdade, atributos não-compostos e atributos compostos, repectivamente. As regras *ANC* e *AC*, destinadas à conversão de atributos, são acionadas pela regra *E* durante a conversão de um tipo entidade.

Regra E (*Entidade*). A conversão de um tipo entidade *E* procede da seguinte forma:

1. gere um elemento complexo ec_E com nome *E*;
2. dado o conjunto $\{a_1, a_2, ..., a_n\}$ de atributos de *E*, para cada atributo a_i $(1 \leq i \leq n)$ faça:

   SE a_i não é um atributo composto **ENTÃO** aplique a *Regra ANC*

   SENÃO aplique a *Regra AC*.

Regra ANC (*Atributo Não-Composto*). A conversão de um atributo não-composto *A* de um tipo entidade *E* convertida como um elemento complexo ec_E procede da seguinte forma:

SE *A* é um atributo monovalorado **ENTÃO** gere um atributo XML *a* como um atributo do elemento ec_E. A ocorrência de *a* em ec_E é definida como [0-1, 1], dependendo se *A* é opcional ou obrigatório. *a* é marcado como um atributo identificador se *A* é um atributo identificador de *E*

SENÃO gere um elemento simples es_a. Um relacionamento hierárquico de ec_E para es_a é também gerado, onde a ocorrência de es_a em ec_E é definida como [0-1, N], dependendo se *A* é opcional ou obrigatório.

Regra AC (*Atributo Composto*). A conversão de um atributo composto A_c de um tipo entidade *E* convertida como um elemento complexo ec_E procede da seguinte forma:

1. gere um elemento complexo ec_{Ac} e um relacionamento hierárquico de ec_E para ec_{Ac} onde a ocorrência de ec_{Ac} em ec_E é definida como [0-1, 1-N], dependendo se A_c é opcional ou obrigatório, e se A_c é um atributo monovalorado ou multivalorado.
2. dado o conjunto $\{a_1, a_2, ..., a_n\}$ de atributos componentes de A_c, para cada atributo a_i $(1 \leq i \leq n)$ aplique a *Regra ANC*

A Figura 4.3 exemplifica a aplicação das regras para conversão de entidades e atributos EER. A entidade *ALUNO* da Figura 4.3(a) é transformada no elemento complexo *Aluno* no esquema lógico XML apresentado na Figura 4.3(b). Os atributos *matrícula*, *nome* e *email* são transformados em atributos no esquema XML. O atributo *matrícula* é marcado como um atributo identificador, visto que atua como identificador da entidade *ALUNO*. A ocorrência do atributo *email* no elemento *Aluno* é definida como [0..1] pois representa um atributo opcional da entidade em questão.

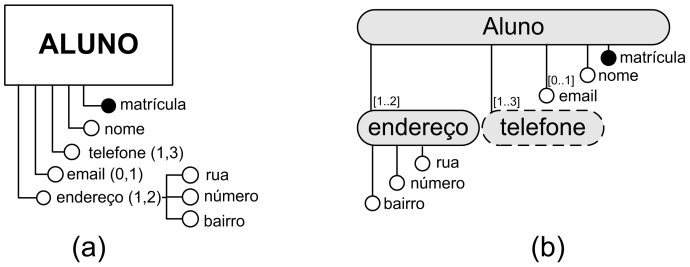

Figura 4.3: Exemplo de aplicação das Regras de Entidades e Atributos: (a) Entidades e atributos de um esquema EER; (b) Elementos e atributos gerados no modelo lógico XML pela aplicação das regras *E*, *ANC* e *AC*

O elemento multivalorado *telefone* gera um elemento simples pela aplicação da Regra *ANC*. A ocorrência do elemento *telefone* em *Aluno* é definida como [1..3], conforme a especificação deste atributo multivalorado no esquema conceitual considerado. Como exemplo da aplicação da Regra *AC*, o atributo composto *endereço* é transformado em um elemento complexo filho do elemento *Aluno*. O modelo de conteúdo do elemento *endereço* é definido pelos seus três atributos componentes: *rua*, *número* e *bairro*.

As regras de conversão definidas nesta seção são regras elementares para o processo de conversão. As regras destinadas a converter outros tipos EER fazem uso das regras para entidades e atributos, conforme apresentado nas próximas seções. Nota-se que existe uma representação direta para tipos entidade e atributos do EER no modelo de dados XML. Por esta razão, regras alternativas para estes tipos não são providas pela abordagem proposta.

4.2.2 Conversão de Hierarquias de Generalização

Nesta seção são definidas três alternativas para conversão de hierarquias de generalização do EER para composições equivalentes no modelo lógico XML. As alternativas diferenciam-se, principalmente, pelo tamanho da porção de esquema XML gerado e pela forma de representação de uma hierarquia de generalização e suas respectivas restrições. Estas regras são baseadas em estratégias para converter generalizações do EER para modelos de BDs convencionais. Estas estratégias são definidas em [BAT 92] e [HEU 04], porém são adaptadas pelo presente trabalho para considerar restrições e construções do

modelo XML. Em resumo, as três alternativas são:

1. *Generalização Modelada pela Superclasse*: Uma hierarquia de generalização é modelada por um único elemento complexo. Neste caso, superclasse e subclasses são representadas por um único elemento XML complexo;

2. *Generalização Modelada pelas Subclasses*: Cada subclasse de uma hierarquia de generalização é diretamente representada por um elemento XML complexo, sendo que a superclasse é representada na forma de atributos em cada um dos elementos gerados para as subclasses;

3. *Generalização Modelada pela Hierarquia*: Nesta alternativa, elementos XML complexos são gerados para representar a superclasse e cada uma das subclasses. Os elementos complexos que representam as subclasses são definidos como elementos filhos do elemento complexo que representa a superclasse.

A seguir, estas três alternativas de conversão são definidas na forma de regras de conversão. A aplicação das respectivas regras é discutida e exemplicada considerando cada um dos casos de restrição sobre hierarquias de generalização (total e disjunta, total e compartilhada, parcial e disjunta, parcial e compartilhada).

Regra GMS (*Generalização Modelada pela Superclasse*). Dado uma entidade E_{sp} definida como a superclasse de um tipo generalização G e o conjunto $\{E_{sb1}, E_{sb2}, ..., E_{sbn}\}$ de subclasses de E_{sp}, a conversão de G procede da seguinte forma:

1. aplique a regra *E* (*Conversão de Entidade*) para conversão de E_{sp} gerando um elemento complexo ce_{sp};

2. Para cada E_{sbi} $(1 \leq i \leq n)$ defina atributos opcionais em ce_{sp} para representar os atributos de E_{sbi}. No caso de um elemento complexo ce_{sbi} já houver sido gerado para representar uma subclasse E_{sbi}, defina tal elemento como sub-elemento de ce_{sp} com ocorrência [0..1];

   SE *G* é uma generalização compartilhada **ENTÃO** crie um elemento simples se_{tipo} como um elemento filho de ce_{sp} onde a ocorrência de se_{tipo} em ce_{sp} é definida como [0-1,N], dependendo se *G* é total ou parcial;

   SENÃO crie um atributo normal a_{tipo} em ce_{sp} onde a ocorrência de a_{tipo} em ce_{sp} é definida como [0-1,1], dependendo se *G* é total ou parcial.

A Figura 4.4 (b) apresenta um fragmento de esquema XML lógico gerado pela aplicação da *Regra GMS* sobre a hierarquia de generalização da Figura 4.4 (a). O elemento simples *tipo* gerado por esta regra, tem a função de identificar a(s) instância(s) de qual(is) sublasse(s) está-se representando. A ocorrência deste elemento simples é determinada de acordo com as restrições de totalidade que podem ser impostas sobre hierarquias de

generalização compartilhadas: [1..n] para generalizações totais e compartilhadas, [0..n] para generalizações parciais e compartilhadas. Para generalizações disjuntas, esta regra cria um atributo normal para atuar como discriminador, que apresenta ocorrência [0..1] para generalizações parciais e disjuntas, e [1..1] para generalizações totais e disjuntas. Alternativamente, a criação de um elemento/atributo discriminador pode ser desconsiderada no caso de um usuário especialista determinar que os valores dos atributos próprios das subclasses podem ser utilizados como discriminadores pela aplicação.

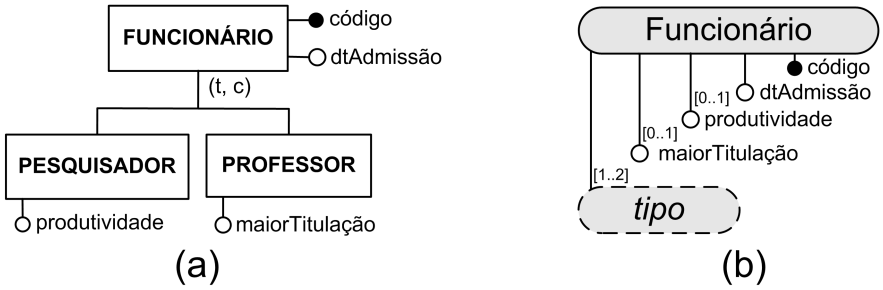

Figura 4.4: (a) Um hierarquia de generalização total e compartilhada; (b) Elementos e atributos gerados no modelo lógico XML pela aplicação da *Regra GMS*

Os atributos das subclasses são definidos como atributos opcionais do elemento complexo gerado, pois do contrário uma dependência incorreta estaria sendo gerada mesmo para generalizações totais e compartilhadas, onde nem sempre as instâncias das subclasses são representadas ao mesmo tempo (compartilhadas). Uma exceção para a definição dos atributos das subclasses no elemento da superclasse ocorre quando um elemento complexo já houver sido gerado para representar uma subclasse. Neste caso, o elemento complexo previamente criado deve ser definido como um sub-elemento opcional do elemento gerado para representação da superclasse. Casos como este ocorrem na conversão de hierarquias de generalização com múltiplos níveis hierárquicos. Um exemplo deste tratamento é apresentado na seção 4.3.

Segundo a definição de tipos generalização [BAT 92], uma subclasse herda os atributos identificadores de sua superclasse. Por este motivo considera-se que nenhum atributo identificador é definido para subclasses em um esquema EER, visto que identificadores localizados na subclasse não fazem sentido segundo esta definição.

Para a aplicação desta regra, assume-se que a distinção explícita entre as subclasses é irrelevante para a maioria das instâncias da superclasse. Esta alternativa de conversão gera a menor porção de esquema XML se comparada as demais alternativas para conversão de hierarquias de generalização. Entretanto, esta estratégia pode criar algumas inconveniências, como a possibilidade de geração de um número elevado de atributos opcionais para representar atributos de subclasses. Além disto, se houver tipos relacionamento associados com as subclasses, estes deverão ser convertidos para relacionamentos com a superclasse. A existência de relacionamentos com as subclasses é uma das principais restrições para a aplicação desta regra, visto que, conceitualmente, pode significar que a distinção entre as subclasses é relevante para a aplicação.

Regra GMSB (*Generalização Modelada pelas Subclasses*). Dado uma entidade E_{sp} definida como a superclasse de um tipo generalização e o conjunto $\{E_{sb1}, E_{sb2}, ..., E_{sbm}\}$ de subclasses de E_{sp}, gere um elemento complexo ce_{sbi} para cada subclasse E_{sbi} ($1 \leq i \leq n$) através da aplicação da regra *E* (*Conversão de Entidade*). Defina os atributos de E_{sp} como atributos obrigatórios em cada ce_{sbi}.

A Figura 4.5(b) apresenta o fragmento de um esquema lógico XML gerado pela aplicação da *Regra GMSB* sobre a generalização total e disjunta da Figura 4.5(a). As subclasses são transformadas em elementos complexos criados para representá-las, sendo os atributos da superclasse replicados para ambos os elementos de subclasse. A restrição de totalidade é garantida porque somente elementos para as subclasses são criados. Conseqüentemente, esta regra não pode ser aplicada para uma generalização com restrição de parcialidade. Além disto, esta alternativa de conversão não é recomendada para generalizações compartilhadas, visto que se uma instância de superclasse é especializada em mais de uma instância de subclasse ocorre redundância de dados quanto aos valores de atributos que representam a superclasse nos elementos das subclasses. Além disto, visto que as subclasses herdam o identificador da superclasse, valores duplicados para os identificadores dos elementos das subclasses poderiam ser gerados.

Com a aplicação da *Regra GMSB*, a noção explícita de que os elementos das subclasses representam subclasses de uma mesma entidade (superclasse) é perdida. Uma questão que pode ser considerada uma restrição a esta regra é o fato de que os relacionamentos associados com a superclasse deverão ser convertidos como relacionamentos com

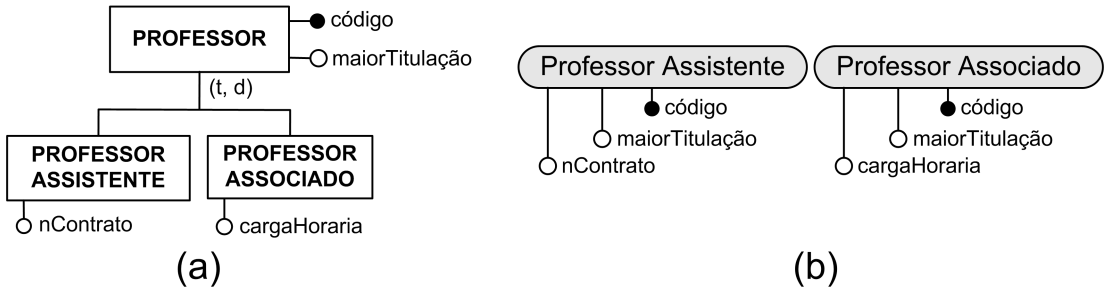

Figura 4.5: (a) Uma hierarquia de generalização total e disjunta; (b) Elementos e atributos gerados no modelo lógico XML pela aplicação da *Regra GMSB*

cada uma das subclasses. Portanto, se existem relacionamentos que envolvem diretamente a superclasse, a *Regra GMSB* pode ser uma alternativa injustificável. Um fator de menor importância é o número de atributos da superclasse. No caso em que a superclasse apresentar um elevado número de atributos, a replicação destes atributos poderia gerar um número elevado de atributos em cada um dos elementos das subclasses.

Na alternativa definida pela *Regra GMH*, a superclasse e suas subclasses são explicitamente representadas por um elemento complexo, sendo que cada elemento mantém seus próprios atributos. Relacionamentos hierárquicos são estabelecidos entre o elemento da superclasse e os elementos das subclasses, representando assim o relacionamento de especialização de forma equivalente à hierarquia conceitual. Entretanto, quando alguma subclasse de uma generalização já houver sido marcada, o relacionamento com o elemento da superclasse é estabelecido por um relacionamento de referência entre o elemento da superclasse e o elemento previamente criado para representar a subclasse marcada. Assume-se como subclasse marcada uma subclasse que anteriormente foi processada por outra hierarquia de generalização e transformou-se em um elemento filho de algum outro elemento complexo ou em atributos de um elemento complexo que não está representando apenas a subclasse em questão. O conceito de subclasse marcada é descrito com maiores detalhes pela seção 4.3.

Na definição da *Regra GMH*, um cuidado especial é tomado antes de definir o construtor de ordem do elemento da superclasse como *exclusivo* para representar a restrição de disjunção. No caso em que pelo menos uma das subclasses já houver sido *marcada*,

o construtor de ordem é definido como *ordenado*, pois nem todas as subclasses serão representadas como sub-elementos da superclasse para representar a disjunção corretamente. Uma outra situação ocorre quando outros sub-elementos forem gerados para o elemento da superclasse além dos elementos que estão sendo gerados por uma aplicação desta regra. Neste caso, é criado um sub-elemento ce_{TIPO} para o elemento da superclasse com construtor de ordem *exclusivo*, sendo os elementos das subclasses transformados em sub-elementos de ce_{TIPO}. Este novo elemento tem a função de agrupar os elementos das subclasses para que a disjunção seja corretamente estabelecida entre eles, evitando que outros sub-elementos gerados para a superclasse se envolvam nessa disjunção erroneamente. Um exemplo da necessidade de criação deste elemento agrupador é apresentado na seção 6.1 deste trabalho.

Regra GMH (*Generalização Modelada pela Hierarquia*). Dado uma entidade E_{sp} definida como a superclasse de um tipo generalização G e o conjunto $\{E_{sb1}, E_{sb2}, ..., E_{sbn}\}$ de subclasses de E_{sp}, a conversão de G procede da seguinte forma:

1. gere um elemento complexo ce_{sp} através da aplicação da regra E (*Conversão de Entidade*);

   SE nenhuma subclasse de G é dita como *marcada* **E** G é disjunta **ENTÃO**

   SE não serão gerados outros sub-elementos para ce_{sp} além dos gerados por esta generalização **ENTÃO** defina o construtor de ordem de ce_{sp} como *exclusivo*;

   SENÃO crie uma elemento complexo ce_{TIPO} como um sub-elemento de ce_{sp} e defina a ocorrência de ce_{TIPO} em ce_{sp} como [0-1,1] dependendo se G é parcial ou total. Define o construtor de ordem de ce_{TIPO} como *exclusivo*;

   SENÃO defina o construtor de ordem de ce_{sp} como *ordernado*

2. Para cada E_{sbi} ($1 \leq i \leq n$):

   SE E_{sbi} não é uma subclasse *marcada* **ENTÃO** gere um elemento complexo ce_{sbi} através da aplicação da regra E (*Conversão de Entidade*)

   SE um elemento ce_{TIPO} foi criado **ENTÃO** faça ce_{sbi} ser um elemento filho de ce_{TIPO}. A ocorrência de ce_{sbi} em ce_{TIPO} é definida como [1..1];

   SENÃO faça ce_{sbi} ser um elemento filho de ce_{sp}. A ocorrência de ce_{sbi} em ce_{sp} é definida como [1..1] se G é total e disjunta e se não existem subclasses marcadas em G. Para os demais casos, defina a ocorrência de ce_{sbi} em ce_{sp} como [0..1];

   SENÃO deixe mce_{sbi} ser um elemento previamente criado para representar E_{sbi} e gere atributo(s) de referência em mce_{sbi} referenciando-se ao identificador de ce_{sp}.

A Figura 4.6 (b) apresenta um fragmento de esquema lógico XML gerado pela aplicação da *Regra GMH* sobre as hierarquias de generalização da Figura 4.6 (a) que constitui

um caso de herança múltipla. A entidade *Funcionário* participa como subclasse na hierarquia onde a entidade *Pessoa* assume o papel de superclasse e também na hierarquia onde *Colaborador* é a superclasse. Supondo que a hierarquia da superclasse *Pessoa* é processada por primeiro, a aplicação da *Regra GMH* sobre esta hierarquia gera o elemento complexo *Pessoa* e os elementos filhos *Aluno* e *Funcionário*, conforme apresentado pela Figura 4.6 (b). Em se tratando de uma generalização parcial, a restrição de parcialidade é representada pela ocorrência [0..1] dos sub-elementos de *Pessoa*: *Aluno* e *Funcionário*. A restrição de disjunção é representada pelo construtor de ordem estabelecido como *exclusivo* para o elemento *Pessoa*.

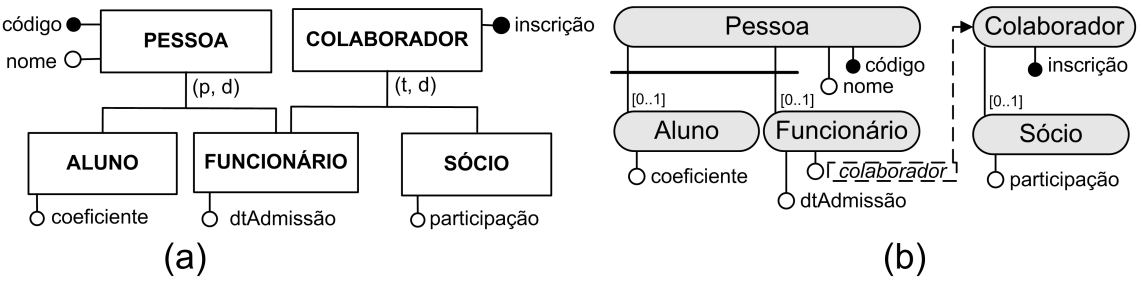

Figura 4.6: (a) Hierarquias de generalização envolvendo um caso de herança múltipla; (b) Elementos e atributos gerados no modelo lógico pela aplicação da *Regra GMH*

Na sequência, ocorre a conversão da hierarquia envolvendo *Colaborador* como superclasse. Neste caso, apenas um elemento complexo *Sócio* é criado como sub-elemento do elemento *Colaborador*. O relacionamento de especialização entre *Colaborador* e *Funcionário* é estabelecido como um relacionamento de referência a partir do elemento previamente criado para representar a entidade *Funcionário*. A representação de *Funcionário* como um sub-elemento de *Colaborador* geraria redundância de dados já que a subclasse *Funcionário* já se encontra representada (*marcada*) como sub-elemento de *Pessoa*.

Mesmo que a hierarquia da superclasse *Colaborador* constitua um caso de generalização total e disjunta, o construtor de ordem do elemento *Colaborador* não pode ser definido como *exclusivo*, visto que não foi possível representar *Funcionário* como um sub-elemento de *Colaborador*. Por esta mesma razão, a restrição de totalidade também não pode ser especificada explicitamente no esquema XML para este caso. Para uma efe-

tiva representação das restrições sobre o esquema XML gerado neste caso, as restrições de disjunção e totalidade deveriam ser estabelecidas entre o sub-elemento *Sócio* e o atributo de referência *colaborador* em *Funcionário*. Dadas as limitações do modelo XML, tais restrições não poderiam ser estabelecidas entre um elemento e um atributo.

Os relacionamentos de referência que podem ser gerados pela *Regra GMH* são sempre estabelecidos a partir das subclasses e não a partir das superclasses. A justificativa para isto é que as subclasses agregam, além do seu modelo de conteúdo específico, o modelo de conteúdo das suas superclasses. Portanto, representar os relacionamentos de referência a partir dos elementos das subclasses torna a representação desses elementos mais completa do que se fossem representados a partir da superclasse.

A alternativa provida pela *Regra GMH* gera a maior porção de esquema XML, mas torna a representação das entidades envolvidas mais flexível, principalmente quando existem tipos relacionamento associados à superclasse e às subclasses. As restrições sobre as hierarquias de generalização são explicitamente representadas por esta regra. Entretanto, alguns casos de composição de hierarquia XML dificultam esta representação, como por exemplo, no caso envolvendo a superclasse *Colaborador* do exemplo fornecido pela Figura 4.6. Além disto, para hierarquias com restrição total e compartilhada, a restrição de totalidade não pode ser adequadamente representada, visto que não há uma representação apropriada no modelo XML para garantir que pelo menos um dos sub-elementos de um elemento complexo estará presente em um documento XML adequado a este esquema.

Conforme ilustrado pela Figura 4.6, a *Regra GMH* é capaz de tratar subclasses que constituem especializações de mais de uma superclasse. Neste exemplo, a subclasse *Funcionário* herda os atributos de superclasses que pertencem a hierarquias diferentes. Esta situação pode gerar alguns conflitos clássicos de herança múltipla [EMB 98], como o conflito de identificadores. No exemplo, a subclasse *Funcionário* estaria conceitualmente herdando os identificadores *código* e *inscrição*. Este conflito é tratado pela *Regra GMH*, representando *Funcionário* como um sub-elemento de um elemento que representa uma das suas superclasses. Neste caso, o modelo de conteúdo do qual faz parte o elemento *Funcionário* é identificado apenas pelo atributo *código*. Desta forma, o elemento *Fun-*

cionário é identificado apenas por um dos identificadores das suas superclasses, sendo que o outro identificador (*inscrição*) é somente utilizado por *Funcionário* para endereçar a instância de sua outra superclasse (*Colaborador*).

4.2.3 Conversão de Tipos União

Categorias ou tipos união podem ser considerados como casos mais restritos de herança múltipla. Assim sendo, as estratégias de conversão são análogas às estratégias para transformação de hierarquias de generalização. Entretanto, alguns ajustes devem ser aplicados na conversão destes tipos, considerando, principalmente, os casos de restrição possíveis: uniões totais e uniões parciais.

Um tipo união com *restrição total* pode ser convertido similarmente como um tipo generalização total e disjunto com algumas variações [BAT 92]. Assim, aplicando-se a *Regra GMS*, cria-se um único elemento para representar a subclasse e seus respectivos atributos, juntamente com os atributos das superclasses. Na aplicação da *Regra GMSB*, somente elementos para representar as superclasses são criados e os atributos da subclasse são replicados em cada um destes elementos. Na *Regra GMH*, elementos criados para representar as superclasses tornam-se sub-elementos do elemento que representa a subclasse. Observa-se que os tratamentos definidos nas regras de generalização para a superclasse são adotados como tratamentos para a subclasse de um tipo união. As regras *UMS*, *UMSP* e *UMH* são definidas na sequência e constituem adaptações para conversão de tipos união nas regras *GMS*, *GMSB* e *GMH*, respectivamente.

Regra UMS (*União Modelada pela Subclasse*). Dado uma entidade E_{sb} definida como a subclasse de um tipo união total U e o conjunto $\{E_{sp1}, E_{sp2}, ..., E_{spn}\}$ de superclasses de E_{sb}, a conversão de U procede da seguinte forma:

1. aplique a regra *E* (*Conversão de Entidade*) para conversão de E_{sb}, gerando um elemento complexo ce_{sb};
2. Para cada E_{spi} $(1 \leq i \leq n)$ defina atributos opcionais em ce_{sb} para representar os atributos de E_{spi}. Defina os atributos identificadores de E_{spi} como atributos únicos em ce_{sb}. No caso de um elemento complexo ce_{spi} já houver sido gerado para representar uma superclasse E_{spi}, defina tal elemento como sub-elemento de ce_{sb} com ocorrência [0..1];
3. Crie um atributo identificador a_{id} para atuar como identificador de ce_{sb};
4. Crie um atributo normal a_{tipo} em ce_{sb}, onde a ocorrência de a_{tipo} em ce_{sb} é definida como [1,1].

Além de inverter o tratamento da superclasse de uma generalização para a subclasse de um tipo união, a principal adaptação da *Regra UMS* com relação à *Regra GMS* é a criação de um atributo para atuar como identificador no elemento gerado para representar a subclasse, sendo os atributos identificadores das superclasses definidos como atributos únicos. Além disto, esta regra só pode ser aplicada para tipos união total visto que a restrição de parcialidade estabelece que as superclasses não são sempre especializadas pela subclasse. O atributo discriminador gerado pela *Regra UMS* é sempre um atributo normal com ocorrência [1..1], pois estará representando uma união total (ocorrência mínima 1) onde existe uma disjunção entre as superclasses representadas pela subclasse (ocorrência máxima 1).

Regra UMSP (*União Modelada pelas Superclasses*). Dado uma entidade E_{sb} definida como a subclasse de um tipo união U e o conjunto $\{E_{sp1}, E_{sp2}, ..., E_{spn}\}$ de superclasses de E_{sb}, gere um elemento complexo ce_{spi} para cada superclasse E_{spi} $(1 \leq i \leq n)$ através da aplicação da regra *E* (*Conversão de Entidade*). Defina os atributos de E_{sb} como atributos obrigatórios em cada ce_{spi} se U é total, e como atributos opcionais se U é parcial.

A alteração estabelecida pela *Regra UMSP* sobre a *Regra GMSB* refere-se a definição dos atributos da subclasse como opcionais quando tratar-se de um tipo união parcial. Isto ocorre porque, novamente, em um tipo união parcial as superclasses não são sempre especializadas pela subclasse.

Regra UMH (*União Modelada pela Hierarquia*). Dado uma entidade E_{sb} definida como a subclasse de um tipo união U e o conjunto $\{E_{sp1}, ..., E_{spn}\}$ de superclasses de E_{sb}, a conversão de U procede como segue:

1. gere um elemento complexo ce_{sb} através da aplicação da regra *E* (*Conversão de Entidade*) para representar E_{sb};

   SE nenhuma superclasse de U é dita como *marcada* **E** U é total **ENTÃO**

   SE não serão gerados outros sub-elementos para ce_{sb} além dos gerados por este tipo união **ENTÃO** defina o construtor de ordem de ce_{sb} como *exclusivo*;

   SENÃO crie uma elemento complexo ce_{TIPO} como um sub-elemento de ce_{sb} e defina a ocorrência de ce_{TIPO} em ce_{sb} como [1,1]. Defina o construtor de ordem de ce_{TIPO} como *exclusivo*;

   SENÃO defina o construtor de ordem de ce_{sb} como *ordenado*

2. Para cada E_{spi} $(1 \leq i \leq n)$ gere um elemento complexo ce_{spi} através da aplicação da regra *E* (*Conversão de Entidade*) se E_{spi} não estiver *marcada*, e faça:

   SE U é total **E** E_{spi} não é uma superclasse *marcada* **ENTÃO**

SE um elemento ce_{TIPO} foi criado **ENTÃO** faça ce_{spi} ser um elemento filho de ce_{TIPO}. A ocorrência de ce_{spi} em ce_{TIPO} é definida como [1..1];

SENÃO faça ce_{spi} ser um elemento filho de ce_{sb}. A ocorrência de ce_{spi} em ce_{sb} é definida como [1..1];

SENÃO SE E_{sb} não está *marcada* **ENTÃO** faça ce_{sb} ser um elemento filho de cada ce_{spi}. A ocorrência de ce_{sb} em cada ce_{spi} é definida como [0..1] quando U é parcial e como [1..1] quando U é total;

SENÃO gere atributo(s) de referência em ce_{spi} referenciando-se ao identificador de ce_{sb}. Defina a ocorrência dos atributos como opcionais se U é parcial, e como obrigatórios se U é total.

3. crie um atributo identificador a_{id} para atuar como identificador de ce_{sb} se este não estiver sido definido como sub-elemento dos elementos gerados para representar as superclasses;

As principais adaptações da *Regra UMH* sobre a *Regra GMH* referem-se à geração de um atributo identificador para o elemento que representa a subclasse, e ao tratamento diferenciado para tipos união total e parcial. Segundo a *Regra UMH*, somente um tipo união total pode definir as superclasses como sub-elementos do elemento da subclasse. Isto ocorre porque estabelecer as superclasses como sub-elementos do elemento da subclasse tornaria a existência das superclasses dependente de uma instância da subclasse quando o tipo união é parcial. Tal dependência não é válida pois em uma união parcial as superclasses nem sempre são especializadas pela subclasse.

Para tipos união parcial ou para tipos união total cujas superclasses estão marcadas, um elemento representando a subclasse é definido como sub-elemento de cada elemento de superclasse. A restrição para a aplicação desta conversão é que a subclasse não pode estar marcada. Neste caso, como sempre existirá uma disjunção entre as superclasses de um tipo união, a redundância é evitada pois uma instância de subclasse nunca estará atuando como um sub-elemento de mais de uma instância de superclasse. Na última opção de conversão provida pela *Regra UMH*, os elementos das superclasses são associados ao elemento da subclasse através de um relacionamentos de referência.

A Figura 4.7 (a) apresenta um exemplo de tipo união total e os fragmentos de esquema XML lógico gerados pela aplicação das regras *UMS*, *UMSP* e *UMH*, respectivamente. Conforme apresentado no exemplo da Figura 4.7 (b), a *Regra UMS* estabelece a representação dos identificadores das superclasses como atributos únicos no elemento que representa a subclasse. Isto porque um identificador composto para o elemento *Funcionário* não faria sentido visto que, seguindo as restrições de tipos união, uma instância

do elemento *Funcionário* não pode representar ao mesmo tempo uma instância de *Pessoa* e *Colaborador*. Por esta mesma razão também não faz sentido eleger um dos identificadores das superclasses como o atributo identificador de *Funcionário*. Neste caso, um novo atributo identificador (*id_func*) foi criado para este fim. Uma situação semelhante ocorre na aplicação da *Regra UMH* na Figura 4.7 (d). Nesta aplicação da *Regra UMH*, um atributo *id_func* é criado para atuar como o identificador do elemento que representa a subclasse *Funcionário*.

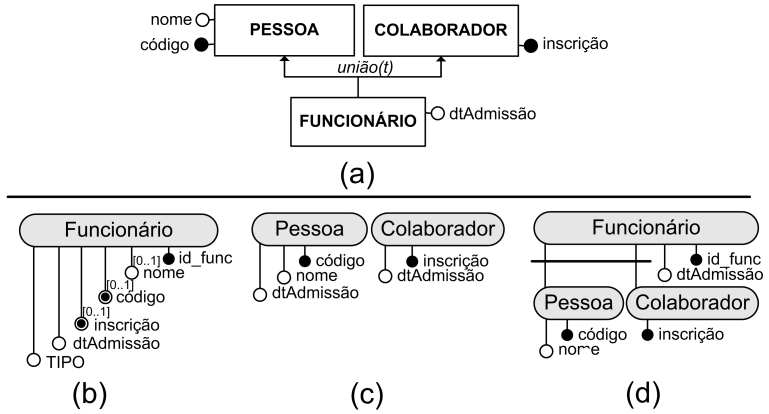

Figura 4.7: (a) Um tipo união total; (b) (c) e (d) Elementos e atributos gerados no modelo lógico XML pela aplicação das regras *UMS*, *UMSP* e *UMH*

Conforme justificado anteriormente, a *Regra UMS* não pode ser aplicada para a conversão de tipos união com restrição *parcial*. A aplicação da *Regra UMSP* gera apenas elementos para representar as superclasses e os atributos da subclasse devem ser definidos como atributos opcionais nos elementos criados, devido à restrição de parcialidade. A restrição parcial também torna necessário que os relacionamentos de especialização sejam estabelecidos através de relacionamentos hierárquicos a partir das superclasses ou através de relacionamentos por referência na aplicação da *Regra UMH* sobre um tipo união parcial. A Figura 4.8 (a) apresenta um exemplo de tipo união parcial e os fragmentos do esquema XML lógico gerado pela aplicação das regras *UMSP* e *UMH*, respectivamente. Em (c) relacionamentos hierárquicos são estabelecidos a partir de cada superclasse com a subclasse, e em (d) relacionamentos de referência são utilizados para a representação da união parcial.

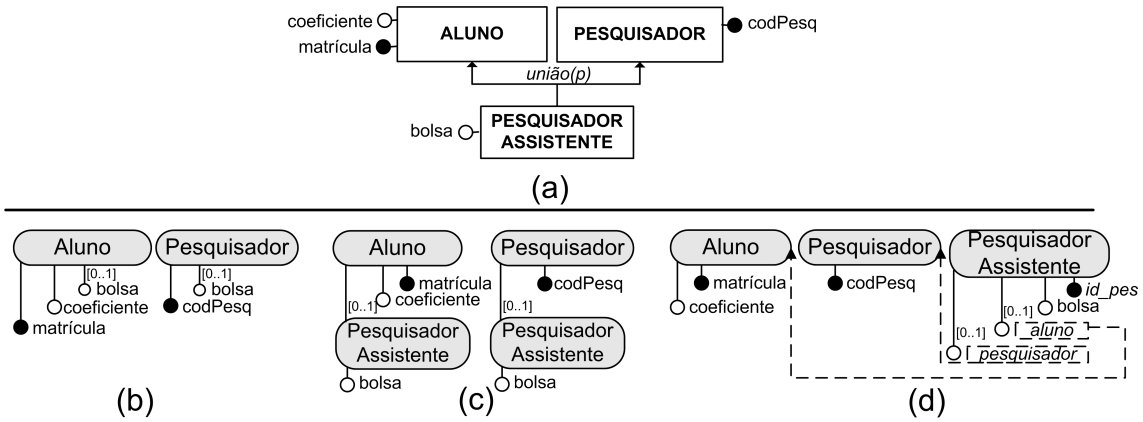

Figura 4.8: (a) Um tipo união parcial; (b), (c) e (d) Elementos e atributos gerados no modelo lógico XML pela aplicação das regras *UMSP* e *UMH*

Um *short paper* contendo a metodologia de conversão para tipos generalização e união foi publicado nos anais do *23rd ACM Symposium on Applied Computing* [SCH 08a].

4.2.4 Conversão de Tipos Relacionamento

Três regras para conversão de tipos relacionamento do EER para composições equivalentes no modelo lógico XML são definidas nesta seção. As restrições de grau e de cardinalidades mínima e máxima dos tipos relacionamento EER são consideradas para a escolha da regra de conversão adequada. Em resumo, as três regras de conversão são:

1. *Relacionamento Modelado por Elemento Único*: Um tipo relacionamento e as entidades relacionadas por ele são representadas por um único elemento complexo. Esta alternativa de conversão é aplicável apenas a relacionamentos com cardinalidade máxima 1:1;

2. *Relacionamento Modelado por Hierarquia*: Esta regra estabelece um relacionamento hierárquico entre elementos complexos criados para representar entidades relacionadas por um tipo relacionamento. Esta estratégia de conversão pode ser aplicada em relacionamentos 1:1 e 1:N;

3. *Relacionamento Modelado por Referências*: Nesta alternativa, relacionamentos de referência são gerados entre os elementos XML complexos criados para representar as entidades de um tipo relacionamento. Esta regra é aplicável para todas as restrições de tipos relacionamento.

A seguir, as três regras de conversão são definidas e suas respectivas aplicações são discutidas e exemplicadas.

Regra RMEU (*Relacionamento Modelado por Elemento Único*). Dado um tipo relacionamento binário *R* o qual relaciona as entidades E_1 e E_2, gere um elemento complexo ce_{E1} através da aplicação da regra *E* (*Conversão de Entidade*) para E_1. Defina os atributos de E_2 e *R* como atributos em ce_{E1}. Caso exista uma atributo identificador para E_2, defina-o como atributo único em ce_{E1}.

A alternativa definida pela *Regra RMEU* é aplicável apenas a tipos relacionamento com cardinalidade máxima 1:1. Para a correta aplicação desta regra exige-se que pelo menos uma das entidades apresente a participação (1,1). Os atributos desta entidade com participação (1,1) são então representados como atributos do elemento complexo que representa a outra entidade. Eventuais atributos identificadores da entidade com participação (1,1) são tratados como atributos únicos no elemento XML gerado. Este tratamento é necessário pois a existência de dois identificadores para o elemento gerado não faria sentido. Observa-se que a conversão de *entidades fracas* é diretamente suportada por esta regra, onde a entidade fraca é representada no modelo de conteúdo do elemento criado para representar a entidade forte, tendo como identificador o atributo identificador da entidade forte. A Figura 4.9(a) apresenta um exemplo de aplicação desta regra. Neste exemplo, os atributos de *Monitor* são adicionados ao modelo de conteúdo do único elemento complexo criado (*Disciplina*) devido à participação (1,1) de *Monitor* no relacionamento. Os atributos do relacionamento *monitoria* e da entidade *Monitor* são definidos como opcionais devido à participação (0,1) de *Disciplina* neste relacionamento.

Regra RMH (*Relacionamento Modelado por Hierarquia*). Dado um tipo relacionamento binário *R* que relaciona as entidades E_1 e E_2, gere um elemento complexo ce_{E1} através da aplicação da regra *E* (*Conversão de Entidade*) para E_1. Gere um elemento complexo ce_{E2} através da aplicação da regra *E* como um sub-elemento de ce_{E1} para representar a entidade E_2. A ocorrência de ce_{E2} em ce_{E1} é definida de acordo com a participação de E_1 em *R*. Defina os atributos de *R* como atributos em ce_{E2}.

A *Regra RMH* pode ser aplicada a tipos relacionamento com cardinalidade máxima 1:1 ou 1:N. Para a correta aplicação desta regra também exige-se que pelo menos uma das entidades apresente participação (1,1) no relacionamento. Nesta estratégia de conversão,

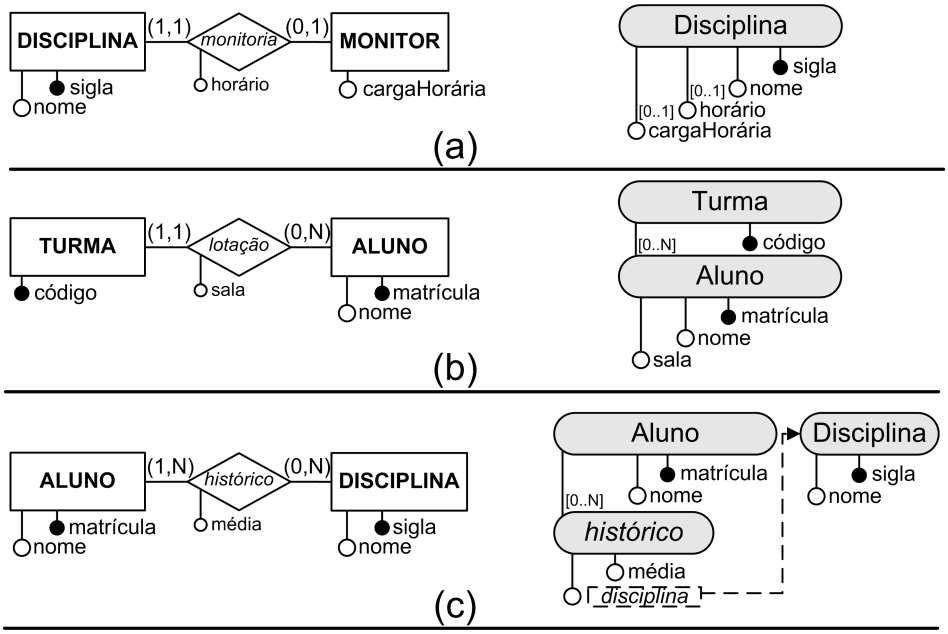

Figura 4.9: Exemplo de aplicação das regras para conversão de tipos relacionamento: (a) aplicação da *Regra RMEU*, (b) aplicação da *Regra RMH* e, aplicação da *Regra RMR*

o elemento complexo que representa a entidade de participação (1,1) é representado como um sub-elemento do elemento que representa a outra entidade. Os atributos do relacionamento são então representados como atributos no modelo de conteúdo do elemento filho. A Figura 4.9(b) apresenta um exemplo de aplicação desta regra.

Regra RMR (*Relacionamento Modelado por Referências*). Dado um tipo relacionamento R e o conjunto $\{E_1, E_2, ..., E_n\}$ de entidades relacionadas por R, para cada E_i $(1 \leq i \leq n)$ gere um elemento complexo ce_{Ei} através da aplicação da regra *E* (*Conversão de Entidade*), e:

SE R é um relacionamento binário sem atributos **E** a participação de uma das entidades em R chamada E_1 é definida como (0-1, 1), gere atributo(s) de referência a partir de ce_{E1} referenciando-se ao identificador de ce_{E2};

SENÃO gere um elemento complexo ce_R como um sub-elemento de ce_{E1} e defina os atributos de R como atributos em ce_R. Para cada E_i $(1 \leq i \leq n)$ gere atributo(s) de referência em ce_R referenciando-se ao identificador de ce_{Ei}.

A *Regra RMR* é aplicada em tipos relacionamento N:N ou em relacionamentos com grau n onde $n > 2$. Esta regra cria um elemento complexo para representar o relacionamento e o torna sub-elemento de um elemento que representa uma das entidades envolvi-

das no tipo relacionamento. Relacionamentos de referência com os elementos das demais entidades são estabelecidos a partir do elemento criado para representar o relacionamento conceitual. A Figura 4.9(c) apresenta um exemplo de aplicação desta regra sobre um tipo relacionamento N:N. Neste caso, o elemento do relacionamento *histórico* é representado como um sub-elemento de *Aluno* e um relacionamento de referência com *Disciplina* é estabelecido a partir do elemento *histórico*.

A redundância de dados é evitada na aplicação da *Regra RMR* sobre relacionamentos N:N quando um elemento para representar o relacionamento é criado como um sub-elemento de uma das entidades do relacionamento. Conforme definido pela regra, o relacionamento com as demais entidades participantes é estabelecido através de relacionamentos de referência entre o elemento criado para representar o relacionamento e os elementos das demais entidades. Isto ocorre porque representar uma das entidades como sub-elemento do elemento da outra entidade tornaria possível que os dados relativos a uma instância do sub-elemento apareçam repetidas vezes para as instâncias do elemento pai, dada a participação máxima N de ambas as entidades.

A *Regra RMR* também é acionada em casos em que as regras *RMEU* e *RMH* não podem ser aplicadas a relacionamentos 1:1 e 1:N. Nestes casos, um relacionamento de referência é estabelecido diretamente da entidade com participação máxima 1 para o elemento que representa a outra entidade. Desta forma não é criado um elemento para representar o relacionamento, minimizando o número de elementos gerados pela aplicação da regra. No entanto, a *Regra RMR* estabelece que este tratamento não pode ser assumido quando existir atributos no relacionamento. Neste caso, a criação de um elemento para representar o relacionamento é necessário para encapsular seus atributos.

Um processo de conversão automático para gerar um esquema lógico XML a partir de um esquema EER é apresentado na próxima seção. Este processo analisa o esquema conceitual e suas restrições para determinar qual das regras de conversão deve ser aplicada em cada caso. No entanto, um processo semi-automático de conversão poderia ser considerado. Neste caso, um usuário especialista teria a função de escolher e determinar a estratégia de conversão para cada fragmento conceitual, considerando as regras propostas. Algumas verificações precisam ser executadas para a aplicação das regras, tanto por um

processo automático quanto por um semi-automático. Um exemplo poderia ser a conversão de um relacionamento 1:1 onde ambas as entidades têm participação opcional no relacionamento. Neste caso, a *Regra RMR* deve ser aplicada, visto que as regras *RMEU* e *RMH* necessitam da existência de uma entidade com participação (1,1) para a correta aplicação. Um outro exemplo de decisão que deve ser tomada pelo processo é a escolha da entidade que deverá representar o elemento pai do elemento criado para representar o relacionamento na aplicação da *Regra RMR*.

4.3 Processo de Conversão

Um processo automático para conversão de esquemas EER em esquemas lógicos XML é proposto nesta seção. O processo automático estabelece a ordem de conversão dos fragmentos de um esquema EER assim como a regra de conversão que deve ser aplicada sobre cada fragmento considerado. Além do objetivo de representar integralmente um esquema EER em um esquema XML, o processo tem por finalidade produzir um esquema lógico XML compacto e livre de redundância de dados. Um esquema lógico XML compacto é obtido pela aplicação, sempre que possível, das regras de conversão que geram a menor porção de esquema XML. O detalhamento do processo de conversão é fornecido pelo *Algoritmo 1* (*Conversão EER-XML Lógico*).

O *Algortimo 1* é composto pelos procedimentos *converterGeneralizacoesUnioes* e *converterRelacionamentos* que são acionados pelo processo nesta seqüência. Tipos generalização e união são convertidos primeiramente devido à maior possibilidade de aplicar otimizações em estruturas XML geradas para representar hierarquias de generalização. Em muitos casos é possível, por exemplo, representar uma hierarquia de generalização através de um único elemento XML. Os fragmentos gerados pela conversão de tipos generalização e união são então considerados juntamente com o esquema conceitual para a conversão de tipos relacionamento, indicado na linha 2 do Algoritmo 1.

Após a execução dos procedimentos, a estrutura preliminar do esquema lógico XML é obtida. O esquema final é definido após a eleição do elemento *root* do esquema lógico. Na linha 3 do algoritmo, todos os elementos complexos de primeiro nível gerados

no esquema são identificados e armazenados pela variável *EP*. Elementos complexos de primeiro nível do esquema são todos os elementos que não possuem um elemento pai. No caso de existir apenas um elemento de primeiro nível, tal elemento é definido como o elemento *root* do esquema lógico XML. Caso contrário, um novo elemento (*root*) é criado e todos os elementos de primeiro nível se tornam sub-elemento de *root*.

Algoritmo 1: Conversão EER-XML Lógico

Entrada: Um esquema conceitual EER - *EC*

Saída: Um esquema Lógico XML - *EL*

1 $XGU \leftarrow converterGeneralizacoesUnioes(EC)$;
2 $XR \leftarrow converterRelacionamentos(EC, XGU)$;
3 *EP* ← conjunto $\{ec_1, ..., ec_n\}$ de elementos complexos de primeiro nível de *XR*;
4 **se** | *EP* | == *1* **então**
5 Faça ec_1 ser o elemento *root* de *EL*;
6 **senão**
7 Gere um elemento complexo ec_r para ser o elemento *root* de *EL*;
8 Torne todos os elementos de *EP* sub-elementos de ec_r;
9 A ocorrência de cada $ec_i (1 \leq i \leq n)$ em ec_r é definida como [0..N];
10 **fim**

A conversão de tipos generalização e união é definida pelo procedimento 1 (*converterGeneralizacoesUnioes*). A linha 2 do procedimento é responsável por ordenar os tipos generalização/união que ocorrem em uma hierarquia de múltiplos níveis, caso exista. O objetivo desta ordenação é fazer com que a conversão de cada tipo generalização/união inicie pelos tipos no nível mais inferior da hierarquia e prossiga até os tipos no topo da hierarquia. Após a ordenação, cada tipo generalização/união é convertido, de acordo com a ordem estabelecida. Na conversão de cada tipo, uma regra de conversão é escolhida para ser aplicada. O critério de escolha das regras de conversão é pela regra que gera a menor porção de esquema XML e que é capaz de representar as restrições do fragmento EER que está sendo analisado. Assim sendo, as regras *GMS*, *GMSB* e *GMH* são avaliadas nesta seqüência para a conversão de um tipo generalização e as regras *UMS*, *UMSP* e *UMH* são avaliadas nesta seqüência para a conversão de um tipo união.

As regras *GMS* e *GMSB* possuem algumas restrições para a sua aplicação, conforme mostra o procedimento 1. A primeira restrição considerada para a aplicação da regra *GMS* é a exigência de que nenhuma das subclasses esteja *marcada*. Uma entidade

Procedimento 1: *converterGeneralizacoesUnioes*

Entrada: Um esquema conceitual EER - *EC*

Saída: Fragmentos gerados no Modelo Lógico XML - *XGU*

```
1  GU ← o conjunto {gu1, ..., gun} de tipos generalização e união de EC;
2  Ordene GU de forma que os primeiros tipos da lista sejam os tipos generalização e união presentes nos níveis mais inferiores
   de uma hierarquia de múltiplos níveis;
3  GU' ← a lista ordenada de GU;
4  para cada gui ∈ GU' faça
5      se gui é um tipo generalização então
6          se (Não há subclasse marcada em gui) E (Não há tipos relacionamento associados às subclasses de gui) E (Não
           há subclasses com mais de uma superclasse) então
7              Aplique a Regra GMS;
8              Marque todas as subclasses de gui;
9          senão
10             se (gui é total e disjunta) E (Não há subclasse marcada em gui) E (Não há tipos relacionamento associados
               à superclasse de gui) então
11                 Aplique a Regra GMSB;
12             senão
13                 Aplique a Regra GMH;
14                 Marque todas as subclasses de gui;
15             fim
16         fim
17     senão
18         se (gui é total) E (Não há superclasse marcada em gui) E (Não há tipos relacionamento associados às
           superclasses de gui) E (Não há superclasses com mais de uma superclasse) então
19             Aplique a Regra UMS;
20             Marque todas as superclasses de gui;
21         senão
22             se (Não há superclasse marcada em gui) E (Não há tipos relacionamento associados à subclasse de gui)
               então
23                 Aplique a Regra UMSP;
24             senão
25                 Aplique a Regra UMH;
26                 Marque todas as superclasses de gui;
27             fim
28         fim
29     fim
30 fim
```

é dita *marcada* pelo procedimento quando ela passa a ser representada no modelo de conteúdo de um elemento que não está somente representando a entidade em questão (entidade marcada). Essa situação ocorre justamente para as subclasses de uma hierarquia

de generalização que foi processada pela regra *GMS* (linha 8 do procedimento), onde as subclasses passam a ser representadas como atributos em um elemento criado para representar a superclasse e todas as suas subclasses. Uma entidade também é *marcada* quando o elemento que a representa passa a ser representado como um sub-elemento de algum outro elemento do esquema. Esta situação ocorre para tipos generalização convertidos pela regra *GMH* (linha 14 do procedimento), onde as subclasses são transformadas em sub-elementos do elemento que representa a superclasse. Além desta exigência, a regra *GMS* estabelece que não devem existir tipos relacionamento associados às subclasses. Esta restrição é imposta pois assume-se que a existência destes relacionamentos indica que a distinção entre superclasse é subclasses é relevante para a aplicação. No entanto, em um processo semi-automático de conversão esta poderia ser uma decisão tomada por um usuário especialista ou até mesmo indicada na forma de parâmetros para um processo automático. A inconveniência gerada pela existência de relacionamentos com subclasses na aplicação desta regra é apresentada pela definição da mesma na seção 4.2.2.

A regra *GMS* também não pode ser aplicada quando alguma das subclasses possuir mais que uma superclasse. Isso ocorre porque, em casos de herança múltipla, a subclasse que possui mais que um pai deve gerar um relacionamento de referência para representar a especialização com pelo menos uma das suas superclasses. Neste caso, se a aplicação da regra *GMS* fosse permitida, o relacionamento de referência seria estabelecido a partir de um elemento que representa não apenas a subclasse, mas também uma das suas superclasses e as demais subclasses desta superclasse (fusão de entidades de uma hierarquia de generalização).

A regra *GMSB* só pode ser aplicada para generalizações totais e disjuntas, conforme justificado na seção 4.2.2. Esta regra também não é aplicada quando existem subclasses marcadas. Além disto, tipos relacionamento com a superclasse não são permitidos. Finalmente, quando as regras *GMS* e *GMSB* não puderem ser aplicadas, a última opção de conversão é provida pela regra que gera a maior porção de esquema, ou seja, a regra *GMH*. Observa-se que apenas a regra *GMH* lida com entidades marcadas. Isto ocorre porque entidades marcadas já estão sendo representadas no modelo de conteúdo de um elemento do esquema XML, o que impede de representá-las novamente no modelo de

conteúdo de um outro elemento pela aplicação das regras *GMS* e *GMSB*.

As linhas 17-29 do procedimento 1 são destinadas à conversão de tipos união. As mesmas restrições aplicadas para as regras *GMS*, *GMSB* e *GMH* são semelhantemente aplicadas as regras *UMS*, *UMSP* e *UMH*. Porém algumas variações ocorrem na aplicação da *Regra UMS* e da *Regra UMSP*. Conforme justificado pela seção 4.2.3, apenas uniões totais são convertidas pela *Regra UMS*. Na *Regra GMSB*, devido a restrição imposta pelos fragmentos gerados verifica-se se o tipo generalização é total e disjunto. Diferentemente, na *Regra UMSP* esta verificação não é realizada visto que, a princípio, tal regra pode ser aplicada tanto para uniões totais quanto para parciais.

Para fins de exemplificação da execução do procedimento 1, considere o esquema conceitual da Figura 4.10 (a) e os fragmentos gerados pelo procedimento em questão no modelo lógico XML (b). Em virtude da existência de uma hierarquia de múltiplos níveis, a conversão é iniciada pelo nível mais inferior da hierarquia. Visto que as generalizações envolvendo *E* e *H* estão no inferior da hierarquia, e ambas no mesmo nível hierárquico, a ordem de conversão entre elas é irrelevante para o procedimento em questão. Neste exemplo, escolheu-se começar pela generalização envolvendo *E* como superclasse. Como trata-se de um caso de herança múltipla, a regra *GMS* não pode ser aplicada. A regra *GMSB* também não é aplicável pois trata-se de uma generalização parcial e disjunta. Sendo assim, a regra *GMH* é então aplicada e em seguida reaplicada para a conversão da generalização envolvendo *H*. Neste último passo, um relacionamento de referência é gerado entre os elementos *G* e *H*.

Os demais tipos generalização são processados seguindo a ordenação da hierarquia de múltiplos níveis. A generalização envolvendo a superclasse *C* é convertida pela regra *GMS*. Neste caso, o atributo da subclasse *D* é representado como atributo no elemento *C* e o elemento *E* é transformando em um sub-elemento de *C*, visto que um elemento para representar a entidade *E* já havia sido previamente gerado. A generalização envolvendo *A* no topo da hierarquia constitui a última conversão a ser processada. A regra *GMSB* é aplicada neste caso pois o relacionamento *R2* associado a subclasse *C* impede a aplicação da regra *GMS*.

Após a conversão de tipos generalização e união, a conversão de tipos relaciona-

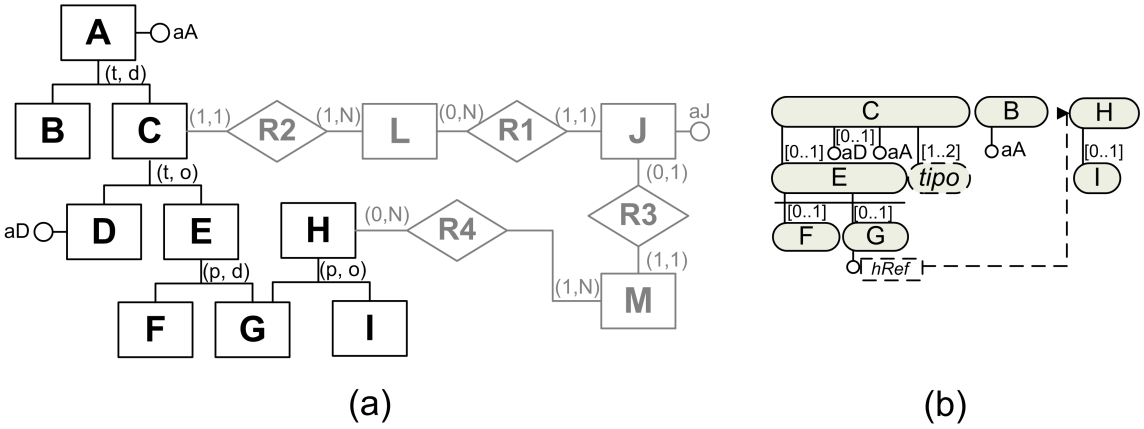

Figura 4.10: (a) Um exemplo de esquema EER e; (b) Elementos XML gerados pela conversão dos tipos generalização do esquema EER

mento é executada. O procedimento *converterRelacionamentos* inicialmente requisita a ordenação dos tipos entidades do esquema conceitual ao procedimento *ordenarEntidades*. Esta ordenação tem por finalidade guiar a conversão de tipos relacionamento através de caminhos no esquema EER determinados pelos *fechamentos funcionais completos* das entidades do esquema. Um *fechamento funcional completo* é determinado pela lista de entidades que podem ser alcançadas a partir de uma entidade de partida através de caminhos determinados pela participação (1,1) das entidades do caminho. Como exemplo, considere o esquema conceitual da Figura 4.11 (a). O *fechamento funcional completo* da entidade *L* é determinado pelas entidades *L*, *C*, *J* e *M* visto que *L* tem associação (1,1) com a entidade *C* e com a entidade *J*, que por sua vez tem associação (1,1) com *M*. Os *fechamentos funcionais completos* das demais entidades são apresentados pela Tabela 4.1. Este levantamento é realizado pela linha 2 do procedimento 2.

Após identificados os *fechamentos funcionais completos* das entidades do esquema conceitual, é gerada uma lista de entidades (*LE*) pela linha 4 do procedimento 2. A lista é composta por todas as entidades que participam em mais de um *fechamento funcional completo*. Estas entidades devem estar ordenadas em *LE* de forma que as entidades que participam em um maior número de *fechamentos funcionais completos* apareçam por primeiro na lista. De acordo com a tabela 4.1, *LE = {M, J, C}* visto que *M* participa em 3

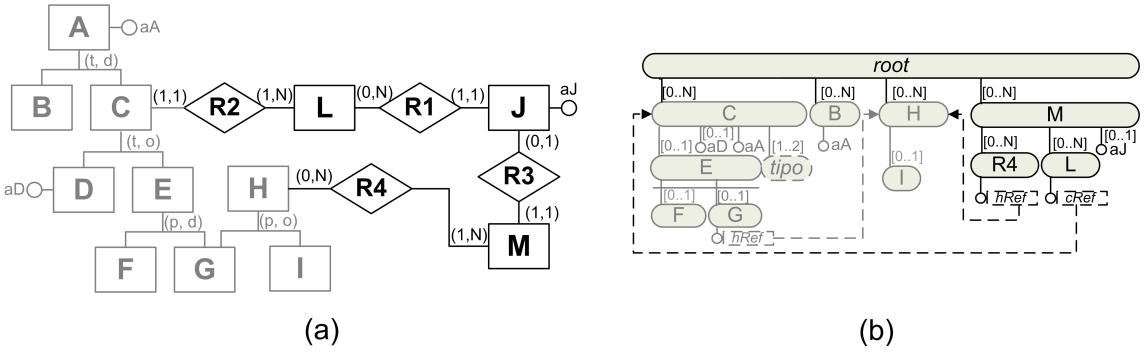

Figura 4.11: (a) Um exemplo de esquema EER e; (b) Elementos XML gerados pela conversão dos tipos relacionamento do esquema EER

Tabela 4.1: Fechamentos funcionais completos das entidades do esquema EER da Figura 4.11

Fechamentos Funcionais Completos	
A = {A}	G = {G}
B = {B}	H = {H}
C = {C}	I = {I}
D = {D}	J = {J, M}
E = {E}	L = {L, C, J, M}
F = {F}	M = {M}

Procedimento 2: *ordenarEntidades*

Entrada: Um esquema conceitual EER - *EC*

Saída: Uma lista ordenada das entidades de *EC*- *LE*

1 $E \leftarrow$ o conjunto $\{e_1, ..., e_n\}$ de tipos entidade de *EC*;
2 Compute os *fechamentos funcionais completos* de cada entidade de *E*;
3 Ordene as entidades de *E* de forma que as entidades que estão presentes no maior número de *fechamentos funcionais completos* apareçam primeiro;
4 *LE* $\leftarrow$ lista das entidades ordenadas de *E* não incluíndo as entidades que participam em apenas um *fechamento funcional completo*;
5 Ordene as entidades descartadas (não presentes em *LE*) de forma que as entidades que participam em um maior número de relacionamentos apareçam primeiro;
6 Anexe em *LE* as entidades descartadas e agora ordenadas;

fechamentos, e *J* e *C* em dois. As entidades remanescentes são então adicionadas ao fim da lista *LE*, de forma que as entidades que possuem um maior número de relacionamentos aparecem por primeiro. Seguindo o exemplo, a nova lista é obtida: *LE = {M, J, C, L, H, A, B, C, D, E, F, G, I}*.

Procedimento 3: *converterRelacionamentos*

Entrada: Um esquema conceitual EER - *EC*;
Fragmentos gerados pelo procedimento *converterGeneralizaçõesUniões - XGU*
Saída: Fragmentos gerados no Modelo Lógico XML - *XR*

1 $LE \leftarrow ordenarEntidades(EC)$;
2 $R \leftarrow$ o conjunto $\{r_1, ..., r_n\}$ de tipos relacionamento de *EC*;
3 **repita**
4 **se** *LE não está vazia* **então**
5 Remova a primeira entidade da lista *LE* e defina-a como uma *entidade de continuação*;
6 **senão**
7 Defina como uma *entidade de continuação* uma entidade participante de um tipo relacionamento desmarcado de *R*;
8 **fim**
9 **enquanto** *existir um* $r_i \in R$ *desmarcado envolvendo uma entidade de continuação* **faça**
10 $E_1 \leftarrow$ uma entidade de continuação envolvida em r_i;
11 **se** *(*r_i *está associado a uma entidade associativa cujo relacionamento interno ainda não foi processado)* **então**
12 Vá para o próximo laço;
13 **fim**
14 **senão se** *(a máxima cardinalidade de* r_i *é 1:1) E (A participação de* E_2 *em* r_i *é (1,1)) E (*E_2 *está desmarcada)* **então**
15 Aplique a *Regra RMEU*;
16 Marque E_2;
17 Defina E_2 como uma *entidade de continuação*;
18 **fim**
19 **senão se** *(*r_i *é binário) E (A participação de* E_2 *em* r_i *é (1,1)) E (*E_2 *está desmarcada) E (*E_1 *!=* E_2*)* **então**
20 Aplique a *Regra RMH*;
21 Marque E_2;
22 Defina E_2 como uma *entidade de continuação*;
23 **fim**
24 **senão**
25 Aplique a *Regra RMR*;
26 **fim**
27 Marque r_i;
28 **fim**
29 **até** *que todo* $r_i \in R$ *esteja marcado* ;

O conceito de *fechamento funcional completo* foi definido por Mok, W. et al. em [MOK 06]. A abordagem de ordenação de entidades e conversão de relacionamentos proposta pelos procedimentos 2 e 3 constitui uma variação da metodologia para geração de documentos XML compactos proposta em [MOK 06]. As variações incluem a consideração da participação mínima das entidades em um relacionamento, bem como o uso de

regras de conversão e relacionamentos de referência para evitar a redundância de dados nos documentos XML conformados ao esquema que está sendo produzido.

Dando sequência ao procedimento 3, a repetição iniciada na linha 3 do procedimento é executada até que todo o tipo relacionamento do esquema tenha sido convertido (marcado). Cada laço desta estrutura de repetição gera uma sub-árvore do esquema lógico, que ao final é associada ao elemento *root*. A entidade de partida que gera o elemento de primeiro nível da sub-árvore é selecionada nas linhas 5 ou 7. A primeira entidade é removida da lista de entidades ordenadas *LE* para que ela inicie a conversão de seus relacionamentos. Caso a lista *LE* esteja vazia, alguma das entidades de um relacionamento não marcado é escolhida. Esta entidade de partida é marcada como uma *entidade de continuação*. Algumas entidades de continuação são marcadas durante o processo para que o procedimento percorra corretamente os *fechamentos funcionais* da entidade de partida. O laço de repetição iniciado na linha 9 converte todos os relacionamentos envolvidos no fechamento da entidade de partida. Cada relacionamento é avaliado dentro de cada laço, e uma regra de conversão é escolhida para execução. Antes da aplicação de alguma das regras, o procedimento verifica se o tipo relacionamento selecionado é associado a alguma entidade associativa cujo relacionamento interno ainda não foi processado. Caso esta condição seja verdadeira (linha 11 do procedimento 3), o procedimento interrompe o laço e passa a processar o próximo tipo relacionamento. Este tratamento garante que a entidade associativa seja primeiro resolvida, sendo os relacionamentos associados a ela processados em algum passo seguinte.

A regra *RMEU* é aplicada para relacionamentos 1:1 quando a outra entidade E_2 possuir participação (1,1) e não estiver marcada. A regra *RMH* é aplicada para relacionamentos binários quando E_2 possuir participação (1,1) e não estiver marcada. Ambas as regras, quando aplicadas, definem E_2 como uma entidade de continuação para dar sequência à navegação do fechamento funcional da entidade de partida. Além disto, E_2 é marcada para impedir que outro relacionamento defina-a como elemento filho ou atributo de um outro elemento. Em outras palavras, evita-se a geração de redundância de dados no esquema XML. A regra *RMR* é aplicada para os casos não atendidos pelas regras *RMEU* e *RMH*.

Para a aplicação da *Regra RMH*, verifica-se se as entidades envolvidas no relacionamento binário não são iguais. Em outras palavras, é verificada a existência de um auto-relacionamento. Ocorrendo este tipo de relacionamento, a aplicação da *Regra RMH* é descartada, visto que a aplicação desta regra geraria redundância ao representar uma entidade como um sub-elemento de um elemento que representa a própria entidade. Para casos de auto-relacionamento, as regras *RMEU* e *RMR* poderão ser aplicadas.

Considerando o exemplo da Figura 4.11 e a lista *LE*, o procedimento inicia definindo *M* como uma entidade de continuação na linha 5. Na sequência, todos os relacionamentos de *M* (*R3*, *R4*) são convertidos dentro do laço de repetição da linha 9. *R3* é convertido pela regra *RMEU* e *R4* pela regra *RMR*. Como todos os relacionamentos da entidade de continuação *M* foram processados e como a entidade *J* foi definida como entidade de continuação durante a conversão de *R3*, o procedimento segue a conversão pelo relacionamento *R1* de *J*. Neste caso a regra *RMH* é aplicada e *L* é definido como um sub-elemento de *M* pois *J* passou a ser representada pelo modelo de conteúdo do elemento *M* após a conversão de *R3*. A entidade *L* é também marcada como entidade de continuação e o processo segue convertendo *R2*. A regra *RMR* é aplicada neste caso, e o relacionamento *R2* é representado através do relacionamento de referência estabelecido a partir do elemento *L* para o elemento *C*. Nesta última conversão, um relacionamento de referência foi necessário visto que *L* não poderia tornar-se um sub-elemento de *C* pois foi previamente marcada durante a conversão de *R1*.

4.4 Considerações Finais

Este capítulo apresentou um conjunto de regras e um processo de conversão adotados pela metodologia de projeto proposta para transformação de esquemas conceituais em esquemas lógicos XML. O algoritmo proposto estabelece um processo de conversão automático que analisa as restrições sobre o esquema conceitual para determinar a regra de conversão a ser aplicada sobre cada fragmento. O algoritmo tem por finalidade gerar um esquema lógico XML livre de redundância, com o menor volume de elementos possível e que represente integralmente um esquema conceitual EER. Um processo de conversão

semi-automático pode também ser considerado neste contexto. Neste caso, a análise e as decisões definidas no algoritmo proposto poderiam servir para guiar o usuário na escolha das regras de conversão mais apropriadas, além de conduzir a ordem de conversão dos fragmentos do esquema conceitual.

A redundância de dados é evitada na geração de esquemas XML em dois pontos principais da abordagem de conversão. Primeiramente não permitindo que um conceito seja representado por mais de um elemento do esquema XML. Este controle é efetuado pelos procedimentos através do conceito de *entidades marcadas*. Uma aparente exceção pode ser encontrada na aplicação da *Regra GMSB*, onde a superclasse é representada por todos os elementos que representam suas subclasses. No entanto esta representação não habilita a redundância de dados pois, conforme estabelecido pelo procedimento 1, esta regra só pode ser aplicada quando tratar-se de um tipo generalização total e disjunto. Isto porque a disjunção entre as subclasses especifica que uma instância da superclasse só será representada por uma das subclasses. O segundo tipo de tratamento aplicado pela abordagem para evitar a redundância esta relacionado a representação de tipos relacionamento com cardinalidade máxima N:N. Nestes casos, o processo de conversão cria um elemento para representar o relacionamento como um sub-elemento de uma das entidades do relacionamento, e estabelece relacionamentos de referência entre o elemento do relacionamento com as demais entidades participantes. Isto ocorre porque representar uma das entidades como sub-elemento do elemento que representa a outra entidade tornaria possível que os dados relativos a uma instância do sub-elemento apareçam repetidas vezes para as instâncias do elemento pai, dada a participação máxima N de ambas as entidades.

O processo de conversão definido por este capítulo considera todos os construtores do EER, incluíndo tipos união, entidades associativas, auto-relacionamentos e entidades fracas. Alguns ajustes de usuário podem ser aplicados após a geração do esquema lógico XML. Em especial, definindo nomes mais adequados para elementos e atributos produzidos pelo processo, como por exemplo o atributo ou elemento simples discriminador gerado pela aplicação das regras *GMS* e *UMS*.

O processo de conversão automático descrito por este capítulo foi implementado em conjunto com este trabalho e produziu a ferramenta *EER-XML* [LIM 08a]. Esta ferra-

menta tem como entrada um documento XML que representa um esquema EER definido pelo aplicativo *ModeladorER* [CAN 05], e é capaz de produzir esquemas XML lógicos e em DTD ou XML Schema a partir de um esquema conceitual. A ferramenta *EER-XML* encontra-se em fase de testes, sendo necessários alguns aprimoramentos em sua interface gráfica. Um artigo descrevendo o processo de conversão foi publicado nos anais do *Workshop on Ontologies and Metamodeling in Software and Data Engineering* [SCH 08c]. Além deste, dois outros artigos relacionados a detalhes da ferramenta foram publicados nos anais da *IV Escola Regional de Banco de Dados* [LIM 08c] e na *Sessão de Pôsteres do Simpósio Brasileiro de Banco de Dados* [LIM 08b].

No capítulo seguinte, este processo de conversão é estendido para considerar informações de carga estimadas para o BD XML. Estas informações são utilizadas pelo processo para geração de estruturas XML otimizadas capazes de responder com mais eficiência às operações esperadas para o BD.

Capítulo 5

Projeto Dirigido por Informações de Carga de Dados

Conforme o processo de conversão proposto no capítulo anterior, alguns relacionamentos entre conceitos de um esquema conceitual não podem ser mapeados para relacionamentos hierárquicos em um esquema XML. Isto ocorre porque esquemas conceituais podem definir grafos inclusive com ciclos, o que impossibilita a representação em uma estrutura em árvore como denotam os esquemas XML. Nestes casos, relacionamentos de referência são necessários para representar relacionamentos entre conceitos de um esquema conceitual em um esquema XML. Embora em muitos casos necessários, relacionamentos de referência geram esquemas XML fragmentados que em geral prejudicam o desempenho de consultas sobre documentos XML conformados a estes esquemas.

Quando uma consulta necessita recuperar elementos que estão associados por um relacionamento de referência em um documento XML, a execução de *junções por valor* é necessária. Em uma *junção por valor*, valores de atributos de referência de um elemento são comparados com valores de atributos identificadores de outro elemento. Diferentemente de *junções por estrutura*, onde se recupera os sub-elementos de um certo elemento, *junções por valor* tornam a execução de consultas custosa na grande maioria dos casos em que são aplicadas.

Como exemplo, considera-se os esquemas lógicos XML da Figura 5.1 e a suposição

que existam, em média, 4 instâncias de *EB* associadas a cada instância de *EA*. Para recuperar as instâncias de elementos *EB* associadas a uma das instâncias de *EA* bastam 4 *junções por estrutura* através do esquema definido na Figura 5.1 (a). Isto ocorre por que *EB* é um sub-elemento de *EA* neste esquema. No esquema definido na Figura 5.1 (b), o relacionamento entre *EA* e *EB* é estabelecido por referência. O elemento *R1* é criado para representar este relacionamento, sendo o atributo *ebRef* uma referência à um elemento do tipo *EB*. Para obter as instâncias do elemento *EB* associadas a uma instância de *EA* neste esquema, é preciso comparar o valor do atributo *ebRef* de cada uma das 4 instâncias de *R1* com os valores do atributo *ebId* de todas as instâncias de *EB* presentes em um documento XML. Supondo que existam 100 instâncias do elemento *EB* no documento XML, 400 *junções por valor*, ou 400 comparações devem ser executadas por esta consulta. Claramente pode-se verificar, neste caso, que o número de comparações aumenta à medida que aumenta o número de instâncias de *EB* no documento XML. Em suma, o número de comparações tende a ser crítico para a execução de consultas frequentemente aplicadas sobre relacionamentos de referência e que envolvem elementos com um número elevado de instâncias.

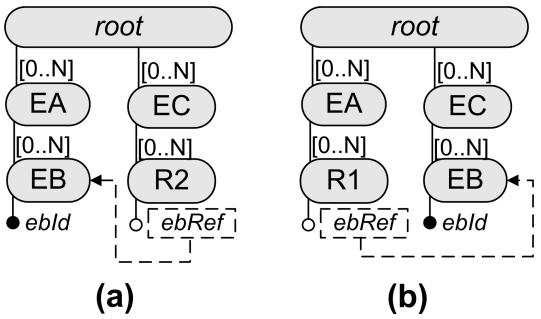

Figura 5.1: (a) Relacionamento hierárquico entre os elementos *EA* e *EB*, (b) Relacionamento de referência entre os elementos *EA* e *EB*

EB pode ser representado como um sub-elemento de *EA* (Figura 5.1(a)) ou como um sub-elemento de *EC* (Figura 5.1(b)). Na Figura 5.1(a), o relacionamento entre *EC* e *EB* é estabelecido por referência, visto que *EB* se transformou em um sub-elemento de *EA*. Supondo que consultas que envolvem o relacionamento entre *EC* e *EB* tenham

uma frequência de execução baixa, se comparada à frequência das consultas envolvendo a associação entre *EA* e *EB*, o esquema da Figura 5.1(b) geraria um maior número de comparações que o esquema da Figura 5.1(a). Este esquema gera um maior número de comparações, pois a consulta com maior frequência de execução é aplicada sobre o relacionamento de referência envolvendo *EA* e *EB*. Por conseguinte, a decisão por representar *EB* como um sub-elemento de *EA* ou de *EC* pode ser tomada com auxílio das informações referentes à frequência das consultas envolvendo estes conceitos, bem como do volume de instâncias esperadas para eles.

Este capítulo descreve uma extensão à abordagem de conversão apresentada no Capítulo 4. Esta extensão é dirigida por informações de carga do BD estimadas sobre um esquema conceitual. Considera-se como informações de carga o volume de instâncias esperadas para os conceitos do esquema e dados referentes às consultas e suas respectivas frequências de execução sobre o BD. O processo de conversão estabelecido pelo *Algoritmo 1*, definido no Capítulo 4, foi ampliado para a realização da análise destas informações. Esta extensão no processo tem por finalidade minimizar o número de comparações que podem ser geradas pela execução de consultas sobre conceitos relacionados por referências.

Embora seja difícil coletar informações sobre a carga de dados, estas informações são consideradas essenciais na tomada de decisões durante a modelagem lógica e física de um BD. De acordo com [BAT 92], 20% das operações mais frequentes sobre os dados produzem 80% da carga do BD. Assim sendo, é suficiente analisar 20% das operações mais frequentes do BD. Sabe-se que a performance do BD só pode ser definitivamente determinada após a modelagem física. Entretanto, a estrutura XML produzida pela modelagem lógica é considerada um fator determinante para a performance de aplicações baseadas em XML [WIW 06, XU 03].

Este capítulo é organizado em quatro seções. A primeira seção apresenta a modelagem de carga que é aplicada sobre um esquema conceitual para a obtenção de dados a serem utilizados pela abordagem de conversão. A seção seguinte define as modificações realizadas sobre o processo de conversão para considerar as informações de carga. Um exemplo de aplicação desta abordagem é apresentada pela seção três. Na última seção,

são apresentadas as conclusões deste capítulo.

Um artigo contendo a abordagem de conversão baseada em análise de carga de dados foi publicado nos anais do *8th ACM Symposium on Document Engineering* [SCH 08b].

5.1 Modelagem de Carga

Esta seção apresenta a modelagem dos dados referentes à carga esperada para um BD XML. O objetivo desta modelagem é produzir informações de apoio ao processo de conversão *EER-XML Lógico* na produção de esquemas que possam responder de forma mais eficiente às principais operações do BD. A metodologia de modelagem proposta por esta seção constitui uma extensão da modelagem de carga para bancos de dados convencionais proposta em [BAT 92]. A modelagem da carga de BD proposta em [BAT 92] inclui duas categorias de informação: volume de dados e carga da aplicação. O volume de dados é mensurado pelo número médio de instâncias esperadas para cada tipo entidade e relacionamento de um esquema EER. A carga da aplicação é estimada por uma descrição das principais operações da aplicação. Esta descrição envolve a frequência de execução das operações e as entidades e relacionamentos acessados por cada uma delas.

A *Definição 1* determina o conceito de esquema EER acrescido de informações referentes ao volume de dados. A *Definição 2* especifica os dados das operações que são classificadas como a carga da aplicação.

Definição 1 (Esquema EER com Volume de Dados) *Um esquema EER com informações adicionais referentes ao volume de dados de um BD é nomeado como EERV. Um EERV define um número médio de instâncias N para cada tipo entidade N(E) ou relacionamento N(R). Para cada tipo entidade E participante de um tipo relacionamento R existe uma cardinalidade média associada Avg(E,R). Avg(E,R) estabelece o número médio de instâncias de R associadas por uma instância de E.*

Um exemplo de esquema EERV é apresentado na Figura 5.2. O número médio de instâncias para tipos entidade e relacionamento é indicado no interior de suas respectivas notações gráficas. Por exemplo, o número médio de instâncias estimado para a entidade

A é 100. A cardinalidade média dos relacionamentos é representada sobre a participação mínima e máxima de cada entidade envolvida. Para o relacionamento *B*, a cardinalidade média para a entidade participante *A* é 5. Observa-se que o valor obtido pela multiplicação do número médio de instâncias de uma entidade com a sua respectiva cardinalidade média em um determinado relacionamento é sempre igual ao valor obtido pela multiplicação destes conceitos para uma outra entidade participante do mesmo relacionamento. No relacionamento *B* da Figura 5.2, o valor desta multiplicação para a entidade A (100*5) é igual ao valor obtido para *C* (500*1), que é de 500 instâncias. Outra questão a ser observada é que o valor desta multiplicação é assumido como o número médio de instâncias para o tipo relacionamento. Neste caso, o número médio de instâncias para o relacionamento *B* é 500.

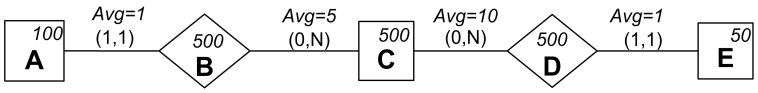

Figura 5.2: Exemplo de um esquema EER acrescido com informações do volume de dados do BD

Definição 2 (Operação) *Uma operação O é uma interação básica com um BD que inclui operações de consulta e atualização de dados. Para cada operação existe uma frequência média f(O) de execução e uma lista de tipos entidade e relacionamento* $\{t_1, t_2, ..., t_n\}$ *acessados por O. A ordem dos tipos entidade e relacionamento nesta lista especifica a sequência na qual os tipos são acessados por O. Cada tipo* t_i $(1 \leq i \leq n)$ *possui um número médio de instâncias* $f_o(t_i)$ *que são acessadas por O.*

A Tabela 5.1 apresenta algumas operações para o esquema da Figura 5.2. Por exemplo, *O1* possui uma frequência média de 100 vezes ao dia ($f(O_1)$=100). Os tipos entidade e relacionamento *A*, *B* e *C* são acessados por *O1* nesta sequência. O conceito inicial *A* possui $f_{o1}(A)$=100. Na sequência de navegação pelos tipos acessados por *O1*, o número médio de instâncias acessadas de *B* e *C* é obtido pela multiplicação de $f_{o1}(A)$ pelo $Avg(A,B)$ que é 5. Desta forma, o valor de $f_{o1}(B)$ e $f_{o1}(C)$ é 500.

Tabela 5.1: Operações do esquema *EERV* da Figura 5.2

O	*f(O)* (por dia)	t_i	$fo(t_i)$
O1	100	A	100
		B	500
		C	500
O2	70	E	70
O3	50	E	50
		D	5000
		C	5000

Com a finalidade de produzir informações que possam ser utilizadas pelo algoritmo de conversão *EER-XML Lógico*, mais três conceitos são introduzidos: *Frequência de Acesso Geral de um Tipo Entidade ou Relacionamento*, *Frequência de Acesso Total do Esquema* e *Frequência de Acesso Mínima*. A *Frequência de Acesso Geral* (FAG) denota o volume total de instâncias de um tipo entidade ou relacionamento que são acessadas por todas as operações das quais este tipo participa. A *Definição 3* estabelece este novo conceito.

Definição 3 (Frequência de Acesso Geral de Tipo Entidade ou Relacionamento - FAG) *Dado um tipo entidade ou relacionamento t de um esquema EERV que é acessado por operações do BD, existe uma frequência de acesso geral f(t) sendo* $f(t) = \sum_{i=1}^{n} f_{oi}(t)$ *e n o número total de operações em que t participa.*

De acordo com os dados fornecidos pela Tabela 5.1, $f(A)$=100, $f(B)$=500, $f(C)$=5500, $f(D)$=5000 e $f(E)$=120.

O processo de converão necessita obter um valor que é considerado por um usuário especialista como uma *Frequência de Acesso Mínima*, isto é, valores que abaixo desta frequência são considerados uma frequência de acesso insignificante para a aplicação do BD. Assume-se que um usuário especialista provê este valor em forma de percentagem no início do processo de conversão. Um valor correspondendo à *Frequência de Acesso Total* de todas as operações consideradas é estabelecido pela *Definição 4* para a obtenção da *Frequência de Acesso Mínima* na *Definição 5*. De acordo com [BAT 92], 20% das operações mais frequentes produzem 80% da carga do BD. Com base nesta afirmação,

o cálculo da *Frequência de Acesso Mínima* assume que a *Frequência de Acesso Total* estimada corresponde a 80% da carga do BD.

Definição 4 (Frequência de Acesso Total - FAT) *Dado um esquema EERV e um conjunto de operações, a frequência de acesso total é dada por* $f_{TOT}(EERV) = \sum_{i=1}^{n} f(t_i)$, *onde n é o número de tipos entidade e relacionamento de um esquema EERV.*

Definição 5 (Frequência de Acesso Mínima - FAM) *Dado um esquema EERV, uma* f_{TOT} *(EERV) correspondendo a 80% da carga da aplicação e um valor percentual* p_{min}*(EERV) representando o acesso mínimo da aplicação, existe uma frequência de acesso mínima* f_{min}*(EERV) que corresponde ao* p_{min}*(EERV) aplicado sobre* f_{TOT}*(EERV) definida por* f_{min} *(EERV) = (*f_{TOT} *(EERV) * *p_{min}*(EERV))/80.*

Considerando as operações da Tabela 5.1, obtém-se $f_{TOT}(EERV)$ = 11220. Conforme especificado pela Definição 4, este valor é obtido pela soma da *FAG* de todos os tipos envolvidos nas operações consideradas. Dado $p_{min}(EEV)$ = 1% como um valor percentual denotando o valor mínimo de acesso relacionado à frequência de acesso geral ($f_{TOT}(EERV)$ = 11220), obtém-se f_{min}=140,25 pela aplicação da fórmula $f_{min}(EERV) = (f_{TOT}(EERV) * p_{min}(EERV))/80$.

A próxima seção apresenta as modificações efetuadas no processo de conversão para considerar as informações levantadas na modelagem de carga.

5.2 Processo de Conversão EER-XML Lógico baseado em Informações de Carga

Esta seção apresenta as alterações realizadas nos procedimentos do *Algoritmo 1* da Seção 4.3 para considerar as informações de carga de um BD. O algoritmo modificado é denominado ***Algoritmo 2***. Considera-se como entrada deste novo algoritmo um esquema EERV, um valor correspondente a *FAM* e um conjunto de valores que estabelecem a *FAG* para cada tipo entidade e relacionamento do esquema *EERV*.

O Procedimento 1 foi modificado para considerar as informações de carga na conversão de tipos generalização e união. O Procedimento 4, mostrado a seguir, contém estas modificações. As alterações estão presentes nas linhas 2, 6, 10, 18 e 22. Na linha 2, os tipos generalização e união continuam sendo ordenados para que o processo execute a conversão no sentido *bottom-up*, em casos de hierarquias de múltiplos níveis. Entretanto, quando há um caso de herança múltipla, os tipos generalização que aparecem por primeiro na lista são aqueles que apresentam um valor maior para a *FAG* de suas superclasses. Isto significa que em um caso de herança múltipla, a superclasse que é mais frequentemente acessada pela aplicação é transformada no elemento pai do elemento que representa uma subclasse que constitui uma especialização de mais de um superclasse. Neste caso, as especializações restantes devem ser representadas por relacionamentos por referência, conforme definido na cláusula ELSE da *Regra GMH* (Seção 4.2.2).

As duas outras modificações realizadas no Procedimento 4 estão relacionadas à aplicação das regras *GMS* e *GMSB*. Na aplicação da Regra *GMS*, ao invés de verificar se não existem tipos relacionamentos associados às subclasses, o novo procedimento verifica se a *FAG* das subclasses é menor que a *FAM*. No caso em que a *FAG* é superior à *FAM* para uma subclasse, o procedimento assume que esta subclasse participa em operações frequentes e que, consequentemente, a distinção entre superclasse e subclasses é relevante e deve ser preservada. O mesmo ocorre na aplicação da Regra *GMSB*. Neste caso, verifica-se se a *FAG* da superclasse é inferior à *FAM*. Caso a superclasse apresentar uma *FAG* superior à *FAM*, o procedimento assume que esta deve ser representada como um elemento atômico e que, portanto, a aplicação da regra *GMSB* não é conveniente.

O Procedimento 3 foi também reformulado para considerar informações de carga na conversão de tipos relacionamento. O Procedimento 5 apresenta este procedimento reformulado. Ao invés de percorrer os fechamentos funcionais completos das entidades do esquema EER, este novo procedimento processa os tipos relacionamento segundo uma ordenação que garante que os relacionamentos com maior *FAG* são convertidos primeiro (linha 2). Esta ordenação é estabelecida para dar prioridade para a conversão de tipos relacionamento que representam o maior impacto para a carga da aplicação. Esta ordenação permite que, por exemplo, se há mais de uma possibilidade de aninhamento para

Procedimento 4: *converterGeneralizacoesUnioes*

Entrada: Um esquema conceitual *EERV- EC* $f_{min}(EERV)$

Saída: Fragmentos gerados no Modelo Lógico XML - *XGU*

1 $GU \leftarrow$ o conjunto $\{gu_1, ..., gu_n\}$ de tipos generalização e união de *EC*;
2 Ordene *GU* de forma que os primeiros tipos da lista sejam os tipos generalização e união presentes nos níveis mais inferiores de uma hierarquia de múltiplos níveis. **Em casos de herança múltipla, os tipos cujas as superclasses possuem uma *FAG* mais alta aparecem primeiro**;
3 $GU' \leftarrow$ a lista ordenada de *GU*;
4 **para cada** $gu_i \in GU'$ **faça**
5 **se** gu_i *é um tipo generalização* **então**
6 **se** *(Não há subclasse marcada em* gu_i*) E (Não há subclasses com mais de uma superclasse) E (**Não há subclasses em* gu_i *com* $f(s) > f_{min}(EERV)$*)* **então**
7 Aplique a *Regra GMS*;
8 Marque todas as subclasses de gu_i;
9 **senão**
10 **se** gu_i *é total e disjunta) E (Não há subclasse marcada em* gu_i*) E (*$f($**superclasse**$) < f_{min}(EERV)$*)* **então**
11 Aplique a *Regra GMSB*;
12 **senão**
13 Aplique a *Regra GMH*;
14 Marque todas as subclasses de gu_i;
15 **fim**
16 **fim**
17 **senão**
18 **se** gu_i *é total) E (Não há superclasse marcada em* gu_i*) E (**Não há superclasse s em* gu_i *com* $f(s) > f_{min}(EERV)$*) E (Não há superclasses com mais de uma superclasse)* **então**
19 Aplique a *Regra UMS*;
20 Marque todas as superclasses de gu_i;
21 **senão**
22 **se** *(Não há superclasse marcada em* gu_i*) E (*$f($**subclasse**$) < f_{min}(EERV)$*)* **então**
23 Aplique a *Regra UMSP*;
24 **senão**
25 Aplique a *Regra UMH*;
26 Marque todas as superclasses de gu_i;
27 **fim**
28 **fim**
29 **fim**
30 **fim**

um elemento que representa uma entidade *A*, o procedimento faz com que *A* se torne um sub-elemento de um elemento que representa uma entidade associada à *A* através do relacionamento com a maior *FAG*.

Para a conversão de cada tipo relacionamento, o procedimento determina qual das

Procedimento 5: *converterRelacionamentos*

Entrada: Um esquema conceitual *EERV - EC*;
Fragmentos gerados pelo procedimento *converterGeneralizaçõesUniões - XGU*;
f_{min}(*EERV*)
Saída: Fragmentos gerados no Modelo Lógico XML - *XR*

1 $R \leftarrow$ o conjunto $\{r_1, ..., r_n\}$ de tipos relacionamento de *EC*;
2 *R* é ordenado de forma que os primeiros tipos da lista são os tipos relacionamento que possuem as maiores *FAG*. Garanta que tipos relacionamento envolvendo entidades associativas estejam dispostos em *R* após os tipos relacionamento internos das entidades associativas;
3 **repita**
4 $r_i \leftarrow$ O primeiro r_i não marcado de *R*;
5 Seja $\{E_1, ..., E_n\}$ o conjunto das entidades relacionadas por r_i;
6 **se** r_i *é binário E há uma entidade* E_i *com participação (1,1) em* r_i **então**
7 E_2 é E_i e E_1 é a outra entidade de r_i;
8 **senão**
9 E_1 é a entidade que possui a maior *FAG* em r_i;
10 **fim**
11 **se** *(a máxima cardinalidade de* r_i *é 1:1) E (A participação de* E_2 *em* r_i *é (1,1)) E (*E_2 *está desmarcada)* **então**
12 Aplique a *Regra RMEU*;
13 Marque E_2;
14 **senão**
15 **se** *(*r_i *é binário) E (A participação de* E_2 *em* r_i *é (1,1)) E (*E_2 *está desmarcada) E (*E_1 *!=* E_2*)* **então**
16 Aplique a *Regra RMH*;
17 Marque E_2;
18 **senão**
19 Aplique a *Regra RMR*;
20 **fim**
21 **fim**
22 Marque r_i;
23 **até** *que todo* $r_i \in R$ *esteja marcado* ;

entidades do relacionamento gera o elemento que deve ser colocado no topo da hierarquia XML gerada pela conversão de um relacionamento. Em se tratando de um relacionamento binário *R* no qual uma das entidades apresenta participação (1,1), a entidade escolhida deve ser a outra entidade envolvida em *R* (linha 7). Para os demais casos, a entidade que possuir a maior *FAG* em *R* é definida como a entidade que gera o elemento no topo desta hierarquia que está sendo gerada (linha 9). Para este último caso, assume-se que *R* é mais freqüentemente acessado através da entidade que apresenta a maior *FAG* em *R*. Um exemplo da aplicação destes novos procedimentos é apresentado na próxima seção.

5.3 Exemplo de Aplicação do Processo de Conversão Baseado em Informações de Carga

A Figura 5.3 apresenta um esquema EERV. Assume-se que o volume de dados e as operações do banco de dados que são representadas por este esquema foram providas por um usuário especialista. As operações são apresentadas pela Tabela 5.2 e a *FAG* de cada conceito do esquema EER é dada pela Tabela 5.3. A *FAT* (soma das *FAG* de todos os conceitos do esquema) é 3825. Supõe-se que a *FAM* é dada sobre 0.1% da *FAT*, que é 4,78.

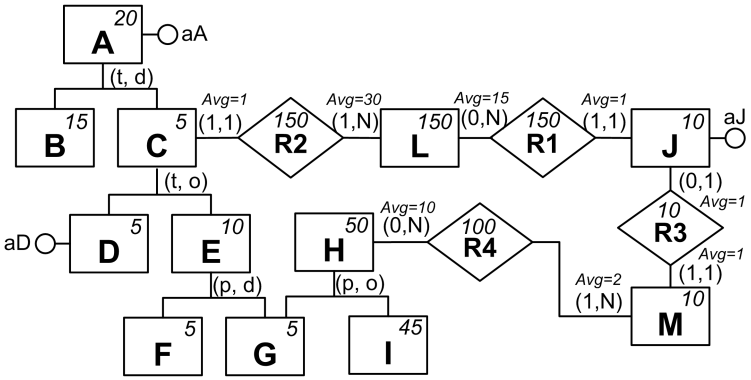

Figura 5.3: Um esquema *EERV* exemplo

Nota-se que a *FAG* de alguns dos conceitos é dada pelo valor zero, pois os respectivos tipos entidade ou relacionamento não são acessadas pelas operações consideradas. Em outras palavras, estes conceitos não estão envolvidos nas principais operações do BD. No entanto, visto que são representados no esquema conceitual, não podem ser desconsiderados pelo processo de conversão. A conversão *EERV-XML Lógico* definida pelo *Algoritmo 2* é exemplificada a seguir, considerando as informações de carga fornecidas. O esquema lógico XML gerado é mostrado na Figura 5.4.

Iniciando pela conversão de tipos generalização, o procedimento continua a converter a hierarquia de múltiplos níveis no sentido *bottom-up*. Porém, para a herança múltipla envolvendo as hierarquias iniciadas por *E* e *H*, o procedimento executa a conversão a partir da hierarquia cuja superclasse apresentar o valor mais alto para a medida de *FAG*.

Tabela 5.2: Operações do esquema *EERV* da Figura 5.3

O	*f(O)* (por dia)	t_i	$fo(t_i)$
O1	100	J	100
		R3	100
		M	100
		R4	1000
		H	1000
O2	50	J	50
		R3	50
		M	50
O3	20	C	20
		R2	600
		L	600
O4	5	J	5
		R1	75
		L	75

Tabela 5.3: Frequência de acesso geral dos tipos do esquema *EERV* da Figura 5.3

t	f_t	*t*	f_t
A	0	I	0
B	0	J	155
C	20	L	675
D	0	M	150
E	0	R1	75
F	0	R2	600
G	0	R3	150
H	1000	R4	1000

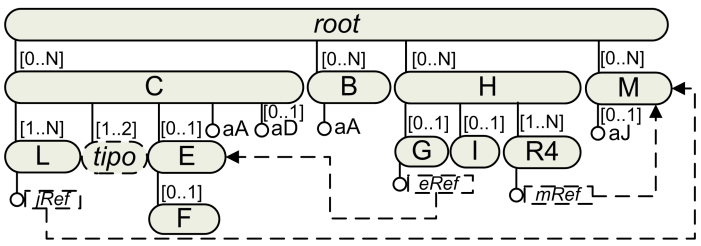

Figura 5.4: O esquema lógico XML gerado pelo *Algoritmo 2*

Neste caso, a conversão inicia pela hierarquia da superclasse *H* visto que sua *FAG* (1000) é maior do que a *FAG* de *E* (0). A subclasse *G* é transformada em um sub-elemento do

elemento que representa a superclasse *H*. Conseqüentemente, a especialização de *G* como uma subclasse de *E* é representada por um atributo de referência no elemento criado para representar *G*.

A definição do *Algoritmo 1* estabelece que a ordem de conversão de hierarquias envolvidas em casos de herança múltipla é indiferente. Neste caso, se *G* fosse transformada em um sub-elemento de *E*, a operação *O1* deveria executar junções por valor caso desejasse obter valores de *G* associados pelo relacionamento de referência que deveria ser estabelecido entre *G* e *H*. Esta situação é evitada pelo esquema gerado pelo *Algoritmo 2* ao processar *G* como um sub-elemento de *H*. Desta forma, um número menor de comparações é exigida para alcançar *G*, sendo ele um sub-elemento de *H*.

A execução do *Algoritmo 2* prossegue na conversão dos demais tipos generalização do esquema. A generalização estabelecida pela superclasse *C* é convertida pela *Regra GMS*, visto que as subclasses *D* e *E* possuem uma *FAG* (0) inferior à *FAM* (4,78). Conforme pode-se constatar pelas operações consideradas, os conceitos *D* e *E* não participam das operações mais relevantes da aplicação e portanto podem ser indiretamente representados por atributos no modelo de conteúdo do elemento que representa a superclasse *C*. Na conversão do tipo generalização estabelecido pela superclasse *A*, a *Regra GMS* não pode ser aplicada, visto que a subclasse *C* apresenta uma *FAG* (20) superior a *FAM* (4,78). Neste caso, a *Regra GMSB* é aplicada, visto que nenhuma restrição é imposta a ela pelas informações de carga consideradas.

O *Algoritmo 2* estabelece uma ordem de conversão para tipos relacionamento. Incialmente são convertidos os relacionamentos que apresentam os mais altos valores para a *FAG*. Considerando as frequências de acesso geral dos conceitos apresentados pela Tabela 5.3, os tipos relacionamento *R4*, *R2*, *R3* e *R1* são convertidos nesta sequência. Na conversão de *R4*, a *Regra RMR* é aplicada e um relacionamento de referência é gerado a partir de *H*, visto que trata-se de um relacionamento N:N. A entidade *H* é escolhida para estabelecer o relacionamento (linha 9 do Procedimento 5), visto que a *FAG* de *H* (1000) é superior à *FAG* de *M* (150).

Nos passos subsequentes, os relacionamentos *R2*, *R3* e *R1* são convertidos pelas regras *RMH*, *RMEU* e *RMR*, respectivamente. A prioridade dada pelo algoritmo na con-

versão de *R2* antes de *R1* faz com que *L* se torne em um sub-elemento de *C* e, conseqüentemente, o relacionamento *R1* seja estabelecido por referência. Esta escolha é tomada visto que a operação *O3* que envolve *R2* gera um maior número de acessos do que *O4* que envolve *R1*. Conseqüentemente, a representação de *R2* através de referências resultaria em um número mais elevado de comparações na execução de *O3* do que na execução de *O4*. Por isso, *R1* é representado por referência. Em outras palavras, prioriza-se a representação hierárquica no esquema XML de relacionamentos conceituais que podem gerar uma maior frequência de acesso de acordo com a descrição das operações.

5.4 Considerações Finais

Este capítulo apresentou as modificações realizadas sobre o processo de conversão *EER-XML Lógico* para considerar informações da carga estimada para o BD que está sendo projetado. A carga do BD é estimada sobre um esquema EER e compreende informações relacionadas ao volume de dados esperado para os conceitos modelados, bem como as principais operações do banco. Uma metodologia para a modelagem destas informações é aplicada para gerar informações que são utilizadas pelo processo de conversão *EERV-XML Lógico*. Algumas modificações nos procedimentos apresentados no capítulo anterior são estabelecidas para a consideração de informações da carga do BD. O objetivo principal destas modificações é permitir que os relacionamentos entre conceitos que serão frequentemente acessados pelas operações do BD sejam representados por relacionamentos hierárquicos no esquema XML sempre que possível. Além disto, esta estratégia faz com que o número de comparações gerado por operações envolvendo referências seja minimizado em termos do número de acesso a elementos em um documento XML.

No próximo capítulo, um estudo de caso é apresentado para validar o processo de conversão proposto. Os experimentos mostram o ganho no desempenho de consultas executadas sobre esquemas XML produzidos por esta abordagem.

Capítulo 6

Estudo de Caso

Uma vez descritas as metodologias para conversão de esquemas EER em esquemas lógicos XML, é apresentado um estudo de caso para avaliar os esquemas lógicos XML produzidos pelas metodologias de conversão propostas por este trabalho. Um esquema EER representando o domínio de compras e vendas de empresas foi utilizado como base para a produção dos esquemas XML. As etapas de transformação do esquema EER em esquemas lógicos XML são descritas detalhadamente por este capítulo, considerando as duas metodologias propostas: (i) conversão EER-XML convencional, e (ii) conversão EER-XML baseada em análise da carga do BD.

O esquema EER utilizado para este estudo de caso é apresentado na Figura 6.1. Este esquema foi produzido com a finalidade de representar os principais processos de negócio relacionados ao setor de compras e vendas, que são as transações relacionadas a pedidos e notas fiscais. Alguns atributos foram omitidos para fins de simplificação do esquema EER. Em consideração às restrições impostas pelo domínio modelado, o esquema conceitual da Figura 6.1 é considerado completo, correto e livre de redundâncias.

Este capítulo está organizado em três seções. Na primeira seção é discutida a conversão do esquema EER em um esquema lógico XML através do processo de conversão proposto no capítulo 4 deste trabalho. Na seção seguinte, a metodologia de conversão baseada em análise de carga do BD é aplicada. Para a aplicação desta metodologia, uma previsão para o volume de dados e para as operações de um BD é considerada para o

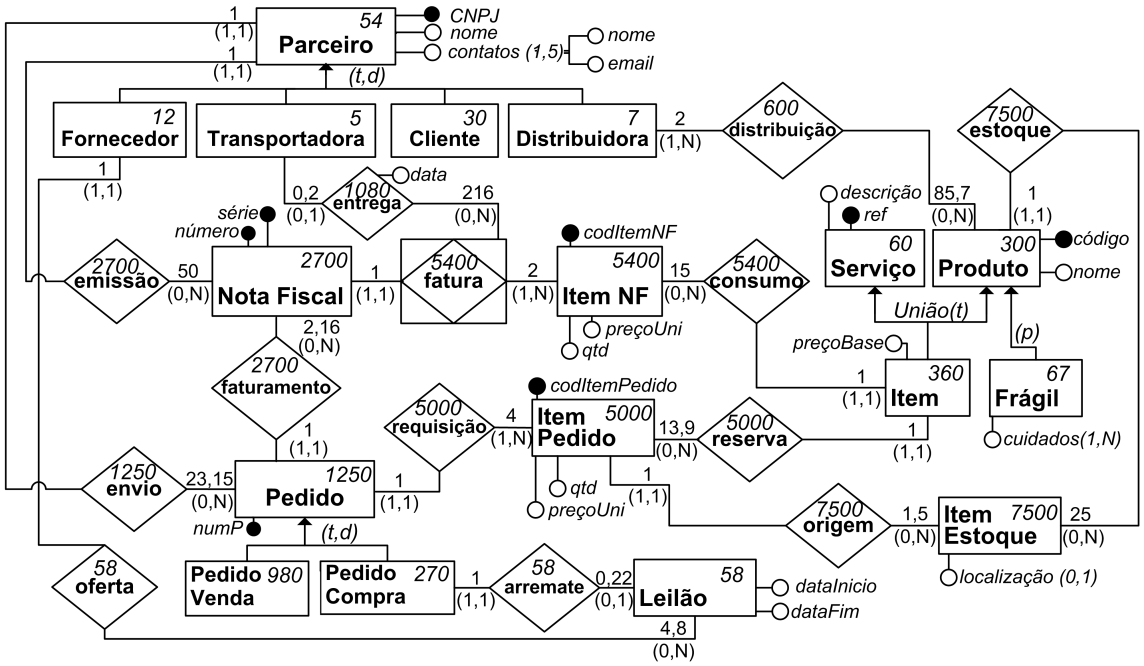

Figura 6.1: Esquema EERV

esquema EER fornecido. Na terceira seção, os esquemas lógicos XML produzidos por ambas as abordagens são avaliados com relação à carga ocasionada pela execução das operações previstas sobre documentos XML adequados a estes esquemas.

6.1 Conversão EER-XML Convencional

O algoritmo de conversão apresentado no capítulo 4 (*Algoritmo 1*) é aplicado para o esquema da Figura 6.1. Inicialmente, constata-se que não existem hierarquias de múltiplos níveis envolvendo as generalizações e o tipo união presente no esquema EER considerado. Consequentemente, a tarefa de ordenação executada pelo procedimento 1 não afeta a ordem de disposição dos tipos generalização e união. Optou-se por iniciar a conversão pelo tipo generalização envolvendo a superclasse *Pedido* e as subclasses *Pedido Venda* e *Pedido Compra*. O tipo relacionamento *arremate* envolvendo a subclasse *Pedido Compra* impede que esta generalização seja convertida pela *Regra GMS*. Pelo mesmo motivo, a *Regra GMSB* não é adequada, dada a existência dos relacionamentos *envio*,

faturamento e *requisição* estabelecidos com a superclasse *Pedido*. Neste caso, a única opção é a aplicação da *Regra GMH*, que gera elementos complexos para representar cada entidade envolvida e associa tais elementos através de relacionamentos hierárquicos. A Figura 6.2 apresenta o esquema lógico XML gerado pela conversão. Considerando que o modelo de conteúdo do elemento *Pedido* será composto por outros sub-elementos além de *Pedido Venda* e *Pedido Compra*, seria incorreto definir os elementos das subclasses como sub-elementos diretos de *Pedido*, visto que o construtor de ordem de *Pedido* deveria ser definido como *exclusivo* para a representação da restrição de disjunção estabelecida por esta generalização. Por esta razão, um elemento agrupador *Tipo* é criado e definido como sub-elemento direto de *Pedido* e como elemento pai dos elementos *Pedido Venda* e *Pedido Compra*. Para representar as restrições de totalidade e disjunção desta generalização, a ocorrência de *Tipo* em *Pedido* é definida como [1..1] e seu construtor de ordem como *exclusivo*.

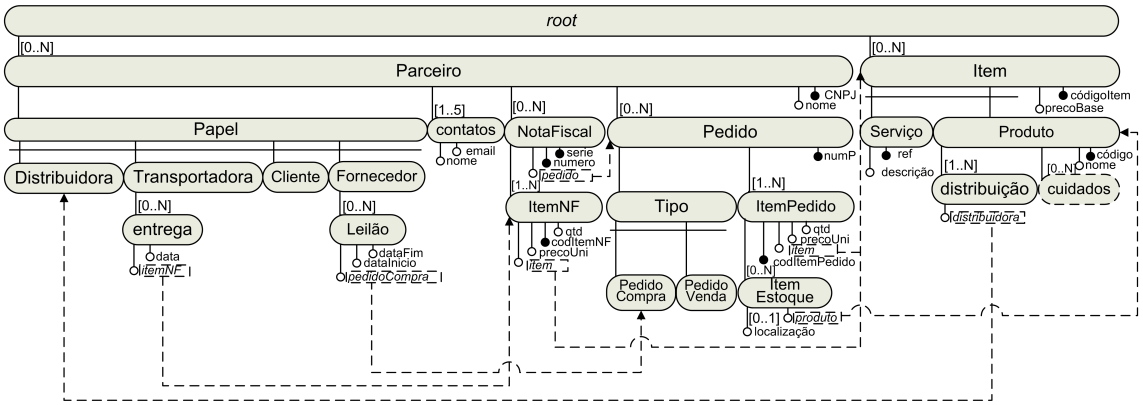

Figura 6.2: Esquema Lógico XML obtido pelo processo convencional

Uma situação semelhante à generalização de *Pedido* ocorre para a generalização envolvendo a superclasse *Parceiro*. A existência de tipos relacionamento envolvendo separadamente a superclasse *Parceiro* e as subclasses *Fornecedor* e *Transportadora* torna a *Regra GMH* a única opção de conversão. Um elemento agrupador foi criado para representar a disjunção dos elementos que representam as subclasses de *Parceiro*. Considera-se que após a execução do *Algoritmo 1*, o nome *Papel* é indicado para o elemento agrupador de *Parceiro* por um usuário especialista no domínio (ajustes de usuário). Na geração do

elemento complexo para representar a entidade *Parceiro*, o atributo composto e multivalorado *contatos* é transformado em um sub-elemento complexo de *Parceiro* contendo os atributos *nome* e *email* e apresentando a ocorrência [1..5].

O tipo união total envolvendo as superclasses *Serviço* e *Produto* e a subclasse *Item* é também convertido pela *Regra GMH*, dada a existência de tipos relacionamento envolvendo separadamente a supeclasse *Produto* (*estoque* e *distribuição*) e a subclasse *Item* (*consumo* e *reserva*). A *Regra GMS* é aplicada na conversão da generalização envolvendo *Produto* e a subclasse *Frágil*, fazendo com que o atributo multivalorado *cuidados* seja representado no modelo de conteúdo do elemento *Produto* como um sub-elemento simples. Embora a ocorrência do atributo *Frágil* tenha sido definida como [1..N] no esquema EER, a ocorrência do elemento simples *cuidados* em *Produto* é definida como [0..N], visto que a *Regra GMS* estabelece que o conteúdo das subclasses deve ser tratado como opcional no elemento criado para representar a superclasse. Neste caso, especificamente, a ocorrência opcional de *cuidados* é válida dada a restrição de parcialidade do tipo generalização envolvendo *Produto* e *Frágil*.

A conversão dos tipos relacionamento é iniciada pela ordenação das entidades do esquema EER. Conforme estabelecido pelo procedimento 2, as entidades devem ser ordenadas de forma que apareçam por primeiro aquelas que participam no maior número de *fechamentos funcionais completos*, sendo que este número deve ser maior que 1. A Tabela 6.1 apresenta os *fechamentos funcionais completos* computados para cada tipo entidade do esquema conceitual. Segundo os dados desta tabela, a lista obtida pela ordenação proposta é: *LE = {Parceiro, Pedido, Item, Fornecedor, NotaFiscal, ItemPedido, PedidoCompra, Produto}*. Na sequência, as entidades que foram descartadas (entidades que participam em apenas um *fechamento funcional completo*) são adicionadas à lista *LE*, de forma que as entidades associadas ao maior número de tipos relacionamento apareçam primeiro. Desta forma, a nova lista de entidades ordenadas é: *LE = {Parceiro, Pedido, Item, Fornecedor, NotaFiscal, ItemPedido, PedidoCompra, Produto,* ***ItemNF, Leilão, ItemEstoque, Transportadora, Distribuidora, Cliente, Serviço, PedidoVenda, Frágil****}*.

Obtendo-se a primeira entidade de *LE*, marca-se a entidade *Parceiro* como uma *entidade de continuação*. Uma vez marcada como *entidade de continuação*, são convertidos

Tabela 6.1: Fechamentos funcionais completos das entidades do esquema EER da Figura 6.1

Fechamentos Funcionais Completos	
Parceiro = {Parceiro}	Pedido = {Pedido, Parceiro}
Fornecedor = {Fornecedor}	ItemPedido = {ItemPedido, Pedido, Parceiro, Item}
Cliente = {Cliente}	Leilão = {Leilão, PedidoCompra, Fornecedor}
Transportadora = {Transportadora}	PedidoVenda = {PedidoVenda}
Distribuidora = {Distribuidora}	PedidoCompra = {PedidoCompra}
NotaFiscal = {NotaFiscal, Parceiro, Pedido}	Frágil = {Frágil}
Produto = {Produto}	Item = {Item}
Serviço = {Serviço}	ItemEstoque = {ItemEstoque, Produto, ItemPedido, Pedido, Item, Parceiro}
ItemNF = {ItemNF, NotaFiscal, Parceiro, Pedido, Item}	

todos os relacionamentos em que a entidade *Parceiro* está envolvida. O relacionamento *emissão* é então processado e a *Regra RMH* é aplicada, transformando *NotaFiscal* em um sub-elemento de *Parceiro*, visto que se trata de um relacionamento 1:N e a entidade do lado 1 (*NotaFiscal*) participa obrigatoriamente do relacionamento. Após a aplicação da regra, a entidade *NotaFiscal* é também marcada como uma *entidade de continuação*. Semelhantemente, a entidade *Pedido* é transformada em um sub-elemento de *Parceiro* pela aplicação da *Regra RMH*, sendo *Pedido* também marcada como uma *entidade de continuação*. Como a entidade *Parceiro* não possui mais nenhum relacionamento não processado, a conversão prossegue a partir da próxima entidade marcada como de *continuação* (*NotaFiscal*).

Na conversão do relacionamento *fatura*, a entidade *ItemNF* é transformada em um sub-elemento de *NotaFiscal*, fazendo com que *ItemNF* seja marcada como *entidade de continuação*. O outro relacionamento não processado envolvendo *NotaFiscal* (*faturamento*) é transformado pela *Regra RMR* e um relacionamento de referência é estabelecido entre *NotaFiscal* e *Pedido*, visto que *NotaFiscal* já havia sido processada como um elemento filho de um outro elemento (*Parceiro*). Uma vez que todos os relacionamentos envolvendo *NotaFiscal* foram processados, a conversão prossegue através da próxima entidade de continuação (*Pedido*). Nesta sequência, a entidade *ItemPedido* é marcada como

entidade de continuação, pois é transformada em um sub-elemento de *Pedido* através da aplicação da *Regra RMH*.

A próxima *entidade de continuação* que ainda possui relacionamentos não processados é *ItemNF*. Na conversão do relacionamento *consumo*, o relacionamento entre *ItemNF* e *Item* é estabelecido através de um relacionamento de referência a partir do elemento *ItemNF*. Embora tratando-se de um relacionamento 1:N com uma das entidades apresentando ocorrência (1,1), a conversão pela *Regra RMH* não é permitida, visto que *ItemNF* já foi processada como um sub-elemento de *NotaFiscal*. Na sequência, o mesmo ocorre para o relacionamento *reserva* envolvendo a *entidade de continuação ItemPedido* com a entidade *Item*. O relacionamento *origem* é processado na sequência, transformando a entidade *ItemEstoque* em um sub-elemento de *ItemPedido*. Como se trata de uma aplicação da *Regra RMH*, a entidade *ItemEstoque* é marcada como *entidade de continuação*, o que faz com que o relacionamento *estoque* seja convertido pelo próximo passo. A conversão de *estoque* produz um relacionamento de referência entre *ItemEstoque* e *Produto*, visto que *ItemEstoque* não pode se transformar em um sub-elemento de *Produto* por ter sido convertido como um sub-elemento de *ItemPedido* no passo anterior.

Neste ponto do processamento, não restam relacionamentos desmarcados envolvendo alguma das entidades marcadas como de continuação. Por este motivo, uma entidade que possui relacionamentos desmarcados é obtida do início da lista de entidades ordenadas (*LE*). A primeira das entidades da lista que apresenta relacionamentos desmarcados é a entidade *Fornecedor*. A conversão do relacionamento *oferta* transforma a entidade *Leilão* em um sub-elemento de *Fornecedor* e o processo continua a partir da entidade *Leilão*, dado o fato de esta ter sido marcada como *entidade de continuação*. Embora uma possibilidade de mapeamento do relacionamento 1:1 *arremate* seja a representação do relacionamento e da entidade *Leilão* pelo modelo de conteúdo do elemento *PedidoCompra* (*Regra RMEU*), o fato de *Leilão* ter sido processada como um sub-elemento de *Fornecedor* impede tal solução. A *Regra RMR* é então aplicada, transformando *arremate* em um relacionamento de referência estabelecido a partir do elemento *Leilão* com o elemento *PedidoCompra*.

Visto que todos os relacionamentos das entidades de continuação foram processa-

dos, a próxima entidade obtida da lista de entidades ordenadas é *Produto*. A partir de *Produto*, o relacionamento *distribuição* é convertido pela *Regra RMR*. Neste caso, como trata-se de um relacionamento N:N, um elemento *distribuição* é criado para representar o relacionamento. O elemento *distribuição* é colocado como um sub-elemento de *Produto*, e um relacionamento de referência é estabelecido entre *distribuição* e o elemento *Distribuidora*. Na sequência, o único relacionamento restante é também convertido pela *Regra RMR* a partir da entidade *Transportadora*. Neste caso, considerando que *fatura* está sendo representada no modelo de conteúdo de *ItemNF*, o relacionamento de referência é estabelecido a partir de *Transportadora* para o elemento *ItemNF*. A alternativa da *Regra RMR* é a única opção para a conversão deste relacionamento 1:N, visto que *fatura* já havia sido processada como um sub-elemento de *NotaFiscal* e também pelo fato de que a participação de *fatura* no relacionamento é opcional (o que também impediria o aninhamento de *fatura* em *Transportadora*). Como último passo da execução do *Algoritmo 1*, o elemento *root* é criado e os elementos *Parceiro* e *Item* são definidos como seus sub-elementos diretos.

Conforme o detalhamento do processo fornecido por esta seção, todas as informações do esquema conceitual considerado foram representadas no esquema lógico XML produzido. Além disto, o esquema lógico gerado não permite redundância de dados e ao mesmo tempo apresenta uma estrutura onde os elementos são relacionados por composições hierárquicas sempre que possível. Diante da variedade de construções do EER apresentadas pelo esquema da Figura 6.1, a cobertura provida pelas regras de conversão do *Algoritmo 1* forneceu soluções para o mapeamento de todos os construtores do EER presentes no esquema.

Os esquemas gerados pela metodologia de conversão em questão apresentam uma estrutura lógica mais compacta e hierarquizada, se comparada com os esquemas gerados por trabalhos que apresentam uma cobertura razoável para os construtores do EER, como em [PIG 05, CHO 03, LIU 06a, FON 06]. Estas abordagens correlatas determinam a estrutura de aninhamento dos esquemas XML através de um conjunto de entidades que são marcadas para serem convertidas como elementos de primeiro nível no esquema XML. Entende-se por elementos de primeiro nível, elementos que estão dispostos como sub-

elementos diretos do elemento *root*. Estas entidades de primeiro nível são identificadas pelo usuário como entidades predominantes para o domínio em [LIU 06a] e [FON 06]. Em [PIG 05] e [CHO 03], estas entidades de primeiro nível são selecionadas por uma análise de cardinalidades dos tipos relacionamento do esquema. Entretanto, em todas estas abordagens, um número considerável de entidades é classificada como entidade de primeiro nível, na maioria dos casos práticos. Este número considerável de elementos de primeiro nível faz com que um grande número de relacionamentos de referência seja estabelecido para representar os relacionamentos conceituais envolvendo estas entidades. Em outra palavras, um esquema XML pouco hierarquizado e bastante desconexo é gerado por tais metodologias.

A seção seguinte apresenta a tranformação do esquema EER através da abordagem de conversão baseada em análise da carga do BD. Nesta próxima seção, mostra-se como a sequência de aplicação das regras de conversão pode ser alterada de forma a gerar uma estrutura XML mais adequada à carga considerada para o BD. Na seção três, os esquemas produzidos pela metodologia de conversão convencional e pela metodologia baseada em análise de carga são comparados quanto ao desempenho que proporcionam às operações mais frequentes do BD.

6.2 Conversão EER-XML baseada em Análise de Carga

Um conjunto de informações referente à carga esperada para o BD que está sendo modelado por este estudo de caso foi fornecido para a aplicação do *Algoritmo 2* proposto pelo capítulo 5 deste trabalho. O esquema conceitual provido pela Figura 6.1 apresenta o volume de dados esperado para o BD em termos do número de instâncias estimadas para cada conceito do esquema e da média de cardinalidade de cada conceito participante de um relacionamento. A Tabela 6.2 apresenta 20% das operações consideradas como as mais frequentes do BD. Considera-se que o conjunto destas 6 operações são responsáveis por gerar 80% da carga do BD, segundo a regra 80-20 [BAT 92]. A frequência de acesso de cada conceito em uma operação denota a frequência de acesso diária àquele conceito pela operação considerada.

Tabela 6.2: Carga de Operações estimada para o Estudo de Caso

#	Conceitos	Frequência de Acesso
O1	Pedido	950
	faturamento	2.052
	NotaFiscal	2.052
	Total:	5.054
O2	PedidoVenda	220
	envio	220
	Cliente	220
	requisição	880
	ItemPedido	880
	reserva	880
	Item	880
	Total:	4.180
O3	ItemPedido	150
	reserva	150
	Produto	150
	distribuição	300
	Distribuidora	300
	estoque	3.750
	ItemEstoque	3.750
	Total:	8.550

#	Conceitos	Frequência de Acesso
O4	Leilão	50
	arremate	50
	PedidoCompra	50
	envio	50
	Fornecedor	50
	requisição	200
	ItemPedido	200
	Total:	650
O5	Parceiro	25
	emissão	1.250
	NotaFiscal	1.250
	fatura	2.500
	ItemNF	2.500
	Total:	7.525
O6	ItemEstoque	5
	origem	5
	ItemPedido	5
	requisição	5
	Pedido	5
	Total:	25

A *Frequência de Acesso Geral* (FAG) dos conceitos do esquema foi calculada e encontra-se listada na Tabela 6.3. Os conceitos que não estão presentes nesta tabela são conceitos que não são acessados pelas operações consideradas e, portanto, não fazem parte da carga principal do BD. No entanto, isto não significa que tais conceitos são desconsiderados pelo processo de conversão.

Tabela 6.3: Frequência de Acesso Geral (FAG) dos conceitos do esquema EER

Conceitos	*FAG*	Conceitos	*FAG*	Conceitos	*FAG*
ItemEstoque	3.755	Cliente	220	faturamento	2.052
NotaFiscal	3.302	Produto	150	emissão	1.250
ItemNF	2.500	Leilão	50	requisição	1.085
ItemPedido	1.235	PedidoCompra	50	reserva	1.030
Pedido	955	Fornecedor	50	distribuição	300
Item	880	Parceiro	25	envio	270
Distribuidora	300	estoque	3.750	arremate	50
PedidoVenda	220	fatura	2.500	origem	5

A *Frequência de Acesso Total (FAT)* do esquema é obtida pela soma da *FAG* de todos os conceitos, chegando-se ao valor de 25.984 de acessos diários. Considerou-se que a *Frequência de Acesso Mínima* (*FAM*) deveria representar 1% da carga total do BD. Este valor percentual foi aplicado sobre a *FAT* do esquema que corresponde a 80% da carga, obtendo-se o valor 324,8 para a *FAM* do esquema.

Uma vez levantados os dados da carga prevista para o BD que está sendo modelado, inicia-se a aplicação do *Algoritmo 2* através do procedimento 4. Na ausência de hierarquias de múltiplos níveis, a ordem de conversão dos tipos generalização e união presentes no esquema EER é indiferente. Iniciando-se pela generalização envolvendo a superclasse *Pedido*, verifica-se que os requisitos para a aplicação da *Regra GMS* são satisfeitos. Comparando-se com a aplicação do *Algoritmo 1*, que impediu a aplicação desta regra devido à existência de relacionamentos com as subclasses, o *Algoritmo 2* assume que a distinção entre as subclasses é irrelevante pois a *FAG* de *PedidoVenda* (220) e *PedidoCompra* (50) é inferior à *FAM* (324,8) considerada. Quando a *FAG* das subclasses é inferior à *FAM*, assume-se que as subclasses são acessadas individualmente por operações que produzem uma frequência de acesso irrelevante (mínima) sobre estas subclasses. Portanto, a decisão de representar um tipo generalização em apenas um elemento complexo mostra-se uma solução adequada.

A Figura 6.3 apresenta o esquema lógico XML gerado pela aplicação do *Algoritmo 2*. Durante a conversão do tipo generalização envolvendo a entidade *Pedido*, um elemento complexo é gerado e o atributo *TIPO* é criado como atributo deste elemento para identificar qual das subclasses está sendo representada em cada instância de *Pedido*. A ocorrência do atributo *TIPO* é definida como [1..1] para representar as restrições de totalidade e disjunção deste tipo generalização.

Semelhante à hierarquia de generalização de *Pedido*, a generalização de *Parceiro* é convertida pela *Regra GMS*. De igual forma, a *FAG* das subclasses *Fornecedor* (50), *Transportadora* (0), *Cliente* (220) e *Distribuidora* (300) é inferior à *FAM* (324,8) considerada para o esquema. Neste caso, o atributo *PAPEL* é criado como atributo discriminador das subclasses que podem ser representadas. Os tipos relacionamento envolvendo as subclasses que foram suprimidas pela aplicação desta regra devem ser posteriormente

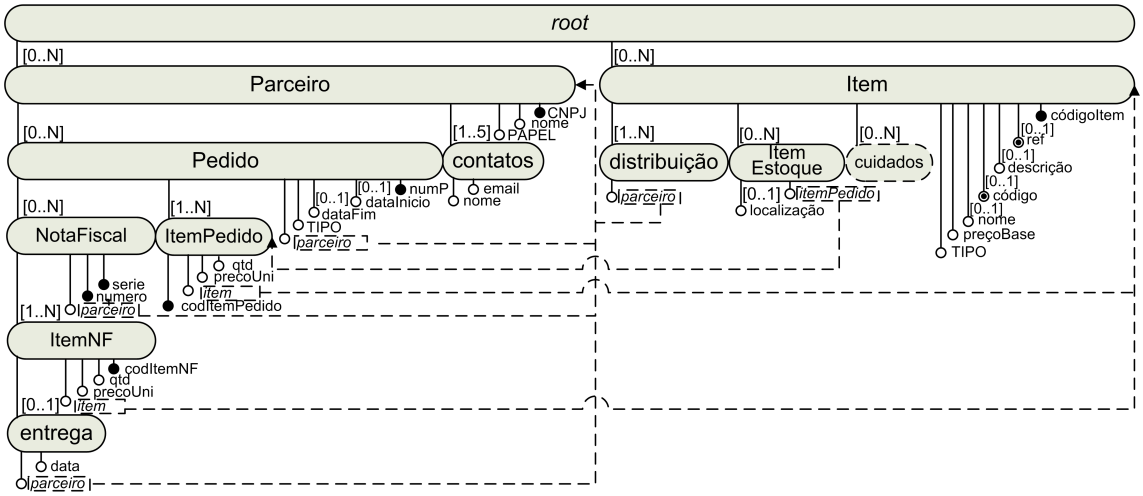

Figura 6.3: Esquema Lógico XML obtido pelo processo baseado em análise de carga

processados como relacionamentos das entidades (superclasses) que estão diretamente representadas no esquema lógico. Por exemplo, após a aplicação da *Regra GMS* sobre a generalização de *Parceiro*, o relacionamento *oferta* entre a subclasse *Fornecedor* e a entidade *Leilão* deve ser processado como sendo um relacionamento entre *Parceiro* e *Leilão*.

A generalização simples envolvendo a superclasse *Produto* e a subclasse *Frágil* é processada pela *Regra GMS*, visto que a entidade *Frágil* não participa em nenhuma das operações consideradas. No passo seguinte, o tipo união total envolvendo as superclasses *Serviço* e *Produto* e a subclasse *Item* é representada pelo elemento complexo *Item*. Nele são representados todos os atributos das superclasses e também da subclasse. Nota-se que os atributos identificadores das superclasses são tratados como atributos únicos, visto que um identificador composto pelos atributos *código* e *ref* para o elemento *Item* não faria sentido pois uma instância de *Item* deve representar ou uma instância de *Produto* ou uma instância de *Serviço* segundo as restrições impostas à tipos união. Para compor o identificador do elemento *Item*, um novo atributo foi criado (*códigoItem*) para este fim.

Em geral, observa-se uma redução de elementos gerados para representar os tipos generalização e união em virtude da constatação baseada na análise da carga do BD de que a distinção entre superclasses e subclasses é desnecessária para os tipos analisados. Por outro lado, aumenta a quantidade de atributos opcionais nos elementos do esquema.

No entanto, a ocorrência de atributos opcionais não caracteriza um problema para os BDs XML, visto que esta estrutura parcial é uma característica já assumida para dados semi-estruturados.

Iniciando a conversão de tipos relacionamento pelo procedimento 5, uma lista de relacionamentos é obtida pela ordenação destes em ordem decrescente pelos valores da *FAG* de cada tipo relacionamento. Considerando os dados da Tabela 6.3, a lista de relacionamentos é dada por *R'* = {*estoque, fatura, faturamento, emissão, requisição, reserva, distribuição, envio, arremate, origem, entrega, consumo, oferta*}. É importante destacar que os tipos relacionamento que não estão presentes na Tabela devem ser considerados nesta lista como os últimos relacionamentos a serem convertidos. A ordem estabelecida pela lista garante que os relacionamentos que geram o maior número de acessos diários são convertidos preferencialmente.

O primeiro relacionamento da lista, *estoque*, é convertido pelo procedimento através da *Regra RMH*. Nesta conversão, a entidade *ItemEstoque* é transformada em um sub-elemento do elemento que representa a entidade *Produto*, neste caso o elemento *Item*. Na sequência, o relacionamento *fatura* é também processado pela *Regra RMH*, fazendo com que *ItemNF* se transforme em um sub-elemento de *NotaFiscal*. O elemento que representa *NotaFiscal* passa a ser representado como um sub-elemento de *Pedido* no passo seguinte pela conversão do relacionamento *faturamento*. O relacionameno *emissão* é o próximo da lista de conversão, que neste caso é processado pela *Regra RMR* estabelecendo um relacionamento de referência entre *NotaFiscal* e *Parceiro*. Esta opção de conversão é selecionada pois *NotaFiscal* não poderia ser transformada em um sub-elemento direto de *Parceiro*, visto que já havia sido processada pelo relacionamento *faturamento* como um filho do elemento *Pedido*.

Dando sequência à conversão dos relacionamentos da lista, o mapeamento do relacionamento *requisição* faz com que *ItemPedido* seja representado como um sub-elemento de *Pedido* pela aplicação da *Regra RMH*. O relacionamento *reserva* é representado por um relacionamento de referência a partir do elemento *ItemPedido*, visto que a entidade *ItemPedido* não pode se transformar em um sub-elemento de *Item* pois anteriormente foi processada como filho do elemento *Pedido*. O relacionamento N:N de nome *distribuição*

é processado pela *Regra RMR*, que gera um elemento para representar este relacionamento como um sub-elemento do elemento que representa *Produto* (neste caso, o elemento *Item*). O elemento *distribuição* é aninhado ao elemento *Item*, visto que a *FAG* desta entidade é superior à *FAG* da outra entidade participante do relacionamento (*Parceiro*). Na seção seguinte são discutidos os efeitos gerados por esta escolha de mapeamento a partir da entidade que apresenta a maior *FAG* do relacionamento.

O próximo relacionamento a ser convertido é *envio*. Como a entidade *Pedido* ainda não havia sido processada como sub-elemento de algum elemento do esquema, *Pedido* transforma-se em um sub-elemento de *Parceiro*. O relacionamento 1:1 envolvendo *PedidoCompra* e *Leilão* (*arremate*) é convertido pela aplicação da *Regra RMEU*, sendo os atributos da entidade *Leilão* representados como atributos do elemento que representa *PedidoCompra*, o elemento *Pedido*. A conversão dos relacionamentos remanescentes *origem*, *entrega*, *consumo* e *oferta* estabelece relacionamentos de referência entre os elementos que representam as entidades envolvidas. Na conversão do relacionamento *entrega*, um elemento para representá-lo é criado como filho do elemento que representa a entidade associativa *fatura*, neste caso o elemento *ItemNF*. Um simples atributo de referência não poderia ser estabelecido a partir de *ItemNF*, visto que a *Regra RMR* estabelece que ocorrendo atributos no relacionamento, um elemento para representar o relacionamento deve ser criado para conter tais atributos, juntamente com atributos de referência que endereçam as demais entidades participantes.

As principais diferenças entre os esquemas lógicos gerados pela aplicação dos *Algoritmos 1* e *2* é a quantidade de elementos presentes no esquema e a estrutura de aninhamento dos elementos. A aplicação da regra para representar tipos generalização e união através de um único elemento é a principal causa do número reduzido de elementos gerados para representar estas composições no esquema da Figura 6.3. Quanto à estrutura de aninhamento dos elementos, os elementos *NotaFiscal*, *Leilão* e *ItemEstoque* foram adicionados como sub-elementos de elementos que representam entidades distintas. Por exemplo, *NotaFiscal* foi representado como um sub-elemento de *Parceiro* no esquema da Figura 6.2 e como sub-elemento de *Pedido* na Figura 6.3. Mediante a análise da carga do BD, constatou-se que o relacionamento *faturamento* apresenta uma *FAG* (2052) superior

à *FAG* do relacionamento *emissão* (1250). Esta constatação faz com que o *Algoritmo2* processe o relacionamento *faturamento* primeiramente transformando *NotaFiscal* em um sub-elemento de *Pedido*. Em um passo seguinte, o relacionamento *emissão* é processado, e como *NotaFiscal* já foi processada como filho de *Pedido*, o relacionamento entre *Parceiro* e *NotaFiscal* é estabelecido por referência. O *Algoritmo2* assume que relacionamentos com os mais altos valores para a *FAG* devem ter a preferência de conversão, justamente para que relacionamentos hierárquicos sejam gerados para representar os relacionamentos conceituais de maior impacto para a carga do BD. As consequências desta estratégia de conversão são avaliadas na próxima seção, mediante a execução das operações sobre documentos XML adequados a estes esquemas lógicos XML.

6.3 Avaliação Experimental

Nesta seção, os esquemas lógicos produzidos são avaliados quanto ao desempenho das operações sobre documentos XML adequados a estes esquemas. O desempenho das operações é medido pelo volume de acesso gerado pela operação sobre cada esquema lógico XML, utilizando-se a mesma metodologia aplicada para mensurar o volume de acesso das operações sobre o esquema conceitual. Para evidenciar o volume de acesso gerado pelas operações sobre cada esquema, consultas em *XQuery* [BOA 07] foram executadas sobre documentos XML armazenados em um SGBD XML Nativo.

6.3.1 Abordagem do Experimento

Juntamente com os esquemas produzidos pela abordagem de conversão convencional e pela abordagem de conversão baseada em análise de carga, é avaliado um esquema lógico que apresenta um aninhamento de seus elementos não apropriado para desempenhar as principais operações estimadas para o BD. Para fins de identificação, este esquema não apropriado é chamado de *esquema de pior caso* e é apresentado na Figura 6.4. O *esquema de pior caso* foi produzido artificialmente, de forma que os relacionamentos conceituais mais frequentemente acessados pelas operações estimadas são representa-

dos através de relacionamentos de referência do modelo lógico XML. O objetivo da avaliação deste esquema é a comparação com os esquemas produzidos pelas metodologias de conversão propostas por este trabalho, visto que o *esquema de pior caso* representa um esquema que poderia ser gerado pela aplicação das abordagens correlatas, como por exemplo, as metodologias propostas em [CHO 03, PIG 05, FON 06, LIU 06a]. Considera-se que tal esquema pode ser gerado pela aplicação das abordagens correlatas dada a ausência de critérios de otimização para estabelecer a estrutura XML nestas metodologias. É importante esclarecer que o *esquema de pior caso* apresentado pela Figura 6.4 não corresponde necessariamente ao pior esquema que pode ser gerado por uma destas abordagens, pois conforme discutido no final da seção 6.1, esquemas altamente fragmentados e pouco hierarquizados podem ser obtidos por tais abordagens.

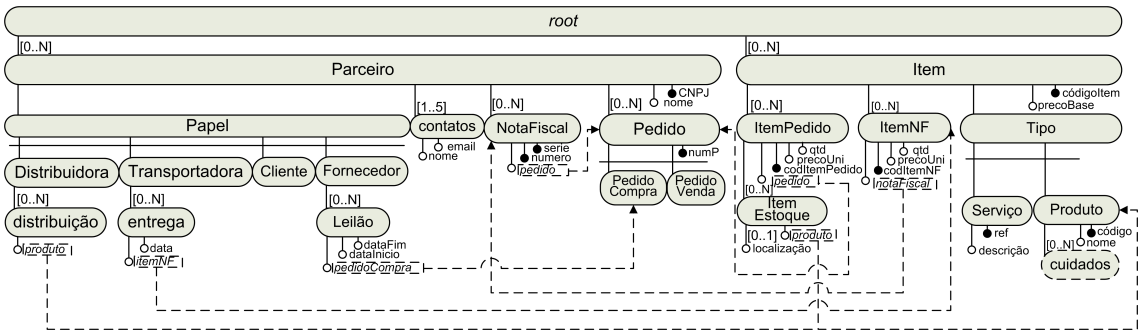

Figura 6.4: Esquema lógico XML de pior caso

Para mensurar o volume de acesso gerado pelas operações sobre os três esquemas, são consideradas as informações de volume de dados estimado para os conceitos do esquema EER (Figura 6.1) e os conceitos e a frequência estimada para cada operação levantada pela modelagem de carga do BD. A mesma metodologia aplicada para produzir os dados da Tabela 6.2 é utilizada, porém agora considerando a execução destas operações sobre a estrutura apresentada por cada um dos três esquemas lógicos XML. A Tabela 6.4 apresenta o volume de acesso gerado por cada esquema em cada conceito (entidade/relacionamento) das operações consideradas. Como exemplo, considere o esquema lógico XML da Figura 6.2 (Esquema Convencional) e a operação *O1* da Tabela 6.4. Visto que a estimativa para a execução de *O1* é 950 vezes por dia, a quantidade de acessos

sobre o elemento *Pedido* é considerada também 950 para o esquema convencional. Em virtude do conceito *faturamento* estar implicitamente representado pelo relacionamento de referência entre *NotaFiscal* e *Pedido*, considera-se que não ocorrem acessos para este conceito. Porém, para obter as notas fiscais relacionadas a um pedido é necessário comparar o identificador do elemento *Pedido* com o atributo de referência *pedido* presente em todos os elementos *NotaFiscal*. Isto significa que para obter as notas fiscais associadas aos 950 pedidos consultados por dia, é necessário comparar o valor dos identificadores destes 950 pedidos com os valores do atributo de referência para todas as 2.700 instâncias de *NotaFiscal*. Desta forma, esta junção por valor ocasiona 2.565.000 (950 * 2.700) acessos por dia sobre o elemento *NotaFiscal* do esquema convencional. A mesma operação (*O1*) gera 1.900 acessos diários sobre *NotaFiscal* no esquema otimizado, devido ao fato de *NotaFiscal* estar representada como um sub-elemento de *Pedido*. Neste caso, como a cardinalidade média para *NotaFiscal* no relacionamento *faturamento* é 2,16, para obter as notas fiscais associadas a um pedido é necessário executar uma junção por estrutura que retorna dois sub-elementos de *NotaFiscal* em média. Considerando as 950 vezes, são contabilizados 2.052 acessos.

É importante esclarecer que índices não estão sendo considerados, visto que constituem decisões do projeto físico. No entanto, observa-se que os dados produzidos pela modelagem de carga poderiam auxiliar na identificação de eventuais índices para agilizar o desempenho de consultas mais custosas.

Com a finalidade de verificar o desempenho destas operações em um SGBD XML nativo, consultas em *XQuery* foram executadas sobre documentos XML armazenados no SGBD Tamino. Documentos XML conformados a cada um dos três esquemas analisados foram armazenados no BD. As consultas *XQuery* foram produzidas de acordo com as operações consideradas neste estudo de caso. A versão *trial* do *Tamino XML Server 4.4.1* disponível em [AG 08] foi utilizada juntamente com seu *suite* de ferramentas. Em especial, as ferramentas *Tamino Manager* para a criação do BD, *Tamino X-Plorer* para a criação de coleções de documentos e as ferramentas *Tamino XQuery* e *Tamino Interactive Interface* para a execução das consultas.

Os experimentos foram realizados em uma máquina com processador Pentium IV

Tabela 6.4: Volume de acesso produzido pelas operações sobre esquemas lógicos XML

#	Frequência	Conceitos	Esquema Convencional	Esquema Otimizado	Esquema de Pior Caso
O1	950	Pedido	950	950	950
		faturamento	-	-	-
		NotaFiscal	2.565.000	2.052	2.565.000
		Total:	2.565.950	3.002	2.565.950
O2	220	PedidoVenda	220	220	220
		envio	-	-	-
		Cliente	220	220	220
		requisição	-	-	-
		ItemPedido	880	880	1.100.000
		reserva	-	-	-
		Item	316.800	316.800	880
		Total:	318.120	318.120	1.101.320
O3	150	ItemPedido	150	150	150
		reserva	-	-	-
		Produto	45.000	45.000	150
		distribuição	300	300	90.000
		Distribuidora	2.100	2.100	300
		estoque	-	-	-
		ItemEstoque	1.125.000	3.750	1.125.000
		Total:	1.172.550	51.300	1.215.600
O4	50	Leilão	50	50	50
		arremate	-	-	-
		PedidoCompra	13.500	-	13.500
		envio	-	-	-
		Fornecedor	50	50	50
		requisição	-	-	-
		ItemPedido	200	200	250.000
		Total:	13.800	300	263.600
O5	25	Parceiro	25	25	25
		emissão	-	-	-
		NotaFiscal	1.250	67.500	1.250
		fatura	-	-	-
		ItemNF	2.500	2.500	6.750.000
		Total:	3.775	70.025	6.751.275
O6	5	ItemEstoque	5	5	5
		origem	-	-	-
		ItemPedido	5	25.000	5
		requisição	-	-	-
		Pedido	5	5	6.250
		Total:	15	25.010	6.260
FREQUÊNCIA DE ACESSO TOTAL:			4.074.210	467.757	11.904.005

de 1.86GHz, equipada com 1GB de memória RAM e capacidade de 120GB de armazenamento em disco rígido. O sistema operacional utilizado foi o Windows XP Professional SP2. Todas as consultas *XQuery* executadas sobre o Tamino foram processadas sobre as mesmas condições do ambiente, utilizando a configuração padrão do *Tamino XML Server*.

Para a produção dos esquemas em *XML Schema* e dos documentos XML persistidos no BD, utilizou-se a ferramenta *XML Spy* [ALT 08]. Inicialmente, esquemas em *XML Schema* foram produzidos, os quais representam diretamente as construções lógicas dos três esquemas analisados. Os Anexos 1, 2 e 3 apresentam as definições *XML Schema* referentes aos esquemas lógicos convencional, otimizado e de pior caso, respectivamente. Estas definições em *XML Schema* foram utilizadas para a produção de documentos XML adequados aos esquemas lógicos analisados por este experimento. Para cada esquema em *XML Schema* foi gerado um documento XML também pela ferramenta *XML Spy*, produzindo um volume de dados de acordo com o que foi estimado na modelagem de carga do BD. Por exemplo, para cada documento foram geradas 2.700 instâncias do elemento *NotaFiscal*, 1.250 instâncias para o elemento *Pedido*, e assim por diante.

Para a produção de valores adequados para atributos identificadores e de referência, foi necessário o desenvolvimento de um programa para alterar os valores destes atributos nos documentos XML produzidos pela ferramenta *XML Spy*. Este programa foi desenvolvido na linguagem Java utilizando a API JDOM [jdo 08] para acesso e alteração de valores dos documentos XML. Esta pequena aplicação apenas gerou valores únicos para os atributos identificadores nos documentos, bem como alterou o valor de atributos de referência com valores válidos utilizados pelos atributos identificadores.

As consultas *XQuery* foram produzidas de acordo com o esquema de navegação entre os conceitos estabelecido para cada operação da Tabela 6.2. Os Anexos 4, 5 e 6 apresentam as consultas aplicadas sobre os documentos XML conformados aos esquemas convencional, otimizado, e de pior caso, respectivamente. Cada uma destas consultas foi executada sobre o documento XML equivalente ao esquema atendido pela consulta em questão. Após a execução da consulta utilizando a ferramenta *Tamino Interactive Interface*, um documento XML é gerado com o retorno da *XQuery* juntamente com um valor representando o tempo de processamento desta consulta. Para o retorno deste valor

Tabela 6.5: Tempo de resposta em segundos para a execução única (Ex. Única) e acumulada (Ex. Acumulada) das operações sobre documentos XML dos esquemas

#	Frequência	Esquema Convencional		Esquema Otimizado		Esquema de Pior Caso	
		Ex. Única	*Ex. Acumulada*	*Ex. Única*	*Ex. Acumulada*	*Ex. Única*	*Ex. Acumulada*
O1	950	0,785	745,750	0,158	150,100	0,888	843,600
O2	220	0,344	75,680	0,208	45,760	3,418	751,960
O3	150	1,980	297,000	0,257	38,550	3,915	587,250
O4	50	0,359	17,950	0,156	7,800	3,455	172,750
O5	25	0,157	3,925	0,219	5,475	29,603	740,075
O6	5	0,105	0,525	0,383	1,915	0,717	3,585
	Total:	3,730	1.140,830	1,381	249,600	41,996	3.099,220

referente ao processamento, foi necessário habilitar o parâmentro *duration* no momento da execução da consulta. Este parâmentro especifica o tempo gasto para processar esta requisição no *Tamino XML Server*.

A Tabela 6.5 apresenta o tempo de resposta em segundos para a execução de cada uma das 6 operações sobre os documentos XML conformados ao esquema convencional, ao esquema otimizado e ao esquema de pior caso. Para cada esquema é apresentado o tempo gasto em uma execução única (Ex. Única) de uma consulta e o tempo de ocupação diário do sistema (Ex. Acumulada) para executar uma consulta considerando sua frequência diária. Por exemplo, para o esquema convencional, uma única consulta *O1* apresenta o tempo de resposta de 0,785 segundos. Considerando a frequência de 950 vezes ao dia, o tempo diário de ocupação do sistema para executar esta operação (Ex. Acumulada) é de 745,75 segundos (0,785 * 950).

Na seção seguinte, os resultados apresentados pelas Tabelas 6.4 e 6.5 são devidamente discutidos.

6.3.2 Discussão dos Resultados

Primeiramente, comparando os valores da última linha da Tabela 6.4 com a *FAT* de 25.984 estimada para o esquema EER, verifica-se que a *FAT* para os esquemas lógicos XML aumenta consideravelmente. Este aumento se deve ao grande número de comparações ocasionadas por junções por valor. Conforme discutido anteriormente, estas junções

são necessárias para recuperar relacionamentos conceituais representados por relacionamentos de referência nos esquemas lógicos XML.

Conforme apresentado nas Figuras 6.2, 6.3 e 6.4, cada um destes esquemas lógicos estabelece um total de 7 relacionamentos por referência. Porém, a *FAT* apresentada por cada um destes esquemas é diferenciada. A maior diferença se dá entre a *FAT* do esquema otimizado (467.757 acessos diários) para a *FAT* do esquema de pior caso (11.904.005 acessos diários). Embora apresentando uma mesma quantidade de relacionamentos por referência, as estruturas de aninhamento apresentadas por estes esquemas produzem um volume de acessos diferenciado frente às operações consideradas.

O esquema otimizado apresenta um aninhamento para os elementos *NotaFiscal*, *Leilão* e *ItemEstoque* diferente do esquema convencional. Esta estrutura de aninhamento diferenciada foi produzida pela prioridade dada à conversão dos relacionamentos *faturamento*, *arremate* e *estoque* pela metodologia de conversão baseada em análise de carga. Por exemplo, a entidade *NotaFiscal* poderia ser convertida como um sub-elemento de *Parceiro* conforme apresentado pelos esquemas convencional e de pior caso, ou como um sub-elemento de *Pedido* conforme apresentado pelo esquema otimizado. Como o relacionamento *faturamento* apresenta uma *FAG* superior à *FAG* do relacionamento *emissão*, o relacionamento *faturamento* é convertido em primeira instância no esquema otimizado, transformando *NotaFiscal* em um sub-elemento de *Pedido*. Neste caso, o relacionamento *emissão* é convertido em um passo seguinte, estabelecendo um relacionamento de referência entre *NotaFiscal* e *Parceiro*. Esta estrutura de aninhamento possibilita que apenas 2.052 acessos sejam necessários para alcançar as instâncias de notas fiscais dos 950 pedidos através da operação *O1*. Para a mesma operação, os esquemas convencional e de pior caso geram 2.565.000 para obter as notas fiscais. Este número elevado de acessos é devido ao relacionamentos de referência estabelecido entre *NotaFiscal* e *Pedido* para representar o relacionamento *faturamento*.

Para a operação *O5*, a situação ocorrida para *O1* se inverte. A partir do elemento *Parceiro*, as instâncias de *NotaFiscal* são obtidas através de uma junção por estrutura nos esquemas convencional e de pior caso, visto que *NotaFiscal* é representada como sub-elemento de *Parceiro*. Neste caso, uma quantidade maior de acessos é gerada para

a obtenção das instâncias de *NotaFiscal* pelo esquema otimizado, visto que o relacionamento entre *Parceiro* e *NotaFiscal* foi estabelecido por referência. No entanto, a baixa quantidade de acessos gerada em *O1* por representar *NotaFiscal* como filho de *Pedido* supera o número de acessos gerado pelo esquema otimizado em *O5*.

O esquema otimizado também apresenta um volume de acesso mais baixo do que os outros esquemas para as operações *O3* e *O4*, devido à estrutura de aninhamento dos conceitos *ItemEstoque* e *Leilão*. Na operação *O6*, a mais elevada quantidade de acessos gerada pelo esquema otimizado é devido à representação do relacionamento *estoque* como um relacionamento de referência entre *ItemEstoque* e *ItemPedido*. Este relacionamento foi estabelecido pois a prioridade foi dada para o relacionamento *estoque*, estabelecendo *ItemEstoque* primeiramente como um filho do elemento *Item*. Na *O3*, o esquema otimizado apresenta a mais baixa quantidade de acessos justamente por conta da estrutura de aninhamento de *ItemEstoque*. Novamente, o ganho gerado por esta representação em *O3* supera a quantidade de acesso gerada em *O6*.

A representação de *ItemNF* como um sub-elemento de *NotaFiscal* e de *ItemPedido* como um sub-elemento de *Pedido* é assumida para os esquemas convencional e otimizado. Uma estrutura de aninhamento diferente para estes elementos é apresentada pelo esquema de pior caso. Esta estrutura diferenciada é a justificativa para o mais elevado volume de acesso gerado pelas operações *O2*, *O4* e *O5* sobre o esquema de pior caso.

Outro diferencial relevante está relacionado ao sentido dos relacionamentos de referência. O relacionamento *distribuição*, por exemplo, é representado como um relacionamento de referência tanto no esquema otimizado como no esquema de pior caso. Entretanto, o relacionamento é estabelecido a partir de *Item* no esquema otimizado, e a partir de *Distribuidora* no esquema de pior caso. A metodologia de conversão baseada na análise de carga estabelece os relacionamentos de referência sempre a partir do elemento que representa a entidade com o valor mais alto para a *FAG* das entidades envolvidas no relacionamento. Este tratamento é apropriado, visto que na maioria dos casos, as entidades com valores mais elevados para a *FAG* em um relacionamento são também as entidades que apresentam o maior número de instâncias. Assim, estabelecendo o relacionamento de referência a partir desta entidade, o número de acessos gerados pelas junções por valor

são minimizados, visto que os valores de um atributo de referência são comparados com as instâncias das entidades que apresentam uma menor quantidade de instâncias no relacionamento. Por exemplo, na operação *O3*, o esquema otimizado apresenta um volume de acesso mais baixo para obter as instâncias de *Distribuidora*, comparando as 300 instâncias de *distribuição* obtidas a partir do elemento *Item* com as 7 instâncias de *Parceiro* (que é o elemento que representa *Distribuidora* neste esquema). Já no esquema de pior caso, é necessário primeiro comparar as 150 instâncias de *Produto* com as 800 instâncias do elemento *distribuição* que constitui um sub-elemento de *Distribuidora*.

O tempo de resposta das consultas apresentado na Tabela 6.5 confirma a diferença no tempo de ocupação gerado pelas consultas executadas sobre o esquema otimizado frente aos demais esquemas. Observa-se que o somatório das execuções únicas de todas as consultas do esquema convencional (3,73 segundos) apresenta um valor próximo ao somatório de execução única sobre o esquema otimizado (1,381 segundos). Isso se deve ao equilíbrio gerado entre estes dois esquemas, pois o esquema otimizado apresenta um menor volume de acesso em algumas operações (*O1*, *O3*, *O4*) e o esquema convencional gera um menor volume de acessos em outras operações (*O5* e *O6*). Porém, considerando a frequência destas consultas, a execução acumulada total destes dois esquemas apresenta uma grande diferença, 249,6 segundos para o esquema otimizado e 1.140,83 para o esquema convencional. Uma das operações responsáveis por esta diferença é *O1*. A execução das 950 vezes de *O1* gera um total de 745,75 segundos para o esquema convencional e apenas 150,1 segundos para o esquema otimizado. Sendo assim, pode-se concluir que em termos de execuções isoladas das consultas, os esquemas convencional e otimizado apresentam um tempo de execução aproximado. Porém, considerado o total de execuções diárias dessas operações, o tempo de ocupação do sistema considerando o esquema otimizado é bastante reduzido.

Quanto ao esquema em pior caso, o valor total das execuções únicas já se mostra bastante superior aos demais esquemas. Este valor superior é decorrente, principalmente, da execução única da operação *05* que gera um total de 29,603 segundos, um tempo de resposta inaceitável para qualquer tipo de aplicação. Em decorrência disto e também devido à execução acumulada de *O1* e *O2*, o total da execução acumulada por este esquema

é também superior: 3.099,22 segundos.

Observa-se que a proporção entre o total de acessos de cada operação apresentado pela Tabela 6.4 e a execução acumulada de uma operação na Tabela 6.5 não é sempre equivalente. Por exemplo, embora o total de acessos gerado por *O2* no esquema convencional e otimizado sejam iguais (318.120), a execução acumulada sofre uma variação: 75,68 segundos para o esquema convencional e 45,76 para o esquema otimizado. Assume-se que estas diferenças podem ser ocasionadas por variações no tamanho dos documentos considerados e características específicas das instâncias utilizadas para cada documento em cada execução de consulta. Estas e outras variações que podem ocorrer não são avaliadas por este estudo, visto que o objetivo do experimento realizado sobre o Tamino é apenas confirmar a diferença entre o volume de acesso gerado entre os esquemas de acordo com a Tabela 6.4.

Embora as operações consideradas neste estudo de caso não representam um grande volume de acesso diário ao BD, é possível constatar, pela comparação realizada, que as informações da carga estimada para o BD guiam o processo de conversão para a geração de um esquema XML mais otimizado. O tempo de ocupação diário ocasionado pelos esquemas que não consideram tais informações (esquema convencional e esquema de pior caso) é consideravelmente superior ao tempo produzido pelo esquema otimizado.

Observa-se que o ganho gerado pela aplicação da metodologia baseada em análise de carga é dependente das otimizações possíveis de serem aplicadas sobre o esquema conceitual e as informações de carga fornecidas. A existência de diversas opções para o aninhamento dos conceitos do esquema e o indicativo de que determinados aninhamentos têm maior impacto nas operações, constituem dados que permitem que otimizações sejam realizadas no esquema XML. Algumas situações podem minimizar o efeito da aplicação da metodologia baseada em análise de carga, como por exemplo a inexistência de opções de aninhamento entre os conceitos do esquema, ou a indicação de que as opções de aninhamento apresentam um impacto equilibrado para a carga do BD. Tal metodologia tende a apresentar ganhos mais significativos para casos em que detecta-se a existência de operações que visivelmente serão executadas com maior frequência, gerando grande carga para o BD.

Capítulo 7

Conclusão

Este trabalho apresenta uma abordagem de conversão para o projeto lógico de BDs XML. A abordagem proposta estabelece um processo de conversão de esquemas conceituais definidos pelo modelo EER em esquemas XML definidos por um modelo lógico XML. Tal proposta é constituída por regras de conversão capazes de transformar cada um dos construtores do modelo conceitual EER em uma representação lógica abstrata quanto aos modelos de implementação XML. Um processo de conversão EER-XML automático é também proposto, com a finalidade de ordenar a aplicação das regras na produção de um esquema XML que não habilita a redundância de dados mas ao mesmo tempo constitui uma representação bem estruturada das informações modeladas pelo projeto conceitual.

Esquemas XML otimizados são produzidos pela consideração de informações relativas à principal carga estimada para o BD que está sendo modelado. As otimizações realizadas no esquema XML durante o processo de conversão tendem a produzir ganhos significativos no desempenho de consultas previstas durante o projeto lógico como o conjunto de operações que produzirão a maior carga do BD. O estudo de caso apresentado demonstra o ganho obtido por documentos conformados a estes esquemas no tempo de ocupação diário do sistema para a execução das operações mais frequentes do BD.

Este capítulo apresenta as contribuições deste trabalho, juntamente com sugestões para trabalhos futuros.

7.1 Contribuições

A principal contribuição deste trabalho é uma proposta para produção de esquemas XML adequada aos objetivos do projeto lógico de um BD. Desta forma, acredita-se estar contribuindo para a área de gerência de dados XML rumo à consolidação de uma metodologia para o projeto de bancos de dados XML. Mesmo baseando-se em antigas soluções, este trabalho aplica princípios e métodos de modelagem solidificados para o projeto de BDs tradicionais com a finalidade de propor soluções para a modelagem de uma natureza de dados mais recente, no caso, dados XML.

Dentre as demais contribuições deste trabalho destacam-se:

- **Um conjunto de regras de conversão que considera todos os construtores de um modelo conceitual para a produção de composições XML equivalentes**. Diferente de trabalhos relacionados, o conjunto de regras proposto provê alternativas de conversão para todos os construtores do modelo EER, conforme apresentado pela última linha da tabela 7.1. Generalizações e tipos união, construções pouco tratadas por trabalhos correlatos, são consideradas pelas regras propostas, visto que constituem construtores fundamentais para a representação de tipos de dados complexos [SMI 77, AK 05].

Tabela 7.1: Comparativo entre Algoritmos de Conversão Conceitual-XML

Abordagens	Entidades/Classes	Atributos	Generalizações	Tipos União	Relacionamentos
[LEE 01]	+	-			+
[ELM 02]	-	-			-
[CHO 03]	+	+	-		+
[PIG 05]	+	-			+
[FON 06]	+	-	-	-	+
[LIU 06a]	+	+	-		+
[WIW 06, JAG 04]	-	-			-
[ROU 02, BIR 00]	-	-			-
[QUA 05]	+	-			-
[KRU 03, KUD 03]	-	-			-
[MOK 06, EMB 01]	-				-
[MAN 04]	+	-			+
EER-XML Lógico	**+**	**+**	**+**	**+**	**+**

- **Um processo de conversão EER-XML flexível quanto aos modelos de implementação XML**. O modelo lógico XML que determina as composições geradas pelo processo de conversão, bem como as estratégias de conversão consideradas possibilitam que o esquema produzido seja facilmente traduzido para um esquema de implementação em alguma das linguagens de definição de esquema XML. Grande parte dos trabalhos correlatos da área são dependentes de uma linguagem de definição de esquema XML, propondo soluções de conversão diretamente para algum modelo de implementação específico. A abordagem proposta atende adequadamente aos níveis conceitual e lógico do projeto de BD, conforme apresentado pela última linha da tabela 7.2. O modelo conceitual (EER) empregado pela proposta em questão é independente de qualquer modelo lógico ou de implementação, e o modelo lógico XML constitui uma representação abstrata para os diferentes modelos de implementação XML (DTD/XML Schema);

Tabela 7.2: Comparativo Geral entre os Trabalhos Relacionados e a Abordagem Proposta

Abordagens	**Modelo Conceitual**	**Modelo Lógico**	**Modelo de Implementação**
[ELM 02]	EER	estruturas hierárquicas	XML-Schema
[CHO 03]	EER	-	DTD
[PIG 05]	ER	-	XML Schema
[FON 06]	EER	-	DTD
[LIU 06a]	EER	-	XML Schema
[WIW 06, JAG 04]	ER	MCT Schema	-
[ROU 02, BIR 00]	UML/ORM	UML+estereótipos	XML Schema
[CON 00, ECK 04]	-	UML+estereótipos	DTD/XML Schema
[COM 06]	-	UXS	XML Schema
[LIU 06b]	UML	XUML	XML Schema
[QUA 05]	UML	-	XML Schema
[KUD 03, KRU 03]	UML	-	DTD/XML Schema
[MOK 06]	Modelo genérico	estruturas hierárquicas	-
[SEN 03]	-	XER	XML Schema
[PSA 01]	-	ERX	XML Schema
[MAN 04]	EReX	XGrammar	-
[EMB 04, AK 07]	C-XML	-	XML Schema
[DOB 01]	-	ORA-SS	DTD
[CHA 02]	-	Redes Semânticas	XML Schema
EER-XML Lógico	**EER**	**XML Lógico**	**DTD/XML Schema**

- **Um processo automático para a conversão de um esquema EER em um esquema lógico XML**. O algoritmo de conversão proposto, juntamente com seus procedimentos, define a execução das regras de conversão sobre os fragmentos de um esquema conceitual com a finalidade de produzir automaticamente um esquema lógico XML equivalente. Tal algoritmo define a estratégia de conversão adequada para um fragmento conceitual baseado nas restrições impostas pelo esquema conceitual. Embora um processo semi-automático possa ser considerado neste contexto, a conversão automática tende a ser mais adequada, principalmente por aplicações que necessitam produzir um grande volume de dados XML diariamente. Uma ferramenta que implementa este processo de conversão automático foi produzida em conjunto com a realização desta dissertação [LIM 08a];

- **Definição de métricas para determinar a influência dos conceitos de um esquema EER na carga estimada para um BD**. Os conceitos de *Frequência de Acesso Geral*, *Frequência de Acesso Total* e *Frequência de Acesso Mínima* são propostos para auxiliar o processo de conversão na identificação de conceitos relevantes e de impacto para a carga do BD;

- **Um processo automático para a produção de esquemas XML otimizados quanto à carga estimada para um BD**. Embora existam trabalhos relacionados focados na aplicação de otimizações na estrutura XML, como [MOK 06] e [BIR 00], tais trabalhos promovem a estruturação dos esquemas com base apenas em restrições impostas pelo esquema conceitual. Um dos trabalhos mais relevantes neste contexto foi proposto por Wiwatwattana, et al. [WIW 06], oferecendo uma solução para eliminar o uso de relacionamentos de referência em esquemas XML. No entanto, conforme discutido pelo capítulo 3, a necessidade de adaptações nas recomendações das linguagens XML e o *overhead* gerado pelo uso de esquemas de *multi-cores* constituem fortes desvantagens para a aplicação de tal abordagem. Os experimentos realizados demonstram que esquemas lógicos XML gerados pelo processo de conversão baseado em análise da carga proposto por este trabalho são capazes de responder com mais eficiência às principais operações do BD, gerando

um tempo de ocupação diário do sistema bastante reduzido se comparado à esquemas não otimizados quanto às operações mais frequentes previstas para um banco de dados.

7.2 Trabalhos Futuros

Atividades futuras relacionadas a este trabalho incluem a consideração do mapeamento de outros modelos conceituais como a UML e a extensão da metodologia para considerar o projeto de implementação de BDs XML. Embora esquemas de implementação possam ser facilmente produzidos a partir dos esquemas lógicos gerados pela abordagem proposta por este trabalho, algumas problemáticas devem ser tratadas em virtude de recursos diferenciados e limitações impostas por cada uma das linguagens de definição de esquema XML.

Considera-se como principal interesse na continuação deste trabalho o aprimoramento do processo de conversão baseado em análise de carga para atuar em procedimentos de *tuning* de banco de dados. Uma das sugestões é o uso desse processo juntamente com técnicas de engenharia reversa para promover o reprojeto de documentos XML. Esta metodologia poderia ser utilizada por aplicações cujos esquemas XML estão em constante evolução. Neste contexto, as operações frequentemente executadas por estas aplicações poderiam servir como parâmetros para reprojeto destes esquemas. Diante deste cenário, técnicas efetivas devem ser definidas para a produção das estimativas do volume de dados e das principais operações executadas sobre o BD. Além disto, a abordagem proposta por este trabalho poderia ser utilizada para a sugestão de índices a serem criados para melhorar o acesso às porções da estrutura XML mais frequentemente acessadas. Uma ferramenta que implementa o processo baseado em análise de carga também está sendo considerada como próxima atividade.

Em virtude do custo gerado por relacionamentos de referência presentes em esquemas XML, considera-se a possibilidade de estender a metodologia para reduzir o número deste tipo de relacionamento em um esquema XML. Para este fim, sugere-se identificar o tipo de acesso gerado (consulta/atualização) pelas operações estimadas durante a mo-

delagem de carga do BD. Com este novo levantamento, seria possível eliminar o uso de relacionamentos de referência para os casos em que se identifica que o relacionamento é relevante para a carga do BD, porém os conceitos envolvidos são pouco acessados para fins de atualização. Lembrando que referências são utilizadas pela abordagem de conversão atual para evitar a redundância de dados, esta nova abordagem a ser proposta permitiria a redundância de dados em alguns pontos importantes do esquema, porém as anomalias de atualização seriam controladas e minimizadas se os conceitos envolvidos nestes pontos de redundância são pouco acessados para fins de atualização.

Em BDs XML, nem sempre os documentos de uma coleção são conformados a um mesmo esquema. Um dos motivos para isto é o fato de que esquemas XML não constituem um requisito para a persistência dos documentos XML na maioria dos SGBDs XML existentes. Além disto, considerando o meio de aplicação do padrão XML, o grande volume de dados gerado pelas aplicações produz diversos documentos para representar um mesmo domínio, utilizando geralmente uma sintaxe diferenciada. Diante disto, soluções de integração devem ser consideradas para a unificação de consultas sobre estes documentos. A metodologia proposta por este trabalho pode ser utilizada neste contexto tanto para o reprojeto dos documentos XML tanto para a geração de um esquema unificado que possa responder de forma mais eficiente às consultas mais importantes da aplicação.

Referências Bibliográficas

[AG 08] AG, S. **Web Methods Tamino Software - XML Starter Kit**. http://www.softwareag.com/Corporate/products/wm/tamino/downloads/download.asp.

[AK 05] AL-KAMHA, R.; EMBLEY, D. W.; LIDDLE, S. W. Representing generalization/specialization in xml schema. In: EMISA: ENTERPRISE MODELLING AND INFORMATION SYSTEMS ARCHITECTURES, 2005. GI, 2005. v.75 of **LNI**, p.250–263.

[AK 07] AL-KAMHA, R.; EMBLEY, D. W.; LIDDLE, S. W. Augmenting traditional conceptual models to accommodate xml structural constructs. In: ER 2007 - 26TH INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, 2007. Springer, 2007. v.4801 of **Lecture Notes in Computer Science**, p.518–533.

[ALT 08] ALTOVA. **XMLSpy - XML editor for modeling, editing, transforming and debugging XML technologies**. http://www.altova.com/xmlspy/.

[BAT 92] BATINI, C.; CERI, S. N. S. **Conceptual Database Design: An Entity-Relationship Approach**. The Benjamin/Cummings Publishing Company, 1992.

[BIR 00] BIRD, L.; GOODCHILD, A. H. T. A. Object role modeling and xml-schema. In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, 2000. Springer Heidelberg, 2000. p.661–705.

[BOA 07] BOAG, S, . E. A. **XQuery 1.0: An XML Query Language (W3C Recommendation)**. http://www.w3.org/TR/xquery/.

[BOU 05] BOURRET, R. **XML and Databases**. http://www.rpbourret.com/xml/XMLAndDatabases.htm#isxmladatabase.

[BRA 00] BRAY, T.;PAOLI, J. E. A. **Extensible Markup Language (XML) 1.0 W3C Recommendation**. http://www.w3.org/TR/REC-xml.

[CAN 05] CANDIDO, C. H. **Aprendizagem em Banco de Dados: Implementação de Ferramenta de Modelagem ER**. Universidade Federal de Santa Catarina, 2005. Monografia de especialização.

[CHA 02] CHANG, E. E. A. A semantic network-based design methodology for xml documents. In: ACM TRANSACTIONS ON INFORMATION SYSTEMS, 2002. [s.n.], 2002. p.390–421.

[CHE 76] CHEN, P. P. The entity-relationship model - toward a unified vies of data. In: ACM TRANS. DATABASE SYSTEMS, 1976. [s.n.], 1976. v.1.

[CHO 03] CHOI, M.; LIM, J. J. K. Developing a unified design methodology based on extended entity-relationship model for xml. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE, 2003. Springer Heidelberg, 2003. p.920–929.

[COM 06] COMBI, C.; OLIBONI, B. Conceptual modeling of xml data. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 2006. [s.n.], 2006. p.467–473.

[CON 00] CONRAD, R.; SCHEFFNER, D. F. J. C. Xml conceptual modeling using xml. In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, 2000. Springer Heidelberg, 2000. p.558–571.

[DIL 05] DILLON, S. **XML to Relational: Bridging the Gap**. Oracle Magazine.

[DOB 01] DOBBIE, G. E. A. Ora-ss: An object-relationship-attribute model for semi-structured data. National University of Singapore, 2001. Relatório TécnicoTechnical Report TR21/00.

[ECK 04] ECKSTEIN, R.; ECKSTEIN, S. Conceptual modeling xml schemata using uml. In: CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING, 2004. [s.n.], 2004. p.122–131.

[ELM 85] ELMASRI, R.; WEELDREYER, J. A. H. A. R. The category concept: An extension to the entity-relationship model. In: DATA KNOWLEDGE ENGINEERING, 1985. [s.n.], 1985. Number1, p.75–116.

[ELM 00] ELMASRI, R.; NAVATHE, S. B. **Fundamentals of Database Systems**. 3. ed. Addison-Wesley, 2000.

[ELM 02] ELMASRI, R.; WU, Y. E. A. Conceptual modeling for customized xml schemas. In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, 2002. [s.n.], 2002. p.429–443.

[EMB 98] EMBLEY, D. **Object Database Development: Concepts and Principles**. Addison Wesley, 1998.

[EMB 01] EMBLEY, D. W.; MOK, W. Y. Developing xml documents with guaranteed good properties. In: ER '01: 20TH INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, 2001. **Proceedings...** London, UK: Springer-Verlag, 2001. p.426–441.

[EMB 04] EMBLEY, D.; LIDDLE, S. K. S. Enterprise modeling with conceptual xml. In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, 2004. [s.n.], 2004. p.150–165.

[ENN 00] ENNSER, L. E. A. **Integrating XML with DB2 XML Extender and DB2 Text Extender**. IBM Redbooks.

[FON 06] FONG, J.; FONG, A. W. H. K. Y. P. Translating relational schema with constraints into xml schema. In: INTERNATIONAL JOURNAL OF SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, 2006. [s.n.], 2006. Number16, p.201–244.

[HAL 96] HALPIN, T. Business rules and object-role modeling. In: DATABASE PROGRAMMING AND DESIGN, 1996. [s.n.], 1996. v.9, p.66–72.

[HAL 98] HALPIN, T. **UML Data Models From An ORM Prospective - Part 1-10**.

[HEU 04] HEUSER, C. A. **Projeto de Banco de Dados**. 5. ed. Instituto de Informática da UFRGS, 2004.

[JAG 02] JAGADISH, H.; AL-KHALIFA, S. E. A. Timber: A native xml database. In: INTERNATIONAL JOURNAL ON VERY LARGE DATABASES, 2002. Springer-Verlag New York, 2002. v.4, p.274–291.

[JAG 04] JAGADISH, H. V. et al. Colorful xml: one hierarchy isn't enough. In: SIGMOD '04: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2004. **Proceedings...** New York, NY, USA: ACM, 2004. p.251–262.

[jdo 08] **JDOM: Documentation**. http://www.jdom.org/downloads/docs.html.

[KRU 03] KRUMBEIN, T.; KUDRASS, T. Rule-based generation of xml schemas from uml class diagrams. In: BERLINER XML TAGE, 2003. XML-Clearinghouse, 2003. p.213–227.

[KUD 03] KUDRASS, T.; KRUMBEIN, T. Rule-based generation of xml dtds from uml class diagrams. In: 7TH EAST EUROPEAN CONFERENCE ON ADVANCES IN DATABASES AND INFORMATION SYSTEMS, 2003. Springer, 2003. v.2798 of **Lecture Notes in Computer Science**, p.339–354.

[LEE 01] LEE, M. E. A. Designing semistructured databases: A conceptual approach. In: INTERNATIONAL CONFERENCE ON DATABASE EXPERT SYSTEMS APPLICATIONS, 2001. [s.n.], 2001. p.12–21.

[LIM 08a] LIMA, C. **Uma Ferramenta para Conversão de Esquemas Conceituais ERE para Esquemas Lógicos XML**. Trabalho de Conclusão de Curso. Departamento de Informática e Estatística da UFSC.

[LIM 08b] LIMA, C.; SCHROEDER, R. M. R. S. Um processo de conversão de esquemas conceituais ere para esquemas lógicos xml. In: SESSãO DE PôSTERES DO SIMPóSIO BRASILEIRO DE BANCO DE DADOS, 2008. [s.n.], 2008.

[LIM 08c] LIMA, C.; SCHROEDER, R. M. R. S. Uma ferramenta para conversão de esquemas conceituais eer para esquemas lógicos xml. In: IV ESCOLA REGIONAL DE BANCO DE DADOS, 2008. [s.n.], 2008. p.109–118.

[LIU 06a] LIU, C.; LI, J. Designing quality xml schemas from e-r diagrams. In: ADVANCES IN WEB-AGE INFORMATION MANAGEMENT, 2006. Springer Heidelberg, 2006. p.508–519.

[LIU 06b] LIU, H.; LU, Y. Y. Q. Xml conceptual modeling with xuml. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2006. [s.n.], 2006. p.973–976.

[MAN 04] MANI, M. Erex: A conceptual model for xml. In: INTERNATIONAL XML DATABASE SYMPOSIUM, 2004. Springer-Verlag, 2004. p.128–142.

[MOK 06] MOK, W. Y., E. D. W. Generating compact redundancy-free xml documents from conceptual-model hypergraphs. In: IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2006. [s.n.], 2006. Number18, p.1082–1096.

[MOR 07] MORO, M. M.; LIM, L.; CHANG, Y.-C. Schema advisor for hybrid relational-xml dbms. In: SIGMOD '07: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2007. **Proceedings...** New York, NY, USA: ACM, 2007. p.959–970.

[PIG 05] PIGOZZO, P; QUINTARELLI, E. An algorithm for generating xml schemas from er schemas. In: ITALIAN SYMPOSIUM ON ADVANCED DATABASE SYSTEMS, 2005. [s.n.], 2005. p.192–199.

[PSA 01] PSAILA, G.; BRUGALI, D. The erx data management system. In: INTERNATIONAL CONFERENCE ON INTERNET COMPUTING, 2001. [s.n.], 2001.

[QUA 05] QUANG, N. H.; RAHAYU, J. W. Xml schema design approach. **IJWIS**, [S.l.], v.1, n.3, p.161–178, 2005.

[ROU 02] ROUTLEDGE, N.; BIRD, L. G. A. Uml and xml schema. In: Society, I. C., editor, AUSTRALIAN DATABASE CONFERENCE, 2002. [s.n.], 2002. p.157–166.

[SAL 01] SALMINEN, A.; TOMPA, F. W. Requirements for xml document database systems. In: ACM SYMPOSIUM ON DOCUMENT ENGINEERING, 2001. [s.n.], 2001. p.85–94.

[SCH 01] SCHöNING, H. Tamino - a dbms designed for xml-schema. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 2001. IEEE Computer Society, 2001. p.149–154.

[SCH 07] SCHROEDER, R.; MELLO, R. S. Análise de abordagens para modelagem conceitual e lógica xml. In: III ESCOLA REGIONAL DE BANCO DE DADOS, 2007. [s.n.], 2007. p.144–153.

[SCH 08a] SCHROEDER, R.; MELLO, R. S. Conversion of generalization hierarchies and union types from extended entity-relationship model to an xml logical model. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 2008. ACM Press, 2008. p.1036–1037.

[SCH 08b] SCHROEDER, R.; MELLO, R. S. Improving query performance on xml documents: A workload-driven design approach. In: DOCENG'08: ACM SYMPOSIUM ON DOCUMENT ENGINEERING, 2008. [s.n.], 2008.

[SCH 08c] SCHROEDER, R.; MELLO, R. S. Xml document modeling from a conceptual schema. In: WORKSHOP ON ONTOLOGIES AND METAMODELING IN SOFTWARE AND DATA ENGINEERING (EM CONJUNTO COM O SBBD), 2008. [s.n.], 2008.

[SEN 03] SENGUPTA, A.; MOHAN, S. D. R. Xer extensible entity relationship modeling. In: XML 2003 CONFERENCE, 2003. [s.n.], 2003. p.140–154.

[SMI 77] SMITH, J. M.; SMITH, D. C. P. Database abstractions: Aggregation and generalization. In: ACM TRANSACTIONS ON DATABASE SYSTEMS, 1977. IEEE Computer Society, 1977. v.2, p.105–133.

[TEO 86] TEOREY, T. J.; YANG, D. F. J. P. A logical design methodology for relational databases using the extended entity-relationship model. In: ACM COMPUTING SURVEYS, 1986. [s.n.], 1986. v.18, p.197–222.

[THO 04] THOMPSON, H.; BEECH, D. E. A. **XML Schema Part 1: Structures W3C Recommendation**. http://www.w3.org/TR/2004/REC-xmlschema-1-20041028.

[w3c 08] **W3C - World Wide Web Consortium**. http://www.w3.org/.

[WIW 06] WIWATWATTANA, N. E. A. Making designer schemas with colors. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 2006. IEEE Computer Society, 2006.

[XU 03] XU, Z. E. A. Dynamic tuning of xml storage schema in vxmlr. In: INTERNATIONAL DATABASE ENGINEERING AND APPLICATIONS SYMPOSIUM, 2003. IEEE Computer Society, 2003. p.76–86.

Anexos

Anexo 1 – *XML Schema* gerado a partir do esquema lógico convencional

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="root">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="parceiro" minOccurs="0" maxOccurs="54">
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:element name="papel">
                        <xsd:complexType>
                           <xsd:choice>
                              <xsd:element name="transportadora">
                                 <xsd:complexType>
                                    <xsd:sequence>
                                       <xsd:element name="entrega" minOccurs="0" maxOccurs="216">
                                          <xsd:complexType>
                                             <xsd:attribute name="data"/>
                                             <xsd:attribute name="ref_coditemnf"/>
                                          </xsd:complexType>
                                       </xsd:element>
                                    </xsd:sequence>
                                 </xsd:complexType>
                              </xsd:element>
                              <xsd:element name="distribuidora"/>
                              <xsd:element name="fornecedor">
                                 <xsd:complexType>
                                    <xsd:sequence>
                                       <xsd:element name="leilão" minOccurs="0" maxOccurs="5">
                                          <xsd:complexType>
                                             <xsd:attribute name="datafim"/>
                                             <xsd:attribute name="datainicio"/>
                                             <xsd:attribute name="ref_nump"/>
                                          </xsd:complexType>
                                       </xsd:element>
                                    </xsd:sequence>
                                 </xsd:complexType>
                              </xsd:element>
                              <xsd:element name="cliente"/>
                           </xsd:choice>
                        </xsd:complexType>
                     </xsd:element>
                     <xsd:element name="contatos" maxOccurs="5">
                        <xsd:complexType>
                           <xsd:attribute name="email"/>
                           <xsd:attribute name="nome"/>
                        </xsd:complexType>
                     </xsd:element>
                     <xsd:element name="notafiscal" minOccurs="0" maxOccurs="50">
                        <xsd:complexType>
                           <xsd:sequence>
```

```
                <xsd:element name="itemnf" maxOccurs="2">
                  <xsd:complexType>
                    <xsd:attribute name="qtd"/>
                    <xsd:attribute name="id_coditemnf"/>
                    <xsd:attribute name="precouni"/>
                    <xsd:attribute name="ref_codigoitem"/>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
              <xsd:attribute name="id_serie"/>
              <xsd:attribute name="id_numero"/>
              <xsd:attribute name="ref_nump"/>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="pedido" minOccurs="0" maxOccurs="23">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="tipo">
                  <xsd:complexType>
                    <xsd:choice>
                      <xsd:element name="pedidovenda"/>
                      <xsd:element name="pedidocompra"/>
                    </xsd:choice>
                  </xsd:complexType>
                </xsd:element>
                <xsd:element name="itempedido" maxOccurs="4">
                  <xsd:complexType>
                    <xsd:sequence>
                      <xsd:element name="itemestoque" minOccurs="0" maxOccurs="2">
                        <xsd:complexType>
                          <xsd:attribute name="ref_codigo"/>
                          <xsd:attribute name="localizacao" use="optional"/>
                        </xsd:complexType>
                      </xsd:element>
                    </xsd:sequence>
                    <xsd:attribute name="qtd"/>
                    <xsd:attribute name="precouni"/>
                    <xsd:attribute name="ref_codigoitem"/>
                    <xsd:attribute name="id_coditempedido"/>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
              <xsd:attribute name="id_nump"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="id_cnpj"/>
        <xsd:attribute name="nome"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="item" minOccurs="0" maxOccurs="360">
      <xsd:complexType>
        <xsd:choice>
          <xsd:element name="servico">
            <xsd:complexType>
              <xsd:attribute name="id_ref"/>
              <xsd:attribute name="descricao"/>
            </xsd:complexType>
```

```
            </xsd:element>
            <xsd:element name="produto">
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:element name="distribuicao" maxOccurs="2">
                        <xsd:complexType>
                           <xsd:attribute name="ref_cnpj"/>
                        </xsd:complexType>
                     </xsd:element>
                     <xsd:element name="cuidados" minOccurs="0" maxOccurs="5"/>
                  </xsd:sequence>
                  <xsd:attribute name="id_codigo"/>
                  <xsd:attribute name="nome"/>
               </xsd:complexType>
            </xsd:element>
         </xsd:choice>
         <xsd:attribute name="id_codigoitem"/>
         <xsd:attribute name="precobase"/>
      </xsd:complexType>
   </xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:keyref name="rtransportadora" refer="kitemnf">
   <xsd:selector xpath=".//entrega"/>
   <xsd:field xpath="@ref_coditemnf"/>
</xsd:keyref>
<xsd:keyref name="rleilao" refer="kpedido">
   <xsd:selector xpath=".//leilao"/>
   <xsd:field xpath="@ref_nump"/>
</xsd:keyref>
<xsd:keyref name="rnotafiscal" refer="kpedido">
   <xsd:selector xpath=".//notafiscal"/>
   <xsd:field xpath="@ref_nump"/>
</xsd:keyref>
<xsd:keyref name="ritemnf" refer="kitem">
   <xsd:selector xpath=".//itemnf"/>
   <xsd:field xpath="@ref_codigoitem"/>
</xsd:keyref>
<xsd:keyref name="ritempedido" refer="kitem">
   <xsd:selector xpath=".//itempedido"/>
   <xsd:field xpath="@ref_codigoitem"/>
</xsd:keyref>
<xsd:keyref name="ritemestoque" refer="kproduto">
   <xsd:selector xpath=".//itemestoque"/>
   <xsd:field xpath="@ref_codigo"/>
</xsd:keyref>
<xsd:keyref name="rdistribuidora" refer="kparceiro">
   <xsd:selector xpath=".//distribuidora"/>
   <xsd:field xpath="@ref_cnpj"/>
</xsd:keyref>
<xsd:key name="knotafiscal">
   <xsd:selector xpath=".//notafiscal"/>
   <xsd:field xpath="@id_serie"/>
   <xsd:field xpath="@id_numero"/>
</xsd:key>
<xsd:key name="kitemnf">
   <xsd:selector xpath=".//itemnf"/>
   <xsd:field xpath="@id_coditemnf"/>
```

```
    </xsd:key>
    <xsd:key name="kparceiro">
      <xsd:selector xpath=".//parceiro"/>
      <xsd:field xpath="@id_cnpj"/>
    </xsd:key>
    <xsd:key name="kpedido">
      <xsd:selector xpath=".//pedido"/>
      <xsd:field xpath="@id_nump"/>
    </xsd:key>
    <xsd:key name="kitempedido">
      <xsd:selector xpath=".//itempedido"/>
      <xsd:field xpath="@id_coditempedido"/>
    </xsd:key>
    <xsd:key name="kitem">
      <xsd:selector xpath=".//item"/>
      <xsd:field xpath="@id_codigoitem"/>
    </xsd:key>
    <xsd:key name="kservico">
      <xsd:selector xpath=".//servico"/>
      <xsd:field xpath="@id_ref"/>
    </xsd:key>
    <xsd:key name="kproduto">
      <xsd:selector xpath=".//produto"/>
      <xsd:field xpath="@id_codigo"/>
    </xsd:key>
  </xsd:element>
</xsd:schema>
```

Anexo 2 – *XML Schema* gerado a partir do esquema lógico otimizado

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="root">
    <xs:annotation>
      <xs:documentation>Comment describing your root element</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="parceiro" minOccurs="0" maxOccurs="54">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="pedido" minOccurs="0" maxOccurs="23">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="notafiscal" minOccurs="0" maxOccurs="2">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="itemnf" minOccurs="0" maxOccurs="2">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="entrega" minOccurs="0">
                                  <xs:complexType>
                                    <xs:attribute name="data"/>
                                    <xs:attribute name="ref_cnpj"/>
                                  </xs:complexType>
                                </xs:element>
```

```
                                    </xs:sequence>
                                    <xs:attribute name="id_coditemnf"/>
                                    <xs:attribute name="qtd"/>
                                    <xs:attribute name="precouni"/>
                                    <xs:attribute name="ref_codigoitem"/>
                                 </xs:complexType>
                              </xs:element>
                           </xs:sequence>
                           <xs:attribute name="id_serie"/>
                           <xs:attribute name="id_numero"/>
                           <xs:attribute name="ref_cnpj"/>
                        </xs:complexType>
                     </xs:element>
                     <xs:element name="itempedido" maxOccurs="2">
                        <xs:complexType>
                           <xs:attribute name="qtd"/>
                           <xs:attribute name="precouni"/>
                           <xs:attribute name="ref_codigoitem"/>
                           <xs:attribute name="id_coditempedido"/>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
                  <xs:attribute name="id_nump"/>
                  <xs:attribute name="datainicio" use="optional"/>
                  <xs:attribute name="datafim" use="optional"/>
                  <xs:attribute name="tipo"/>
                  <xs:attribute name="ref_cnpj"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="contatos" maxOccurs="5">
               <xs:complexType>
                  <xs:attribute name="email"/>
                  <xs:attribute name="nome"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         <xs:attribute name="id_cnpj"/>
         <xs:attribute name="nome"/>
         <xs:attribute name="papel"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="item" minOccurs="0" maxOccurs="360">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="distribuicao" maxOccurs="2">
               <xs:complexType>
                  <xs:attribute name="ref_cnpj"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="itemestoque" minOccurs="0" maxOccurs="25">
               <xs:complexType>
                  <xs:attribute name="ref_coditempedido"/>
                  <xs:attribute name="localizacao" use="optional"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="cuidados" minOccurs="0" maxOccurs="5"/>
         </xs:sequence>
         <xs:attribute name="id_codigoitem"/>
```

```
            <xs:attribute name="id_ref" use="optional"/>
            <xs:attribute name="descricao" use="optional"/>
            <xs:attribute name="id_codigo" use="optional"/>
            <xs:attribute name="nome" use="optional"/>
            <xs:attribute name="precobase"/>
          </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:keyref name="rentrega" refer="kparceiro">
    <xs:selector xpath=".//entrega"/>
    <xs:field xpath="@ref_cnpj"/>
</xs:keyref>
<xs:keyref name="rnotafiscal" refer="kparceiro">
    <xs:selector xpath=".//notafiscal"/>
    <xs:field xpath="@ref_cnpj"/>
</xs:keyref>
<xs:keyref name="ritemnf" refer="kitem">
    <xs:selector xpath=".//itemnf"/>
    <xs:field xpath="@ref_codigoitem"/>
</xs:keyref>
<xs:keyref name="ritempedido" refer="kitem">
    <xs:selector xpath=".//itempedido"/>
    <xs:field xpath="@ref_codigoitem"/>
</xs:keyref>
<xs:keyref name="rpedido" refer="kparceiro">
    <xs:selector xpath=".//pedido"/>
    <xs:field xpath="@ref_cnpj"/>
</xs:keyref>
<xs:keyref name="rdistribuidora" refer="kparceiro">
    <xs:selector xpath=".//distribuicao"/>
    <xs:field xpath="@ref_cnpj"/>
</xs:keyref>
<xs:keyref name="ritemestoque" refer="kitempedido">
    <xs:selector xpath=".//itemestoque"/>
    <xs:field xpath="@ref_coditempedido"/>
</xs:keyref>
<xs:key name="knotafiscal">
    <xs:selector xpath=".//notafiscal"/>
    <xs:field xpath="@id_serie"/>
    <xs:field xpath="@id_numero"/>
</xs:key>
<xs:key name="kitemnf">
    <xs:selector xpath=".//itemnf"/>
    <xs:field xpath="@id_coditemnf"/>
</xs:key>
<xs:key name="kparceiro">
    <xs:selector xpath=".//parceiro"/>
    <xs:field xpath="@id_cnpj"/>
</xs:key>
<xs:key name="kpedido">
    <xs:selector xpath=".//pedido"/>
    <xs:field xpath="@id_nump"/>
</xs:key>
<xs:key name="kitempedido">
    <xs:selector xpath=".//itempedido"/>
    <xs:field xpath="@id_coditempedido"/>
</xs:key>
```

```
      <xs:key name="kitem">
        <xs:selector xpath=".//item"/>
        <xs:field xpath="@id_codigoitem"/>
      </xs:key>
      <xs:unique name="uservico">
        <xs:selector xpath=".//item"/>
        <xs:field xpath="@id_ref"/>
      </xs:unique>
      <xs:unique name="uproduto">
        <xs:selector xpath=".//item"/>
        <xs:field xpath="@id_codigo"/>
      </xs:unique>
    </xs:element>
</xs:schema>
```

Anexo 3 – *XML Schema* gerado a partir do esquema lógico de pior caso

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="root">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="parceiro" minOccurs="0" maxOccurs="54">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="papel">
                <xs:complexType>
                  <xs:choice>
                    <xs:element name="distribuidora">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="distribuicao" minOccurs="0" maxOccurs="86">
                            <xs:complexType>
                              <xs:attribute name="ref_codigo"/>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="transportadora">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="entrega" minOccurs="0" maxOccurs="216">
                            <xs:complexType>
                              <xs:attribute name="data"/>
                              <xs:attribute name="ref_coditemnf"/>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="fornecedor">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="leilao" minOccurs="0" maxOccurs="5">
                            <xs:complexType>
```

```
                                    <xs:attribute name="datafim"/>
                                    <xs:attribute name="datainicio"/>
                                    <xs:attribute name="ref_nump"/>
                                 </xs:complexType>
                              </xs:element>
                           </xs:sequence>
                        </xs:complexType>
                     </xs:element>
                     <xs:element name="cliente"/>
                  </xs:choice>
               </xs:complexType>
            </xs:element>
            <xs:element name="contatos" maxOccurs="5">
               <xs:complexType>
                  <xs:attribute name="email"/>
                  <xs:attribute name="nome"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="notafiscal" minOccurs="0" maxOccurs="50">
               <xs:complexType>
                  <xs:attribute name="id_serie"/>
                  <xs:attribute name="id_numero"/>
                  <xs:attribute name="ref_nump"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="pedido" minOccurs="0" maxOccurs="23">
               <xs:complexType>
                  <xs:choice>
                     <xs:element name="pedidocompra"/>
                     <xs:element name="pedidovenda"/>
                  </xs:choice>
                  <xs:attribute name="id_nump"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         <xs:attribute name="id_cnpj"/>
         <xs:attribute name="nome"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="item" minOccurs="0" maxOccurs="360">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="itempedido" minOccurs="0" maxOccurs="14">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="itemestoque" minOccurs="0" maxOccurs="2">
                        <xs:complexType>
                           <xs:attribute name="ref_produto"/>
                           <xs:attribute name="localizacao" use="optional"/>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
                  <xs:attribute name="qtd"/>
                  <xs:attribute name="precouni"/>
                  <xs:attribute name="id_coditempedido"/>
                  <xs:attribute name="ref_nump"/>
               </xs:complexType>
            </xs:element>
```

```
            <xs:element name="itemnf" minOccurs="0" maxOccurs="15">
               <xs:complexType>
                  <xs:attribute name="qtd"/>
                  <xs:attribute name="precouni"/>
                  <xs:attribute name="id_coditemnf"/>
                  <xs:attribute name="ref_serie"/>
                  <xs:attribute name="ref_numero"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="tipo">
               <xs:complexType>
                  <xs:choice>
                     <xs:element name="produto">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element name="cuidados" minOccurs="0" maxOccurs="5"/>
                           </xs:sequence>
                           <xs:attribute name="id_codigo"/>
                           <xs:attribute name="nome"/>
                        </xs:complexType>
                     </xs:element>
                     <xs:element name="servico">
                        <xs:complexType>
                           <xs:attribute name="id_ref"/>
                           <xs:attribute name="descricao"/>
                        </xs:complexType>
                     </xs:element>
                  </xs:choice>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         <xs:attribute name="id_codigoitem"/>
         <xs:attribute name="precobase"/>
      </xs:complexType>
   </xs:element>
</xs:sequence>
</xs:complexType>
<xs:keyref name="rtransportadora" refer="kitemnf">
   <xs:selector xpath=".//entrega"/>
   <xs:field xpath="@ref_coditemnf"/>
</xs:keyref>
<xs:keyref name="rleilao" refer="kpedido">
   <xs:selector xpath=".//leilao"/>
   <xs:field xpath="@ref_nump"/>
</xs:keyref>
<xs:keyref name="rnotafiscal" refer="kpedido">
   <xs:selector xpath=".//notafiscal"/>
   <xs:field xpath="@ref_nump"/>
</xs:keyref>
<xs:keyref name="ritemnf" refer="knotafiscal">
   <xs:selector xpath=".//itemnf"/>
   <xs:field xpath="@ref_serie"/>
   <xs:field xpath="@ref_numero"/>
</xs:keyref>
<xs:keyref name="ritempedido" refer="kpedido">
   <xs:selector xpath=".//itempedido"/>
   <xs:field xpath="@ref_nump"/>
</xs:keyref>
```

```
      <xs:keyref name="ritemestoque" refer="kproduto">
        <xs:selector xpath=".//itemestoque"/>
        <xs:field xpath="@ref_codigo"/>
      </xs:keyref>
      <xs:keyref name="rdistribuidora" refer="kproduto">
        <xs:selector xpath=".//distribuicao"/>
        <xs:field xpath="@ref_codigo"/>
      </xs:keyref>
      <xs:key name="knotafiscal">
        <xs:selector xpath=".//notafiscal"/>
        <xs:field xpath="@id_serie"/>
        <xs:field xpath="@id_numero"/>
      </xs:key>
      <xs:key name="kitemnf">
        <xs:selector xpath=".//itemnf"/>
        <xs:field xpath="@id_coditemnf"/>
      </xs:key>
      <xs:key name="kparceiro">
        <xs:selector xpath=".//parceiro"/>
        <xs:field xpath="@id_cnpj"/>
      </xs:key>
      <xs:key name="kpedido">
        <xs:selector xpath=".//pedido"/>
        <xs:field xpath="@id_nump"/>
      </xs:key>
      <xs:key name="kitempedido">
        <xs:selector xpath=".//itempedido"/>
        <xs:field xpath="@id_coditempedido"/>
      </xs:key>
      <xs:key name="kitem">
        <xs:selector xpath=".//item"/>
        <xs:field xpath="@id_codigoitem"/>
      </xs:key>
      <xs:key name="kservico">
        <xs:selector xpath=".//servico"/>
        <xs:field xpath="@id_ref"/>
      </xs:key>
      <xs:key name="kproduto">
        <xs:selector xpath=".//produto"/>
        <xs:field xpath="@id_codigo"/>
      </xs:key>
    </xs:element>
</xs:schema>
```

Anexo 4 – Consultas *XQuery* utilizadas sobre documentos XML do esquema convencional

Operação	**Script *XQuery***

O1	for $p in input()//pedido
	where $p/@id_nump="id_nump779"
	return <pedido numero={$p/@id_nump}>{
	for $n in input()//notafiscal where $p/@id_nump=$n/@ref_nump return $n
	}</pedido>

O2	for $p in input()//pedido where $p/@id_nump="id_nump0" return <Pedido numero={$p/@id_nump} parceiro_nome={$p/../@nome}

	parceiro_cnpj={$p/../@id_cnpj}> {
	for $i in input()//item where $i/@id_codigoitem=$p/itempedido/@ref_codigoitem
	return <itempedido precounitario={$p/itempedido/@precouni}
	item={$i/@id_codigoitem}/>
	}</Pedido>

O3	for $ip in input()//itempedido, $i in input()//item
	where $ip/@id_coditempedido="id_coditempedido1" and
	$ip/@ref_codigoitem=$i/@id_codigoitem
	return <ItemPedido codigoitempedido={$ip/@id_coditempedido}
	nomeproduto={$i/produto/@nome} codigoproduto={$i/produto/@id_codigo}> {
	for $d in input()//parceiro where $i/produto/distribuicao/@ref_cnpj=$d/@id_cnpj
	return <Distribuidora nome={$d/@nome}/>
	}{ for $e in input()//itemestoque where $e/@ref_codigo=$i/produto/@id_codigo
	return <ItemEstoque localizacao={$e/@localizacao}/>
	}</ItemPedido>

O4	for $l in input()//leilão, $p in input()//pedido where $l/@ref_nump="id_nump307" and $l/@ref_nump=$p/@id_nump return <Leilão pedidonumero={$p/@id_nump} nomeparceiro={$p/../@nome}>{$p/itempedido}</Leilão>
O5	for $p in input()//parceiro where $p/@id_cnpj="id_cnpj1" return <Parceiro cnpj={$p/@id_cnpj} nome={$p/@nome}>{$p/notafiscal}</Parceiro>
O6	for $ie in input()//itemestoque where $ie/@ref_codigo="id_codigo331" return <ItemEstoque localizacao={$ie/@localizacao} coditempedido={$ie/../@id_coditempedido} numeropedido={$ie/../../@id_nump}/>

Anexo 5 – Consultas *XQuery* utilizadas sobre documentos XML do esquema otimizado

Operação	**Script *XQuery***
O1	for $p in input()//pedido where $p/@id_nump="id_nump779" return <pedido numero={$p/@id_nump}>{$p/notafiscal}</pedido>

O2	for $p in input()//pedido
	where $p/@id_nump="id_nump0"
	return <Pedido numero={$p/@id_nump} parceiro_nome={$p/../@nome}
	parceiro_cnpj={$p/../@id_cnpj}>{
	for $i in input()//item
	where $i/@id_codigoitem=$p/itempedido/@ref_codigoitem
	return <itempedido precounitario={$p/itempedido/@precouni}
	item={$i/@id_codigoitem}/>
	}</Pedido>

O3	for $ip in input()//itempedido, $i in input()//item
	where $ip/@id_coditempedido = "id_coditempedido1" and
	$ip/@ref_codigoitem=$i/@id_codigoitem
	return <ItemPedido codigoitempedido= {$ip/@id_coditempedido} nomeproduto=
	{$i/@nome} codigoproduto= {$i/@id_codigo}>{
	for $d in input()//parceiro where $i/distribuicao/@ref_cnpj=$d/@id_cnpj
	return <Distribuidora nome={$d/@nome}/>}
	{$i/itemestoque}</ItemPedido>

O4	for $p in input()//pedido where $p/@id_nump="id_nump307" return <Leilão pedidonumero={$p/@id_nump} nomecliente={$p/../@nome}>{$p/itempedido}</Leilão>
O5	for $p in input()//parceiro where $p/@id_cnpj="id_cnpj1"

	return <Parceiro cnpj={$p/@id_cnpj} nome={$p/@nome}>{
	for $n in input()//notafiscal where $p/@id_cnpj=$n/@ref_cnpj return $n}
	</Parceiro>

O6	for $ie in input()//itemestoque, $ip in input()//itempedido where $ie/@ref_coditempedido="id_coditempedido0" and $ip/@id_coditempedido=$ie/@ref_coditempedido return <ItemEstoque localizacao={$ie/@localizacao} coditempedido={$ip/@id_coditempedido} numeropedido={$ip/../@id_nump}/>

Anexo 6 – Consultas *XQuery* utilizadas sobre documentos XML do esquema de pior caso

Operação	**Script *XQuery***

O1	for $p in input()//pedido
	where $p/@id_nump="id_nump0"
	return <pedido numero={$p/@id_nump}>{
	for $n in input()//notafiscal where $p/@id_nump=$n/@ref_nump
	return $n}
	</pedido>

O2	for $p in input()//pedido
	where $p/@id_nump="id_nump0"
	return <Pedido numero={$p/@id_nump} parceiro_nome={$p/../@nome}
	parceiro_cnpj={$p/../@id_cnpj}>{
	for $ip in input()//itempedido where $ip/@ref_nump=$p/@id_nump
	return <itempedido precounitario={$ip/@precouni}
	item={$ip/../@id_codigoitem}/>
	}</Pedido>

O3	for $ip in input()//itempedido
	where $ip/@id_coditempedido="id_coditempedido4003"
	return <ItemPedido codigoitempedido={$ip/@id_coditempedido} nomeproduto=
	{$ip/../tipo/produto/@nome} codigoproduto={$ip/../tipo/produto/@id_codigo}> {
	for $d in input()//distribuicao
	where $d/@ref_codigo=$ip/../tipo/produto/@id_codigo
	return <Distribuidora nome={$d/../../../@nome}/> }{
	for $e in input()//itemestoque
	where $e/@ref_codigo=$ip/../tipo/produto/@id_codigo
	return <ItemEstoque localizacao={$e/@localizacao}/>
	} </ItemPedido>

O4	for $l in input()//leilao, $p in input()//pedido
	where $l/@ref_nump="id_nump979" and $l/@ref_nump=$p/@id_nump
	return <Leilão pedidonumero={$p/@id_nump} nomeparceiro={$p/../@nome}>{
	for $ip in input()//itempedido where $ip/@ref_nump=$p/@id_nump return $ip
	}</Leilão>

O5	for $p in input()//parceiro
	where $p/@id_cnpj="id_cnpj1"
	return <Parceiro cnpj={$p/@id_cnpj} nome={$p/@nome}>{
	for $nf in $p/notafiscal return <NotaFiscal numero={$nf/@numero}>{
	for $in in input()//itemnf
	where $in/@ref_numero=$nf/@id_numero and $in/@ref_serie=$nf/@id_serie
	return $in}</NotaFiscal>
	}</Parceiro>

O6	for $ie in input()//itemestoque, $p in input()//pedido where $ie/@ref_codigo="id_codigo62" and $ie/../@ref_nump=$p/@id_nump return <ItemEstoque localizacao={$ie/@localizacao} coditempedido={$ie/../@id_coditempedido} numeropedido={$p/@id_nump}/>

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo